



IBM i

e-business and Web Serving

IBM HTTP Server for i

7.1







IBM i

e-business and Web Serving  
IBM HTTP Server for i

*7.1*

**Note**

Before using this information and the product it supports, read the information in "Notices," on page 509.

| This edition applies to IBM i 7.1 (product number 5770-SS1) and to all subsequent releases and modifications until  
| otherwise indicated in new editions. This version does not run on all reduced instruction set computer (RISC)  
| models nor does it run on CISC models.

© Copyright IBM Corporation 1997, 2010.

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

---

# Contents

## IBM HTTP Server for i . . . . . 1

What's new for IBM i 7.1 . . . . .	1
PDF file for IBM HTTP Server for i . . . . .	1
Installing HTTP Server . . . . .	2
Compatibility considerations . . . . .	2
Verify the prerequisites . . . . .	2
Install HTTP Server on your server . . . . .	3
Verify the HTTP Server installation . . . . .	4
Overview of IBM Web Administration for i . . . . .	4
Web browser requirements . . . . .	4
User profile requirements to use the Web Administration for i interface . . . . .	5
Starting Web Administration for i . . . . .	7
User interface conventions . . . . .	7
Configuring SSL for ADMIN wizard . . . . .	11
HTTP Server Concepts . . . . .	13
Fundamental directive, context, and server area concepts on HTTP Server . . . . .	13
Content negotiation for HTTP Server . . . . .	17
Virtual hosts on HTTP Server . . . . .	23
Proxy server types and uses for HTTP Server . . . . .	24
Supported file systems for Web content served by HTTP Server . . . . .	26
Logging . . . . .	27
Security . . . . .	29
Performance . . . . .	31
Extending HTTP Server functionality . . . . .	39
High availability . . . . .	43
Web Publishing with the PUT Method . . . . .	48
WebDAV for HTTP Server . . . . .	49
Scenarios: HTTP Server . . . . .	49
JKL Toy Company creates an HTTP Server . . . . .	50
JKL Toy Company adds a new directory to HTTP Server . . . . .	52
JKL Toy Company adds user directories for HTTP Server . . . . .	54
JKL Toy Company enables cookie tracking on HTTP Server . . . . .	58
JKL Toy Company creates virtual hosts on HTTP Server . . . . .	61
JKL Toy Company adds password protection for HTTP Server . . . . .	64
JKL Toy Company adds dynamic content with server-side includes for HTTP Server . . . . .	68
JKL Toy company enables Secure Sockets Layer (SSL) protection on HTTP Server . . . . .	70

JKL Toy Company enables single signon for HTTP Server . . . . .	75
JKL Toy Company monitors Web server activity with logs on HTTP Server . . . . .	91
Tasks . . . . .	94
Getting started with the IBM Web Administration for i interface . . . . .	94
HTTP Server tasks . . . . .	95
Compression tasks . . . . .	108
Fast Response Cache Accelerator tasks . . . . .	109
Log and log file tasks . . . . .	112
Proxy tasks . . . . .	115
Security tasks . . . . .	117
WebDAV tasks . . . . .	121
Web tasks . . . . .	122
Virtual host tasks . . . . .	132
CGI tasks . . . . .	135
Apache module tasks . . . . .	136
Programming . . . . .	136
Application Programming Interface . . . . .	137
CGI programming . . . . .	182
Apache module programming . . . . .	198
Handler for HTTP Server . . . . .	200
Server-side scripting languages . . . . .	200
Running Java Web applications . . . . .	202
Troubleshooting . . . . .	202
Troubleshooting Web Administration for i . . . . .	202
Troubleshooting HTTP Server . . . . .	206
Troubleshooting CGI programs . . . . .	211
Reference information for HTTP Server . . . . .	214
Directives for HTTP Server . . . . .	214
Log file format tokens . . . . .	489
Regular expression notation for HTTP Server . . . . .	491
CL commands for HTTP Server . . . . .	492
Environment variables set by HTTP Server . . . . .	493
Server-side include commands for HTTP Server . . . . .	502
Time formats for HTTP Server . . . . .	505
Related information for HTTP Server . . . . .	507
Legal notices for Apache Software Foundation on HTTP Server . . . . .	508

## Appendix. Notices . . . . . 509


Programming interface information . . . . .	510
Trademarks . . . . .	511
Terms and conditions . . . . .	511




---

## IBM HTTP Server for i

The IBM® HTTP Server for i is a Web server implementation that is based on the open-source server code provided by the Apache Software Foundation and that is optimized for the IBM i environment. With the IBM HTTP Server for i, you have everything you need to quickly and easily establish a Web presence.


The IBM HTTP Server for i documentation contains getting started, task oriented, and scenario-based information, supporting reference material, and conceptual information. Information for the IBM Web Administration for i interface is also included. See the IBM HTTP Server for i  Web site for additional product information.


**Important:** Information for this topic supports the latest PTF levels for IBM HTTP Server for i. It is recommended that you install the latest PTFs to upgrade to the latest level of the IBM HTTP Server for i. See the IBM HTTP Server for i Support  Web page for more information.

---

### What's new for IBM i 7.1

Read about new or significantly changed information for the IBM HTTP Server for i topic collection.

**Important:** Information for this topic supports the latest PTF levels for IBM HTTP Server for i. It is recommended that you install the latest PTFs to upgrade to the latest level of the IBM HTTP Server for i. See the IBM HTTP Server for i Support  Web page for more information.



See the HTTP Server: What's New  topic for a list of recent enhancements made to the IBM HTTP Server for i.

| The following changes have been made to IBM HTTP Server for i in IBM i 7.1:

- | • The IBM Web Administration for i has been enhanced so that users without \*ALL0BJ special authority  
| can use many of its features, such as managing and creating different types of servers. For more  
| information about this enhancement, see "User profile requirements to use the Web Administration for  
| i interface" on page 5.

### How to see what's new or changed

To help you see where technical changes have been made, the information center uses:

- The  image to mark where new or changed information begins.
- The  image to mark where new or changed information ends.

To find other information about what's new or changed this release, see the Memo to users.

---

### PDF file for IBM HTTP Server for i

You can view and print a PDF file of this information.

To view or download the PDF version of this document, select IBM HTTP Server for i (about 2300 KB).


### Saving PDF files

To save a PDF on your workstation for viewing or printing:

1. Right-click the PDF link in your browser.

2. Click the option that saves the PDF locally.
3. Navigate to the directory in which you want to save the PDF.
4. Click **Save**.

## Downloading Adobe Reader

You need Adobe Reader installed on your system to view or print these PDFs. You can download a free copy from the Adobe Web site ([www.adobe.com/products/acrobat/readstep.html](http://www.adobe.com/products/acrobat/readstep.html)) .


---

## Installing HTTP Server

This topic provides information about how to install the IBM HTTP Server for i, which includes the support for the IBM Web Administration for i interface.

## Compatibility considerations

This topic describes considerations when you are moving from an earlier release of IBM i to the most current release, or you are moving from an earlier HTTP Server version of IBM HTTP Server for i to a newer version.

You should read about any compatibility issues by reading HTTP Server compatibility information  on the HTTP Server home page. Before reading the information, you will need to determine the HTTP Server version you are currently using. Use one of the following methods to determine the HTTP Server version:

- Use the "-V" option on the Start TCP/IP Server (STRTCPSVR) command. For example, if HTTP Server (Apache 2.2.11) is installed, STRTCPSVR SERVER(\*HTTP) HTTPSVR(APACHEDFT '-V') displays:  
Server version: Apache/2.2.11 (i5)  
Server built: Jul 21 2009 19:24:31
- From the IBM Web Administration for i, select **Manage an HTTP server**. In the server introduction, the Apache version level is included in the header. For example:  
Manage Apache server "WEBSERVER" - Apache/2.2.11

## Verify the prerequisites

Before you begin your installation, use this information to ensure that you meet all the hardware, software, and system requirements for installing IBM HTTP Server for i.

## Hardware requirements

You need a communication hardware adapter that is supported by the TCP/IP protocol stack.

## Software requirements

The following licensed programs must be installed on your system:

- Extended Base Directory Support (5770-SS1 Option 3)
- Host Servers (5770-SS1 Option 12)
- Qshell (5770-SS1 Option 30)
- IBM Portable Application Solutions Environment for i (5770-SS1 Option 33)
- TCP/IP Utilities (5770-TC1)
- IBM Developer Kit for Java™ (5761-JV1 Option 11).

The following software products may need to be installed depending on your needs:

- **WebSphere® Application Server**



If you plan to use WebSphere Application Server with the HTTP Server, install a version of the WebSphere Application Server Apache plug-in that is compatible with your current level of HTTP Server. If the proper WebSphere Application Server PTFs are not loaded, the mismatch will prevent the

HTTP Server from starting. See the WebSphere Application Server for IBM i  product Web page for information about the latest WebSphere Application Server and WebSphere Application Server Apache plug-in PTFs.


- **Digital Certificate Manager**

In order to provide the required support for handling digital server certificates used by Secure Sockets Layer (SSL) for secure Web serving, you must install IBM i Digital Certificate Manager (5770-SS1 Option 34).

- **HA Switchable Resources**

If you want to configure a high availability Web server cluster, then you need to install HA Switchable Resources (5770-SS1 Option 41), or use a business partner tool to manage clusters.

- **Zend Core for IBM i**

If you want to run PHP scripts, you will need the PHP Zend Core PHP runtime and any software that is required by the Zend Core for IBM i product. See the Zend and IBM i  product Web page for information about Zend Core for IBM i.

## System configuration settings

Perform or verify the following configuration settings:

- Ensure at least one TCP/IP interface is available and active. You can use the Work with TCP/IP Network Status (NETSTAT) command to see a list of TCP/IP interfaces. For example:

```
NETSTAT OPTION(*IFC)
```

**Note:** You can add TCP/IP interfaces using the Add TCP/IP Interface (ADDTCPIFC). You can start TCP/IP interfaces using the Start TCP/IP Interface (STRTCPIFC) command.


- Ensure the system TCP/IP host and domain name information is set. You can use the Change TCP/IP Domain (CHGTCPDMN) command to set TCP/IP domain information.
- Ensure that LOCALHOST is in the TCP/IP host table. You can use the Configure TCP/IP (CFGTCP) command to display a menu that allows a user to define or change TCP/IP configuration settings.
- Ensure that the Share Memory Control (QSHRMEMCTL) system value is set to 1.

## Install HTTP Server on your server

Follow these steps to install IBM HTTP Server for i on your IBM i server.

Before installing IBM HTTP Server for i, you need to ensure that your server meets all the hardware and software prerequisites. In addition, you should be aware of any compatibility issues.

To install IBM HTTP Server for i (5770-DG1) on your IBM i server, complete the following steps:

1. Insert the installation media for HTTP Server into your system.
2. At the IBM i command line, type G0 LICPGM and press **Enter**.
3. Select option 11 (Install licensed programs) on the Work with Licensed Programs display to see a list of licensed programs.
4. Select and install IBM HTTP Server for i (5770-DG1). See the Software installation process for help with licensed program installation.
5. Load and apply the latest HTTP Server group PTF .

The IBM HTTP Server for i licensed program is now installed with the latest fixes. You are now ready to verify the installation.

## Verify the HTTP Server installation

To verify that you have successfully completed the IBM HTTP Server for i installation, follow these steps.

Before you can verify the IBM HTTP Server for i installation it is assumed you have installed the licensed program. For more information about installing the product, see "Install HTTP Server on your server" on page 3.

The HTTP Server is installed with a default server called APACHEDFT. To test your installation, do the following:

1. Access the IBM Web Administration for i from your browser. For information about how to access the Web Administration for i interface, see "Starting Web Administration for i" on page 7.
2. Click the **Manage** tab.
3. Click the **HTTP Servers** subtab.
4. Select the APACHEDFT server from the **Server list**.
5. Click the **Start icon** next to the **Server list**.
6. Click the Refresh icon and check if the server status is still shown as "Running". If your HTTP Server does not start, see "Troubleshooting" on page 202.
7. Open another Web browser and go to `http://your.server.name` where *your.server.name* is the host name of your IBM i server to view the default Welcome page. The default Welcome page is a Web page that is returned by the APACHEDFT Web server.

Upon successful completion of these steps you will have verified the installation of IBM HTTP Server for i.

---

## Overview of IBM Web Administration for i

The HTTP Server and other Web applications can be managed through the IBM Web Administration for i interface. The Web Administration for i is an application that is loaded in the HTTP Administration server, and accessed from a Web browser.

One of the key differences between IBM HTTP Server for i and other Web server products is the graphical user interface (GUI) provided for setting up and managing your servers. The Web Administration for i interface combines forms, tools, and wizards to create a simplified environment to set up and manage many different servers on your system. The Web Administration for i interface is rich in function, examples, error-checking, and ease-of-use.

Using the Web Administration for i interface, it is no longer necessary to memorize directive names and their proper usage or syntax. Directives are represented in the interface by descriptive field names, along with help text for every field. For Apache users, it is no longer necessary to memorize the supported context of directives. The Web Administration for i enforces supported context for the directives.

The Web Administration for i supports several wizards that guide you through a series of advanced steps to accomplish a task. With a few clicks of a button, you can have a Web server or application server running in no time at all. The Web Administration for i supports the creation of many types of servers, including Web servers, and application servers such as WebSphere Application Server for System i®, WebSphere Portal Server, IBM Integrated Web Application Server for i, and IBM Integrated Web Services Server for i.

## Web browser requirements

To use the IBM Web Administration for i interface you need a Web browser that supports the HTTP 1.0 or 1.1 protocol, frames, and JavaScript.

Suggested Web browsers include Microsoft Internet Explorer V6.0 or higher and Firefox V3.0 or higher.

**Note:** Consult your Web browser documentation for more information.

## User profile requirements to use the Web Administration for i interface

By default, only users with \*ALLOBJ and \*IOSYSCFG special authorities can manage and create Web-related servers on the system through the use of the IBM Web Administration for i interface. Web-related servers include instances of IBM HTTP Server, WebSphere Application Server, Integrated Application Server, and Integrated Web Services Server. A user without the necessary IBM i special authorities to manage or create Web-related servers requires an administrator to grant that user permission to a server or group of servers.

To be able to access the Web Administration for i interface, the IBM i user profile used to sign on must meet at least one of the following conditions:

- The user profile has \*ALLOBJ and \*IOSYSCFG special authorities.
- The user profile has been granted permission to an entire class of servers, or a specific server.
- The user profile has been granted permission to create servers.

For example, if a user wants to create an HTTP server using the Web Administration for i interface, the user profile must either have \*ALLOBJ and \*IOSYSCFG special authorities, or have permission to create HTTP servers.

Only users with \*ALLOBJ and \*IOSYSCFG special authority are allowed to grant, revoke, or manage user permissions. The granting of permissions to a user profile is done through the Web Administration for i interface by giving user profiles that need to access the Web Administration for i interface roles to specific servers or a class of servers.

**Note:** Granting \*ALLOBJ authority to a user profile or using the QSECOFR user profile to access the Web Administration for i interface is not recommended.

## Roles

*Roles* define a set of permissions that define what operations a user is allowed to perform on a server. The Web Administration for i interface defines the following roles:

### Administrator

Any IBM i user profile with \*ALLOBJ and \*IOSYSCFG special authority is identified with the role of Administrator. An Administrator has unrestricted use of every feature in the Web Administration for i interface, including the ability to manage user permissions. An Administrator cannot be assigned any other role.

**Note:** A user profile cannot be assigned this role.

### Developer

Is allowed to view and modify a server, including the ability to delete a server. A Developer can use Web Performance Monitor and Web Performance Advisor, but cannot change system-wide settings, such as memory pool allocations.

### Operator

Is allowed to view a server, including the capability to start and stop a server. In addition, an Operator is allowed to modify trace settings for a server.

If a user with a role of Developer or Operator has no role assigned to them for a server, they are not allowed to view the server or any of its attributes.

## Permissions

A *permission* is the ability to perform an operation on a server. The ability for a user to perform operations on a server is determined by the role they have been assigned for the server. The Web Administration for i roles are defined with the following permissions:

Table 1. Permissions corresponding to each role.

Permissions	Roles		
	Administrator	Developer	Operator
Start/Stop server	x	x	x
Delete server	x	x	
Install/Remove applications	x	x	
Install/Remove Web services <sup>Note 1</sup>	x	x	
Start/Stop applications	x	x	x
Start/Stop Web services <sup>Note 1</sup>	x	x	x
Modify server attributes	x	x	
Modify application attributes	x	x	
Create database connections	x	x	
Delete database connections	x	x	
Modify server tracing	x	x	
Use Web Performance Advisor	x	x	
Use Web Performance Monitor	x	x	
Use Web Log Monitor	x	x	
Create server <sup>Note 2</sup>	x		

- Notes:**
1. Web services deployed within integrated Web services servers.
  2. An administrator granting permissions to a user profile needs to explicitly grant the create-server permission.

Only an Administrator can grant permissions. The granting of permissions to a user profile is done through the Web Administration for i interface by giving user profiles that need to access the Web Administration for i interface roles to specific servers or a class of servers.

**Note:** If a user creates a server, they are automatically assigned the role of Developer to the newly created server.

Permissions can be granted to a specific server or to all servers of a certain type. The Web Administration for i interface supports granting permissions to the following types of servers:

- Integrated Web Application Servers
- Integrated Web Services Servers
- WebSphere Application Servers
- HTTP Servers

When granting permissions, you should be aware of the following points:

- If you grant a user permission to create an application server or Web services server, then you must also grant the user permission to create HTTP Servers. This is due to the association between an HTTP Server and the application server or Web services server.
- If you grant a user permissions to an application server or Web services server, and you do not explicitly grant the user permissions to the associated HTTP Server(s), the user is automatically granted

the same permissions to the associated HTTP Servers(s). This is also true in reverse. If you grant a user permissions to an HTTP Server, and you do not explicitly grant the user permissions to the associated application server or Web services server, the user is automatically granted the same permissions to the associated application server or Web services server.

**Note:** A warning message is displayed on the Web Administration for i interface when permissions are implicitly granted to a user.

- If you attempt to grant a user different permissions to an HTTP Server and the associated application server or Web services server, the user is granted the higher permission and both servers get assigned that permission.

**Note:** A warning message is displayed on the Web Administration for i interface when permissions to servers are upgraded.

If a user has no permissions to any servers, and no permission to create any type of server, then the user is not allowed to access the Web Administration for i interface.

## Starting Web Administration for i

The Web Administration for i allows you to create and manage different types of servers, including Web servers and application servers. Complete the following steps to start the Web Administration for i interface.

It is assumed that you have met the user profile requirements to access the Web Administration for i interface.

To start the Web Administration for i interface, complete the following steps:

**Note:** Enter your user profile name and password when prompted.

1. Start the HTTP Administration server.
  - a. In System i Navigator, expand **your\_system** > **Network** > **Servers**, and select **TCP/IP**.
  - b. Right-click **HTTP Administration**, and select **Start**.

**Note:** The administration server can also be started using the STRTCPSVR SERVER(\*HTTP) HTTPSVR(\*ADMIN) command at an IBM i command prompt.

2. Bring up the IBM Navigator for i by accessing the following URL from a Web browser where *your\_system* is your IBM i server host name:  
`http://your_system:2001`
3. From the IBM Navigator for i welcome page click the **IBM i Tasks Page** link.
4. Click the **IBM Web Administration for i** link.

From here, you can create different types of servers or work with existing servers, depending on your needs.

**Note:** If the Web Administration for i interface does not start, see “Troubleshooting” on page 202.


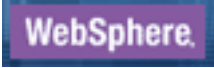


## User interface conventions

This topic describes the conventions used by the IBM Web Administration for i interface when displaying information to a user.

### Header images

The Web Administration for i interface has several images in the header, or top most portion, of the GUI. These images are hyperlinks to helpful information.

Table 2. Header images

Header Image	Description
	Image hyperlink to the IBM i Information Center entry page.
	Image hyperlink to the WebSphere Application Server Family Web page. This Web page contains information on WebSphere products, including support and service information.
	Image hyperlink to the IBM Web page where you can find information on all of IBM's products.
	Image hyperlink to the IBM HTTP Server for i Web page. This Web page contains additional information on PTFs and support, developer documentation, and other topics.

## Tabs and subtabs

Navigation of the Web Administration for i interface is done through tabs. There are two types of tabs, the main task tabs on the top (referred to in the documentation as *tabs*) and more specific subtabs underneath (referred to in the documentation as *subtabs*).

Table 3. Main task tabs

Tab Name	Description
Setup	The <i>Setup</i> tab contains the setup tasks for your servers. Setup tasks include the common tasks and wizards for the Web Administration for i interface.
Manage	The <i>Manage</i> tab contains tasks to manage your servers. The All Servers, HTTP Server, and Application Server subtabs are available under the Manage tab. You can manage all servers, or choose a specific server to manage on your IBM i server.
Advanced	The <i>Advanced</i> tab contains advanced tasks that you can perform on your servers. The advanced tasks include global settings for your IBM i server, Internet Users and Groups management, and the management of permissions to servers.
Related Links	The <i>Related Links</i> tab contains hyperlinks to useful information related to features, functions, and uses of the Web Administration for i interface and all products supported by the interface. From here, you can find general and support documentation.

Use the subtabs to quickly manage your servers or to set up advanced tasks.

Table 4. Manage subtabs

Subtab Name	Description
All Servers	The <i>All Servers</i> subtab opens a form to view all the currently configured servers on your system. This form also provides you the ability to start, stop, restart, and configure your servers, as well as monitor and manage details. Select the server that you want to work with by clicking the button to the left of the server name. Clicking on the server name will also take you directly to managing the details for the selected server.
HTTP Servers	The <i>HTTP Servers</i> subtab opens forms for managing HTTP Servers currently configured on your system. Use these forms to set up and manage your HTTP Server quickly and easily.

Table 4. Manage subtabs (continued)

Subtab Name	Description
Application Servers	The <i>Application Servers</i> subtab opens forms for managing application server currently configured on your system including: WebSphere Application Servers, WebSphere Portal servers, integrated Web application server, and Web services servers.

Table 5. Advanced subtabs

Subtab Name	Description
Settings	<p>The <i>Settings</i> subtab displays a Web page containing links to forms for managing your global server settings.</p> <p>Global server settings are values that apply to each IBM HTTP Server for i configuration. The values provided here can be overridden individually within each HTTP Server configuration file.</p>
Internet Users and Groups	<p>The <i>Internet Users and Groups</i> subtab displays a Web page containing links to forms for managing validation lists, group files, and digital certificates.</p> <p><i>Validation lists</i> are used in conjunction with other resources to limit access to server resources. Each validation list contains a list of Internet users and passwords. Use the Internet users and groups form to list and manage digital certificates associated with validation lists. Validation list entries also require you to identify an authentication protocol type to associate with the user id and password. Validation lists are case-sensitive and reside in IBM i libraries. A validation list is used to store user ID and password information about remote users. You can use existing validation lists or create your own.</p> <p>A <i>group file</i> identifies a group of users with a common security profile. A group file contains IBM i user profiles. A user profile is an object with a unique name that contains the user's password, the list of special authorities assigned to a user, and the objects the user owns or has access to.</p> <p>A <i>digital certificate</i> is a form of personal identification that can be verified electronically. Only the certificate owner who holds the corresponding private key can present a certificate for authentication through a Web browser session. The key can be validated through any readily available public key. Use the Digital Certificate Manager to create, distribute, and manage digital certificates.</p>
Permissions	<p>The <i>Permissions</i> subtab displays a Web page containing links to forms for managing and adding permissions.</p> <p>A permission is the ability to perform an operation on a server. The ability for a user to perform operations on a server is determined by the role they have been assigned for the server by a Web administrator. For more information, see "User profile requirements to use the Web Administration for i interface" on page 5.</p>

**Note: Common Tasks and Wizards** are available on all tabs of the interface.

## Lists

The Web Administration for i interface organizes large groupings of servers and configuration files into different lists. Click the list and select the server or server area you want to work with.

Table 6. Lists

List Name	Description
Server	The <i>Server list</i> contains the name of every server currently configured on your system. This includes HTTP Servers, integrated Web application server, Web services server, WebSphere Application Servers, WebSphere Application Servers - Express, and WebSphere Portal servers. The server list only shows the servers for the selected type. For example, if the subtab <b>HTTP Servers</b> is selected, the servers list will show only HTTP Servers, not WebSphere Application Servers.
Server area	The <i>Server area</i> allows you to work with the individual containers within your HTTP configuration.

## Tasks, wizards, property forms, and tools

Each subtab opens specific tasks, wizards, property forms, and tools that provide you the ability to configure and manage your server.

Table 7. Tasks, Wizards, and Property Forms

Name	Description
Task	<i>Tasks</i> are property forms that guide you through advanced configuration steps. Individual tasks are sometimes grouped together to form advanced configuration tasks.
Wizards	<i>Wizards</i> guide you through a series of advanced steps to accomplish a task. Wizards cannot save your progress and must be completed to successfully update or create a server.
Property forms	<i>Property forms</i> are forms with field values that may be set for specific configuration requirements. Each property form has help text to assist you in managing your servers.
Tools	<i>Tools</i> provide easy access to log files, the server configuration file, directive index, and real time HTTP server statistics. Tools are useful for problem solving and server maintenance.

**Note:** The Web Administration for i checks any changes you make for errors. A message will be displayed below the forms (in the error window) detailing any errors.

## Server status

The Web Administration for i interface shows you the current status of your servers. The status of the server is displayed with the following icons.

Table 8. Server states






Server State	Description
 Stopped	<b>Stopped.</b> The server is currently stopped. The server is no longer available. The IP address and port number are not in use.









Table 8. Server states (continued)

Server State	Description
 Running	<b>Running.</b> The server is currently running. The IP address and port number are in use.
 Stopping	<b>Stopping.</b> The server is attempting to stop. the IP address and port number are still in use.
 Creating	<b>Creating.</b> The server is being configured and created. The IP address and port number are not in use.
 Loading...	<b>Loading.</b> The Web Administration for i interface is loading the selected form, wizard, or Web browser frame.

## Server buttons

The Web Administration for i interface uses server stop, start, and restart buttons to manage your server's status. Use the following buttons to change your server's status at anytime.

Table 9. Buttons

Button	Description
 	<b>Start.</b> Click this button to start the server you are currently working with. The button is gray and unavailable when the server is running.
 	<b>Stop.</b> Click this button to stop the server you are currently working with. The button is gray and unavailable when the server has stopped.
	<b>Restart.</b> Click this button to restart the server you are currently working with. The restart button stops the server and then attempts to restart it. <b>Note:</b> This option is not available on all subtabs.
	<b>Refresh.</b> Click this button to refresh the Web Administration for i interface display. Some changes made by property forms may not be readily displayed. The refresh button clears the Web Administration for i interface display and updates the current server status.

## Configuring SSL for ADMIN wizard

The IBM Web Administration for i interface provides the Configure SSL for ADMIN wizard to configure Secure Sockets Layer (SSL) for the ADMIN server. SSL has become an industry standard for enabling applications for secure communication sessions over an unprotected network, such as the Internet.

The ADMIN server runs all of the programs listed on the IBM i Tasks page ([http://\[your\\_isystem\]:2001](http://[your_isystem]:2001)) including the Web Administration for i and the Digital Certificate Manager (DCM). By default, the ADMIN server listens on a non-SSL (non-secure) connection over port 2001. If you want to configure the

ADMIN server to use secure communications over SSL, but lack experience with DCM and SSL, the wizard simplifies the process and removes the need to manually configure the ADMIN server configuration.

The Configure SSL for Admin wizard updates the ADMIN server configuration file to enable SSL on port 2010; optionally port 2001 may be left enabled for non-SSL traffic. The wizard uses the Digital Certificate Manager to issue a digital certificate, connects the certificate and the ADMIN server, and restarts the ADMIN server. The restart of the ADMIN server usually takes one minute or so. While the restart is being performed, the Web Administration for i interface is unavailable.

## Secure Sockets Layer and digital certificates

SSL is actually two protocols. The protocols are the record protocol and the handshake protocol. The record protocol controls the flow of the data between the two endpoints of an SSL session.

The handshake protocol authenticates one or both endpoints of the SSL session and establishes a unique symmetric key used to generate keys to encrypt and decrypt data for that SSL session. SSL uses asymmetric cryptography, digital certificates, and SSL handshake flows, to authenticate one or both endpoints of an SSL session. Typically, SSL authenticates the server. Optionally, SSL authenticates the client; however, this wizard only authenticates the server, not the client. A digital certificate, issued by a Certificate Authority, can be assigned to each of the endpoints or to the applications using SSL on each endpoint of the connection.

A digital certificate is an electronic credential that you can use to establish proof of identity in an electronic transaction. IBM i provides extensive digital certificate support that allows you to use digital certificates as credentials in a number of security applications. In addition to using certificates to configure SSL, you can use them as credentials for client authentication in both SSL and virtual private network (VPN) transactions. Also, you can use digital certificates and their associated security keys to sign objects. Signing objects allows you to detect changes or possible tampering to object contents by verifying signatures on the objects to ensure their integrity.

Capitalizing on the IBM i support for certificates is easy when you use Digital Certificate Manager (DCM), a free feature, to centrally manage certificates for your applications. DCM allows you to manage certificates that you obtain from any Certificate Authority (CA). Also, you can use DCM to create and operate your own Local CA to issue private certificates to applications and users in your organization.

The digital certificate is comprised of a public key and some identifying information that a trusted Certificate Authority (CA) has digitally signed. Each public key has an associated private key. The private key is not stored with or as part of the certificate. In both server and client authentication, the endpoint which is being authenticated must prove that it has access to the private key associated with the public key contained within the digital certificate.

## Prerequisites and assumptions

The Configure SSL for ADMIN wizard requires a user profile with \*ALLOBJ and \*SECADM special authorities and Digital Certificate Manager installed on your system.

## Start the Configure SSL for Admin wizard

The Configure SSL for ADMIN wizard can be started from the Web Administration for i interface:

1. Access the IBM Web Administration for i from your browser. For information about how to access the Web Administration for i interface, see “Starting Web Administration for i” on page 7.
2. From the IBM Web Administration for i interface, select the **ADMIN-Apache** server.
3. In the navigation pane, expand **HTTP Tasks and Wizards** , and select **Configure SSL for ADMIN**.

**Note:** If **Configure SSL for ADMIN** is not displayed in the navigation pane, either the latest IBM HTTP Server for i (5770-DG1) PTF group has not been properly installed, or the ADMIN server has not been selected.

The Configure SSL for ADMIN welcome page displays. Click **Next** to begin the wizard. After the updates are made, the wizard restarts the ADMIN server. The ADMIN server can be accessed securely at ([https://\[your\\_system\]:2010/HTTPAdmin](https://[your_system]:2010/HTTPAdmin)).

**Related information:**

Secure Sockets Layer (SSL)

Digital Certificate Manager

---

## HTTP Server Concepts

This topic provides conceptual information of the various functions and features of IBM HTTP Server for i.

### Fundamental directive, context, and server area concepts on HTTP Server

The IBM HTTP Server for i is configured using directives. A directive is used to define an attribute of the HTTP Server or how the HTTP Server operates. For example, the Listen directive defines what port the HTTP Server should wait on to handle incoming requests.

HTTP Server directives are extensive, functional, and built around the concept of context.

#### Directive categories of HTTP Server

The HTTP Server directives may be categorized into two main groups. These are Assignment directives and Container directives.

##### Assignment directives

Used to configure the function or characteristics of the HTTP Server. For example, the Listen directive assigns the port the HTTP Server should use to handle incoming requests.

##### Container directives

Used to group directives together within the HTTP Server. The container directives group one or more assignment directives which are used to control the function intended specifically within the context of the container. For example, the <Directory> directive is used to enclose a group of assignment directives that only apply to the directory and subdirectory context.

When dealing with container directives, individual assignment directives may not be valid within one or more container directives due to improper context. See "Directives for HTTP Server" on page 214 for more information on the specific context a directive may or may not be used.

#### HTTP Server directive contexts

Understanding the context concept is necessary to increase the productivity and usefulness of your HTTP Server. The IBM Web Administration for i interface assists in managing context areas of your server. By selecting a different area of the server area, you are changing the context you are managing.

These types of directive contexts are supported:

##### server config

Also called "Server Area", "Global Level" or "Global Context". The attributes set by directives in the server config context can and most likely will be inherited by the container directives and assignment directives used in the configuration.

## directory

Also called "Container Context", the directory context should not be confused with <Directory> containers. If the directive supports this context, the directive can be used in most containers (<Directory>, <File>, <Proxy>, and <Location> for example). This context support does not apply to virtual hosts. There are limited exceptions where directives are not supported in all of the containers associated with this context. See "Directives for HTTP Server" on page 214 for specific directive exceptions.

## virtual host

The virtual host context refers to directives that are allowed to be configured, or assigned, in the <Virtual Host> container directive.

## .htaccess

Also called ".htaccess files", the .htaccess context refers to directives supported in per-directory configuration files. Per-directory configuration files are read by the server from the physical directory where they reside. The directives within this file are applied to any objects that are to be served from the directory where the file exists, and may also be carried forward to sub-directories. Note that the use of .htaccess files is not recommended due to the additional overhead incurred by the server.

## HTTP Server container types

The directives used to configure HTTP Server containers are encased in greater than (>) and lesser than (<) brackets. For example, <Directory> is a container directive. Each container is comprised of an opening directive, such as <Directory>, and closed with the same context directive prefixed with a slash (/). For example, </Directory>.

There are six different types of container directives. Five of the six container directives listed below have variants which results in a total of eleven different container directives (shown below with the opening and closing tags).

### Directory and DirectoryMatch

```
<Directory directory>...</Directory>
```

```
<DirectoryMatch regex>...</DirectoryMatch>
```

### Files and FilesMatch

```
<Files filename>...</Files>
```

```
<FilesMatch regex>...</FilesMatch>
```

### Location and LocationMatch

```
<Location URL>...</Location>
```

```
<LocationMatch regex>...</LocationMatch>
```

### Proxy and ProxyMatch

```
<Proxy criteria>...</Proxy>
```

```
<ProxyMatch regex>...</ProxyMatch>
```

### VirtualHost

```
<VirtualHost addr[:port] >...</VirtualHost>
```

### Limit and LimitExcept

```
<Limit method method>...</Limit>
```

```
<LimitExcept method method>...</LimitExcept>
```

### Version

```
<IfVersion> [[!]operator version > ... </IfVersion>
```

**Note:** Not all directives enclosed by brackets (<>) are container directives. For example, directives <IfModule> and <IfDefine> are used to define parts of the HTTP Server configuration that are conditional and are ignored once the server has started; however, they are not directive containers.

## Context and server area relationship

The following table shows server area and context relationship.

Server area	Context
Global configuration	server config
Directory container	directory (&lt;Directory&gt; or &lt;DirectoryMatch&gt;)
File container	directory (&lt;File&gt; or &lt;FileMatch&gt;)
Location container	directory (&lt;Location&gt; or &lt;LocationMatch&gt;)
Proxy container	directory (&lt;Proxy&gt; or &lt;ProxyMatch&gt;)
Virtual host container	virtual host (&lt;VirtualHost&gt;)
Limit except container	<Limit> or <LimitExcept> <b>Note:</b> The context depends on the location of the <Limit> and <LimitExcept> container. It will inherit the context of the area it is in. For example, if the <Limit> and <LimitExcept> are within a directory container, the <Limit> or <LimitExcept> will be assigned the same values as the directory container.
Version containe	<IfVersion>

See "Directives for HTTP Server" on page 214 for more information on all the supported HTTP Server directives and the context in which the directives may be used.

## Directives within containers

The container directives <Directory>, <Location> and <Files> can contain directives which only apply to specified directories, URLs or files respectively. This also includes .htaccess files that can be used inside a directory container to apply directives to that directory.

Files that are included in the configuration file are processed by the HTTP Server at start time. Changes to files that are included in the configuration file (such as include files and group files, but not .htaccess files) do not take effect until the server is restarted.

Everything that is syntactically allowed in <Directory> is also allowed in <Location> (except a sub-<Files> section). Semantically however some things, and the most notable are AllowOverride and the two options FollowSymLinks and SymLinksIfOwnerMatch, make no sense in <Location>, <LocationMatch> or <DirectoryMatch>. The same for <Files> -- while syntactically correct, they are semantically incorrect.

## Directive inheritance

Directives inherit first from the top most (or "parent") directive container, then from more specific directive containers within.

In the following example, *Directory A* is the parent container to *Directory B*. *Directive b* first inherits its parameters from *Directory A* and *directive a* by default. If the parameters for *directive b* are defined, then *directive b* does not inherit, but uses its own parameter settings. Note that *directive a* does not inherit any parameter settings from *directive b*, since *directive a* is the parent to *directive b*. Inheritance only goes from parent to child.

```
<Directory A>
directive a
<Directory B>
  directive b
</Directory>
</Directory>
```

**Note:** Best practice for security of your HTTP Server is to put all security directives into each container to ensure that each directory or file is secured.

## How the directives are merged

The order of merging is:

1. <Directory> (except regular expressions) and .htaccess done simultaneously (with .htaccess overriding <Directory>)
2. <DirectoryMatch>, and <Directory> with regular expressions
3. <Files> and <FilesMatch> done simultaneously
4. <Location> and <LocationMatch> done simultaneously

Apart from <Directory>, each directive group (directives within container directives) is processed in the order that they appear in the configuration files. <Directory> (directive group 1 above) is processed in the order shortest directory component to longest. If multiple <Directory> sections apply to the same directory they are processed in the configuration file order. Configurations included through the Include directive will be treated as if they were inside the including file at the location of the Include directive.

Container directives inside a <VirtualHost> container directive are applied after the corresponding directives outside of the virtual host definition. This allows virtual hosts to override the main server configuration.

## Using container directives

### General guidelines:

- If you are attempting to match objects at the file system level then you must use the <Directory> and <Files> container directives.
- If you are attempting to match objects at the URL level then you must use the <Location> container directive.

### Notable exception:

- Proxy control is done via <Proxy> containers. Directives which are valid in a <Directory> container are also valid in a <Proxy> container. A <Proxy> container is very similar to a <Location> container, since it deals with virtual paths and locations rather than with physical paths. The directives in <Proxy> containers are processed after the directives in <Location> containers are processed, but before directives in <Directory> containers are processed. The directives in <Proxy> containers are also inherited into more specific <Proxy> containers in the same way as the directives in a <Directory> container.

### .htaccess parsing:

- Modifying .htaccess parsing within a <Location> container directive has no effect. The .htaccess parsing has already occurred.

### <Location> and symbolic links:

- It is not possible to use Options FollowSymLinks or Options SymLinksIfOwnerMatch inside a <Location>, <LocationMatch> or <DirectoryMatch> container directives (the Options are simply ignored). Using the Options in question is only possible inside a <Directory> container directive (or a .htaccess file).

### <Files> and Options:

- Using an Options directive inside a <Files> container directive has no effect.

**Note:** A <Location>/<LocationMatch> sequence is performed just before the name translation phase (where Aliases and DocumentRoots are used to map URLs to filenames). The results of this sequence are removed after the translation has completed.

### Related information:

“Directives for HTTP Server” on page 214

This topic provides information about the supported directives for IBM HTTP Server for i.

## Content negotiation for HTTP Server

The IBM HTTP Server for i supports content negotiation, type-map files, MultiViews, negotiation methods, dimensions of negotiation,, negotiation algorithm, media types, and wildcards.

A resource may be available in several different representations. For example, it might be available in different languages or different media types, or a combination. One way of selecting the most appropriate choice is to give the user an index page, and let them select; however it is often possible for the server to choose automatically. This works because browsers can send as part of each request information about what representations it prefers. For example, a browser could indicate that it would like to see information in French, if possible, else English will do. Browsers indicate their preferences by headers in the request. To request only French representations, the browser would send:

```
Accept-Language: fr
```

Note that this preference will only be applied when there is a choice of representations and they vary by language.

As an example of a more complex request, this browser has been configured to accept French and English, but prefers French, and to accept various media types, preferring HTML over plain text or other text types, and preferring GIF or JPEG over other media types, but also allowing any other media type as a last resort:

```
Accept-Language: fr; q=1.0, en; q=0.5
Accept: text/html; q=1.0, text/*; q=0.8, image/gif; q=0.6,
      image/jpeg; q=0.6, image/*; q=0.5, */*; q=0.1
```

The HTTP Server supports 'server driven' content negotiation, as defined in the HTTP/1.1 specification. It fully supports the Accept, Accept-Language, Accept-Charset and Accept-Encoding request headers. The HTTP Server also supports 'transparent' content negotiation, which is an experimental negotiation protocol defined in RFC 2295 and RFC 2296. It does not offer support for 'feature negotiation' as defined in these RFCs.

A **resource** is a conceptual entity identified by a URI (RFC 2396). The HTTP Server provides access to **representations** of the resource(s) within its namespace, with each representation in the form of a sequence of bytes with a defined media type, character set, encoding, or other. Each resource may be associated with zero, one, or more than one representation at any given time. If multiple representations are available, the resource is referred to as **negotiable** and each of its representations is termed a **variant**. The ways in which the variants for a negotiable resource vary are called the dimensions of negotiation.

## Content negotiation

In order to negotiate a resource, the server needs to be given information about each of the variants. This is done in one of two ways:

- Using a type -map (for example, a \*.var file) which names the files containing the variants explicitly.
- Using a 'MultiViews' search, where the server does an implicit filename pattern match and chooses from among the results.

## Using a type-map file

A type map is a document which is associated with the handler named type-map (or, for backwards-compatibility with older HTTP Server configurations, the mime type application/x-type-map). Note that to use this feature, you must have a handler set in the configuration that defines a file suffix as type-map; this is best done with an AddHandler in the server configuration file, as shown below.

```
AddHandler type-map var
```

Type map files have an entry for each available variant; these entries consist of contiguous HTTP-format header lines. Entries for different variants are separated by blank lines. Blank lines are illegal within an entry. It is conventional to begin a map file with an entry for the combined entity as a whole (although this is not required, and if present will be ignored). An example map file is:

```
URI: jk1
```

```
URI: jk1.en.html
Content-type: text/html
Content-language: en
```

```
URI: jk1.fr.de.html
Content-type: text/html;charset=iso-8859-2
Content-language: fr, de
```

If the variants have different source qualities, that may be indicated by the "qs" parameter to the media type, as in this picture (available as jpeg, gif, or ASCII-art):

```
URI: jk1
```

```
URI: jk1.jpeg
Content-type: image/jpeg; Qs=0.8
```

```
URI: jk1.gif
Content-type: image/gif; Qs=0.5
```

```
URI: jk1.txt
Content-type: text/plain; Qs=0.01
```

The "Qs" value can vary in the range 0.000 to 1.000. Note that any variant with a "Qs" value of 0.000 will never be chosen. Variants with no "Qs" parameter value are given a "Qs" factor of 1.0. The "Qs" parameter indicates the relative 'quality' of this variant compared to the other available variants, independent of the client's capabilities. For example, a jpeg file is usually of higher source quality than an ASCII file if its attempting to represent a photograph; however, if the resource being represented is an original ASCII art, then an ASCII representation would have a higher source quality than a jpeg representation. A "Qs" value is therefore specific to a given variant depending on the nature of the resource it represents.

The full list of headers recognized are:

**URI** The uri of the file containing the variant (of the given media type, encoded with the given content encoding). These are interpreted as URLs relative to the map file; they must be on the same server, and they must refer to files to which the client would be granted access if they were to be requested directly.

### Content-Type

The media type --- charset, level and "Qs" parameters may be given. These are often referred to as MIME types; typical media types are image/gif, text/plain, or text/html; level=3.

### Content-Language

The languages of the variant, specified as an Internet standard language tag from RFC 1766 (for example, en for English, or kr for Korean).

### Content-Encoding

If the file is compressed, or otherwise encoded, rather than containing the actual raw data, this



states how it was done. The HTTP Server only recognizes encodings that are defined by an `AddEncoding` directive. This normally includes the encodings `x-compress` for compressed files, and `x-gzip` for gzip'd files. The `x-` prefix is ignored for encoding comparisons.

### Content-Length

The size of the file. Specifying content lengths in the type-map allows the server to compare file sizes without checking the actual files.

### Description

A human-readable textual description of the variant. If the HTTP Server cannot find any appropriate variant to return, it will return an error response which lists all available variants instead. Such a variant list will include the human-readable variant descriptions.

## MultiViews

MultiViews is a per-directory option, meaning it can be set with an `Options` directive within a `<Directory>`, `<Location>` or `<Files>` container in the configuration file, or (if `AllowOverride` is properly set) in `.htaccess` files. Note that `Options All` does not set `MultiViews`; you have to ask for it by name.

The effect of `MultiViews` is as follows: if the server receives a request for `/some/dir/jkl`, if `/some/dir` has `MultiViews` enabled, and `/some/dir/jkl` does not exist, then the server reads the directory looking for files named `jkl.*`, and effectively fakes up a type map which names all those files, assigning them the same media types and content-encodings it would have if the client had asked for one of them by name. It then chooses the best match to the client's requirements.

`MultiViews` may also apply to searches for the file named by the `DirectoryIndex` directive, if the server is trying to index a directory. If the configuration files specify:

```
DirectoryIndex index
```

The server will arbitrate between `index.html` and `index.html3` if both are present.

If one of the files found when reading the directive is a CGI script, it is not obvious what should happen. The code gives that case special treatment --- if the request was a `POST`, or a `GET` with `QUERY_ARGS` or `PATH_INFO`, the script is given an extremely high quality rating, and generally invoked; otherwise it is given an extremely low quality rating, which generally causes one of the other views (if any) to be retrieved.

## The negotiation methods

After the HTTP Server has obtained a list of the variants for a given resource, either from a type-map file or from the filenames in the directory, it invokes one of two methods to decide on the 'best' variant to return, if any. It is not necessary to know any of the details of how negotiation actually takes place in order to use the HTTP Server content negotiation features. However the rest of this document explains the methods used for those interested.

There are two negotiation methods:

1. **Server driven negotiation with the HTTP Server algorithm** is used in the normal case. The HTTP Server algorithm is explained in more detail below. When this algorithm is used, the HTTP Server can sometimes 'fiddle' the quality factor of a particular dimension to achieve a better result. The ways the HTTP Server can fiddle quality factors is explained in more detail below.
2. **Transparent content negotiation** is used when the browser specifically requests this through the mechanism defined in RFC 2295. This negotiation method gives the browser full control over deciding on the 'best' variant, the result is therefore dependent on the specific algorithms used by the browser. As part of the transparent negotiation process, the browser can ask the HTTP Server to run the 'remote variant selection algorithm' defined in RFC 2296.

## Dimensions of negotiation

### Media Type

Browser indicates preferences with the Accept header field. Each item can have an associated quality factor. Variant description can also have a quality factor (the "Qs" parameter).

### Language

Browser indicates preferences with the Accept-Language header field. Each item can have a quality factor. Variants can be associated with none, one or more than one language.

### Encoding

Browser indicates preference with the Accept-Encoding header field. Each item can have a quality factor.

### Charset

Browser indicates preference with the Accept-Charset header field. Each item can have a quality factor. Variants can indicate a charset as a parameter of the media type.

### Client (Browser)

The User-Agent HTTP header is used to determine browser type.

## The negotiation algorithm

The HTTP Server can use the following algorithm to select the 'best' variant (if any) to return to the browser. This algorithm is not further configurable. It operates as follows:

1. First, for each dimension of the negotiation, check the appropriate Accept\* header field and assign a quality to each variant. If the Accept\* header for any dimension implies that this variant is not acceptable, eliminate it. If no variants remain, go to step 4.
2. Select the 'best' variant by a process of elimination. Each of the following tests is applied in order. Any variants not selected at each test are eliminated. After each test, if only one variant remains, select it as the best match and proceed to step 3. If more than one variant remains, move on to the next test.
  - a. Multiply the quality factor from the Accept header with the quality-of-source factor for this variant's media type, and select the variants with the highest value.
  - b. Select the variants with the highest language quality factor.
  - c. Select the variants with the best language match, using either the order of languages in the Accept-Language header (if present), or else the order of languages in the LanguagePriority directive (if present).
  - d. Select the variants with the highest 'level' media parameter (used to give the version of text/html media types).
  - e. Select variants with the best charset media parameters, as given on the Accept-Charset header line. Charset ISO-8859-1 is acceptable unless explicitly excluded. Variants with a text/\* media type but not explicitly associated with a particular charset are assumed to be in ISO-8859-1.
  - f. Select those variants which have associated charset media parameters that are not ISO-8859-1. If there are no such variants, select all variants instead.
  - g. Select the variants with the best encoding. If there are variants with an encoding that is acceptable to the user-agent, select only these variants. Otherwise if there is a mix of encoded and non-encoded variants, select only the non-encoded variants. If either all variants are encoded or all variants are not encoded, select all variants.
  - h. Select the variants that correspond to the User-Agent header received on the HTTP Request.
  - i. Select the variants with the smallest content length.
  - j. Select the first variant of those remaining. This will be either the first listed in the type-map file, or when variants are read from the directory, the one whose file name comes first when sorted using ASCII code order.

3. The algorithm has now selected one 'best' variant, so return it as the response. The HTTP response header Vary is set to indicate the dimensions of negotiation (browsers and caches can use this information when caching the resource).
4. To get here means no variant was selected (because none are acceptable to the browser). Return a 406 status (meaning "No acceptable representation") with a response body consisting of an HTML document listing the available variants. Also set the HTTP Vary header to indicate the dimensions of variance.

## Editing quality values

The HTTP Server sometimes changes the quality values from what would be expected by a strict interpretation of the HTTP Server negotiation algorithm above. This is to get a better result from the algorithm for browsers which do not send full or accurate information. Some of the most popular browsers send Accept header information which would otherwise result in the selection of the wrong variant in many cases. If a browser sends full and correct information these fiddles will not be applied.

## Media types and wildcards

The Accept: request header indicates preferences for media types. It can also include 'wildcard' media types, such as "image/\*" or "\*/\*" where the \* matches any string. So a request including Accept: image/\*, \*/\* would indicate that any type starting "image/" is acceptable, as is any other type (so the first "image/\*" is redundant). Some browsers routinely send wildcards in addition to explicit types they can handle. For example, Accept: text/html, text/plain, image/gif, image/jpeg, \*/\*.

The intention of this is to indicate that the explicitly listed types are preferred, but if a different representation is available, that is OK too. However under the basic algorithm, as given above, the \*/\* wildcard has exactly equal preference to all the other types, so they are not being preferred. The browser should really have sent a request with a lower quality (preference) value for \*.\* , such as: Accept: text/html, text/plain, image/gif, image/jpeg, \*/\*; q=0.01.

The explicit types have no quality factor, so they default to a preference of 1.0 (the highest). The wildcard \*/\* is given a low preference of 0.01, so other types will only be returned if no variant matches an explicitly listed type.

If the Accept: header contains *no* "q" factors at all, the HTTP Server sets the "q" value of "\*/\*", if present, to 0.01 to emulate the desired behavior. It also sets the "q" value of wildcards of the format "type/\*" to 0.02 (so these are preferred over matches against "\*/\*"). If any media type on the Accept: header contains a "q" factor, these special values are *not* applied, so requests from browsers which send the correct information to start with work as expected.

## Variants with no language

If some of the variants for a particular resource have a language attribute, and some do not, those variants with no language are given a very low language quality factor of 0.001.

The reason for setting this language quality factor for variant with no language to a very low value is to allow for a default variant which can be supplied if none of the other variants match the browser's language preferences. For example, consider the situation with three variants:

- jkl.en.html, language en
- jkl.fr.html, language fr
- jkl.html, no language

The meaning of a variant with no language is that it is always acceptable to the browser. If the request Accept-Language header includes either en or fr (or both) one of jkl.en.html or jkl.fr.html will be returned. If the browser does not list either en or fr as acceptable, jkl.html will be returned instead.

## Extensions to transparent content negotiation

The HTTP Server extends the transparent content negotiation protocol (RFC 2295) as follows. A new {encoding ..} element is used in variant lists to label variants which are available with a specific content-encoding only. The implementation of the RVSA/1.0 algorithm (RFC 2296) is extended to recognize encoded variants in the list, and to use them as candidate variants whenever their encodings are acceptable according to the Accept-Encoding request header. The RVSA/1.0 implementation does not round computed quality factors to 5 decimal places before choosing the best variant.

## Hyperlinks and naming conventions

If you are using language negotiation you can choose between different naming conventions, because files can have more than one extension, and the order of the extensions is normally irrelevant (see `mod_mime` for details).

A typical file has a MIME-type extension (for example, `html`), maybe an encoding extension (for example, `gz`), and of course a language extension (for example, `en`) when we have different language variants of this file.

Examples:

- `jkl.en.html`
- `jkl.html.en`
- `jkl.en.html.gz`

Examples of filenames together with valid and invalid hyperlinks:

Filename	Valid hyperlink	Invalid hyperlink
<code>jkl.html.en</code>	<code>jkl</code> <code>jkl.html</code>	-
<code>jkl.en.html</code>	<code>jkl</code>	<code>jkl.html</code>
<code>jkl.html.en.gz</code>	<code>jkl</code> <code>jkl.html</code>	<code>jkl.gz</code> <code>jkl.html.gz</code>
<code>jkl.en.html.gz</code>	<code>jkl</code>	<code>jkl.html</code> <code>jkl.html.gz</code> <code>jkl.gz</code>
<code>jkl.gz.html.en</code>	<code>jkl</code> <code>jkl.gz</code> <code>jkl.gz.html</code>	<code>jkl.html</code>
<code>jkl.html.gz.en</code>	<code>jkl</code> <code>jkl.html</code> <code>jkl.html.gz</code>	<code>jkl.gz</code>

Looking at the table above you will notice that it is always possible to use the name without any extensions in an hyperlink (for example, `jkl`). The advantage is that you can hide the actual type of a document resp. file and can change it later, for example, from `html` to `shtml` or `cgi` without changing any hyperlink references.

If you want to continue to use a MIME-type in your hyperlinks (for example `jdk1.html`) the language extension (including an encoding extension if there is one) must be on the right hand side of the MIME-type extension (for example, `jdk1.html.en`).

## Caching

When a cache stores a representation, it associates it with the request URL. The next time that URL is requested, the cache can use the stored representation. But, if the resource is negotiable at the server, this might result in only the first requested variant being cached and subsequent cache hits might return the wrong response. To prevent this, the HTTP Server normally marks all responses that are returned after content negotiation as non-cacheable by HTTP/1.0 clients. The HTTP Server also supports the HTTP/1.1 protocol features to allow caching of negotiated responses.

For requests which come from an HTTP/1.0 compliant client (either a browser or a cache), the directive `CacheNegotiatedDocs` can be used to allow caching of responses which were subject to negotiation. This directive can be given in the server config or virtual host, and takes no arguments. It has no effect on requests from HTTP/1.1 clients.

### Related information:

“Setting up content and language negotiation for HTTP Server” on page 96

Content negotiation for an HTTP Server instance can be set up using the IBM Web Administration for i interface. Content negotiation is defined as the process where the client provides a set of preferences (such as language) to the server, and the server finds the best resource match to those the client prefers.

## Virtual hosts on HTTP Server

This topic provides information about virtual host types on the IBM HTTP Server for i Web server.

The concept of virtual hosts allows more than one Web site on one system or Web server. The servers are differentiated by their host name. Visitors to the Web site are routed by host name or IP address to the correct virtual host. Virtual hosting allows companies sharing one server to each have their own domain names. For example `www.company1.com` and `www.company2.com` can both be hosted on the same server.

## HTTP Server virtual host types

There are three variations of virtual hosts on HTTP Server:

### IP address-based virtual host

The IP address-based virtual host requires one IP address per Web site (host name). This approach works very well, but requires a dedicated IP address for every virtual host. For more information on virtual hosts refer to the `<VirtualHost>` directive.

### Name-based virtual host

The name-based virtual host allows one IP address to host more than one Web site (host name). This approach allows practically an unlimited number of servers, ease of configuration and use, and requires no additional hardware or software. The main disadvantage to this approach is that the client must support HTTP 1.1 (or HTTP 1.0 with 1.1 extensions) that include the host name information inside the HTTP document requests. The latest versions of most browsers support HTTP 1.1 (or HTTP 1.0 with 1.1 extensions), but there are still old browsers that only support HTTP 1.0. For more information on virtual hosts refer to the `<VirtualHost>` directive.

### Dynamic virtual host

The dynamic virtual host allows you to dynamically add Web sites (host names) by adding directories of content. This approach is based on automatically inserting the IP address and the contents of the `Host:` header into the pathname of the file that is used to satisfy the request.

The advantages of a dynamic virtual host are:

- A smaller configuration file so that the server starts faster and uses less memory.

- Adding virtual hosts does not require the configuration to be changed or the server to be restarted.

The disadvantage of a dynamic virtual host is that you cannot have a different log file for each virtual host. For more information on dynamic virtual hosts refer to `mod_vhost_alias`.

#### **Related information:**

“Virtual host tasks” on page 132

This topic provides step-by-step tasks for configuring virtual hosts in the IBM HTTP Server for i Web server.

“JKL Toy Company creates virtual hosts on HTTP Server” on page 61

This scenario discusses how to create virtual hosts in an IBM HTTP Server for i Web server.

## **Proxy server types and uses for HTTP Server**

This topic provides information about proxy server types and uses for the IBM HTTP Server for i Web server.

Proxy servers receive requests intended for other servers and then act to fulfill, forward, redirect, or reject the requests. Exactly which service is carried out for a particular request is based on a number of factors which include: the proxy server's capabilities, what is requested, information contained in the request, where the request came from, the intended destination, and in some cases, who sent the request.

The two most attractive reasons to use a proxy server are its ability to enhance network security and lessen network traffic. A proxy server enhances network security by providing controls for receiving and forwarding (or rejecting) requests between isolated networks, for example, forwarding requests across a firewall. A proxy server lessens network traffic by rejecting unwanted requests, forwarding requests to balance and optimize server workload, and fulfilling requests by serving data from cache rather than unnecessarily contacting the true destination server.

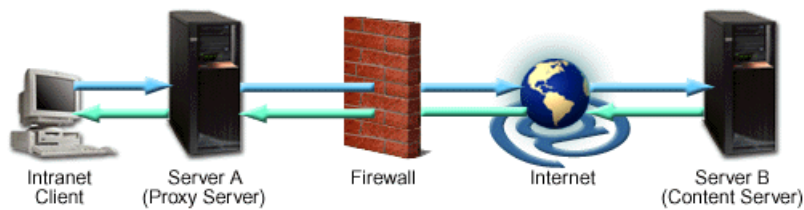
HTTP Server has proxy server capabilities built in. Activating these services is simply a matter of configuration. This topic explains three common proxy concepts: forward proxy, reverse proxy, and proxy chaining.

### **Forward proxy**

A forward proxy is the most common form of a proxy server and is generally used to pass requests from an isolated, private network to the Internet through a firewall. Using a forward proxy, requests from an isolated network, or intranet, can be rejected or allowed to pass through a firewall. Requests may also be fulfilled by serving from cache rather than passing through the Internet. This allows a level of network security and lessens network traffic.

A forward proxy server will first check to make sure a request is valid. If a request is not valid, or not allowed (blocked by the proxy), it will reject the request resulting in the client receiving an error or a redirect. If a request is valid, a forward proxy may check if the requested information is cached. If it is, the forward proxy serves the cached information. If it is not, the request is sent through a firewall to an actual content server which serves the information to the forward proxy. The proxy, in turn, relays this information to the client and may also cache it, for future requests.

The following image shows a forward proxy configuration. An intranet client initiates a request that is valid but is not cached on Server A (Proxy Server). The request is sent through the firewall to the Internet server, Server B (Content Server), which has the information the client is requesting. The information is sent back through the firewall where it is cached on Server A and served to the client. Future requests for the same information will be fulfilled by the cache, lessening network traffic (proxy caching is optional and not necessary for forward proxy to function on your HTTP Server).

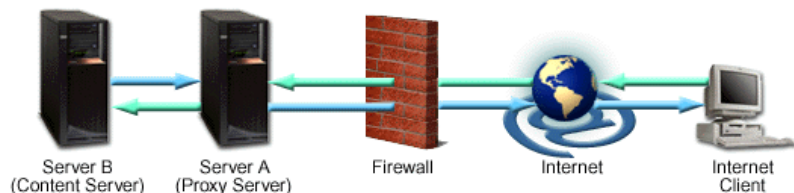


For information on how to configure a forward proxy, see “Setting up forward proxy for HTTP Server” on page 115.

## Reverse proxy

A reverse proxy is another common form of a proxy server and is generally used to pass requests from the Internet, through a firewall to isolated, private networks. It is used to prevent Internet clients from having direct, unmonitored access to sensitive data residing on content servers on an isolated network, or intranet. If caching is enabled, a reverse proxy can also lessen network traffic by serving cached information rather than passing all requests to actual content servers. Reverse proxy servers may also balance workload by spreading requests across a number of content servers. One advantage of using a reverse proxy is that Internet clients do not know their requests are being sent to and handled by a reverse proxy server. This allows a reverse proxy to redirect or reject requests without making Internet clients aware of the actual content server (or servers) on a protected network.

A reverse proxy server will first check to make sure a request is valid. If a request is not valid, or not allowed (blocked by the proxy), it will not continue to process the request resulting in the client receiving an error or a redirect. If a request is valid, a reverse proxy may check if the requested information is cached. If it is, the reverse proxy serves the cached information. If it is not, the reverse proxy will request the information from the content server and serve it to the requesting client. It also caches the information for future requests.



The above image shows a reverse proxy configuration. An Internet client initiates a request to Server A (Proxy Server) which, unknown to the client, is actually a reverse proxy server. The request is allowed to pass through the firewall and is valid but is not cached on Server A. The reverse proxy (Server A) requests the information from Server B (Content Server), which has the information the Internet client is requesting. The information is served to the reverse proxy, where it is cached, and relayed through the firewall to the client. Future requests for the same information will be fulfilled by the cache, lessening network traffic and load on the content server (proxy caching is optional and not necessary for proxy to function on your HTTP Server). In this example, all information originates from one content server (Server B).

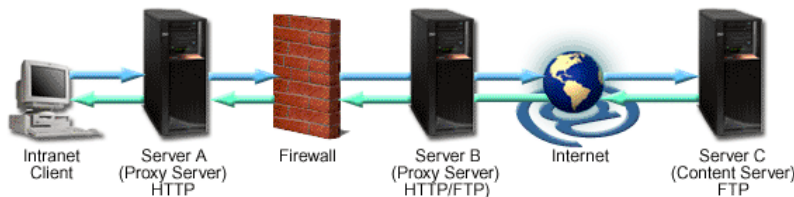
For information on how to configure a reverse proxy, see “Setting up reverse proxy for HTTP Server” on page 116.

## Proxy chaining

A proxy chain uses two or more proxy servers to assist in server and protocol performance and network security. Proxy chaining is not a type of proxy, but a use of reverse and forward proxy servers across

multiple networks. In addition to the benefits to security and performance, proxy chaining allows requests from different protocols to be fulfilled in cases where, without chaining, such requests would not be possible or permitted. For example, a request using HTTP is sent to a server that can only handle FTP requests. In order for the request to be processed, it must pass through a server that can handle both protocols. This can be accomplished by making use of proxy chaining which allows the request to be passed from a server that is not able to fulfill such a request (perhaps due to security or networking issues, or its own limited capabilities) to a server that can fulfill such a request.

The first proxy server in a chain will check to make sure a request is valid. If a request is not valid, or not allowed (blocked by the proxy), it will reject the request resulting in the client receiving an error or a redirect. If a request is valid, the proxy may check if the requested information is cached and simply serve it from there. If the requested information is not in cache, the proxy will pass the request on to the next proxy server in the chain. This server also has the ability to fulfill, forward, redirect, or reject the request. If it acts to forward the request then it too passes the request on to yet another proxy server. This process is repeated until the request reaches the last proxy server in the chain. The last server in the chain is required to handle the request by contacting the content server, using whatever protocol is required, to obtain the information. The information is then relayed back through the chain until it reaches the requesting client.



The above image shows a proxy chaining configuration. The intranet client makes a request to Server C (Content Server FTP). Server A (Proxy Server HTTP) does not contain the requested information in cache, so the request is passed through the firewall to Server B (proxy server HTTP/FTP). Server B has both HTTP and FTP protocols and is able to change the HTTP request to an FTP request. Server C receives the FTP request and passes back the requested information to Server B. Server B, in turn, passes the fulfilled request back to the intranet client using the HTTP protocol. The request is sent through the firewall and Server A where the request is cached and given to the intranet client.

For information on how to configure proxy chaining, see “Set up proxy chaining for HTTP Server” on page 117.

Reasons for passing requests through a proxy chain vary. For example, you may use proxy chaining to pass information through multiple networks where a client on one network cannot communicate directly with a proxy server on a different network, and it needs a second proxy to relay its requests. You may also use it to cache information in multiple locations or to allow certain protocols to be used outside a firewall which are not allowed through a firewall.

#### **Related information:**

“Proxy tasks” on page 115

The IBM HTTP Server for i supports proxy tasks.

## **Supported file systems for Web content served by HTTP Server**

This topic provides information about supported file systems for Web content by the HTTP Server.

The HTTP Server can serve content from any of the following file systems:

- Root (/)
- QSYS.LIB



- QOpenSys
- QDLS
- NFS
- QFileSvr.400
- QNTC
- QOPT
- UDFS

A file system provides the support that allows users and applications to access specific segments of storage that are organized as logical units. These logical units are files, directories, libraries, and objects.

Each file system has a set of logical structures and rules for interacting with information in storage. These structures and rules may be different from one file system to another. From the perspective of structures and rules, the support for accessing database files and various other object types through libraries can be thought of as a file system. Similarly, you can think of the support for accessing documents (which are really stream files) through the folders structure as a separate file system.

As you decide from which file system to serve files, you might want to consider the following:

- Serving from the root (or /) directory gives you the fastest response times.
- Will the tools you use to maintain your site be compatible with the file system you choose?
- How easy must it be to move content from platform to platform?

Remember that any individual server can serve content (CGI scripts; HTML files; graphics such as .jpegs, GIFs, and image maps; and so on) from many file systems at once. You can configure your server to serve content from whatever file systems suit your needs.

Before you start serving your content from the Integrated File System, you must ensure that the world can access the files that you want to serve. You must grant the QTMHHTTP user profile or \*PUBLIC the following authorities and permissions to enable Web serving with the HTTP Server:

- QTMHHTTP or \*PUBLIC must have \*USE authority to all library system objects that you intend to serve.
- If you use any of the log directives with any Integrated File System directory name, the directory must exist, and QTMHHTTP or \*PUBLIC must have \*RWX authority.
- The QTMHHTTP user profile or \*PUBLIC must be granted \*RX authority to all objects (HTML pages, graphics, and so on) that you intend to serve.
- To use CGI programs to access any of the objects you serve, the QTMHHTTP1 user profile or \*PUBLIC needs the same authority to the objects as QTMHHTTP.

**Note:** When considering from which file system to serve files, keep in mind that AllowOverride should be None for QDLS. Also, file serving and manipulation from QSYS and other EBCDIC file systems might result in performance bottlenecks.

#### **Related information:**

File systems

## **Logging**

The HTTP Server provides many logging features.

### **Log formats for HTTP Server**


This topic provides information about log formats and log files.

Log files contain one line for each request. A line is composed of several tokens separated by spaces. If a token does not have a value then it is represented by a hyphen (-). A line in a log file might look like the following:

```
192.168.1.3 - - [18/Feb/2000:13:33:37 -0600] "GET / HTTP/1.0" 200 5073
```

The following log file types are supported:

#### **Common (Access)**

This format is the common log file format defined by the W3C working group. This format is compatible with many industry standard log tools. For more information see Logging Control In W3C httpd .

The common log format is defined by the following string:

```
"%h %l %u %t \"%r\" %>s %b"
```

#### **Extended (Access, Referer, and Agent)**

This format has two types: NCSA extended log format and the W3C extended log format. The NCSA extended log format is the common log format appended with the agent and referer information. The W3C extended log format is defined by the W3C working group and allows you to determine the format of the log entry. For more information see Extended Log File Format



NCSA's extended format is defined by the following string:

```
"%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\""
```

#### **Data Description Specification (DDS)**

This format is an IBM i database (physical) file in QSYS.LIB. This format allows you to write a database query program to generate reports. This format contains the same information as the common log format.

#### **Related information:**

"Log file format tokens" on page 489

This topic provides information about tokens used to define log file formats.

"Setting up logs on HTTP Server" on page 112

Set up logs to record events and other information for your IBM HTTP Server for i instance using the IBM Web Administration for i interface.

### **Web Log Monitor**

| The Web Log Monitor provides users the ability to monitor the contents of log files for HTTP and application servers. Rules can be defined to describe what contents in a log file are to be monitored for. | When a defined rule is matched in the specified log file, a notification is sent to the configured notification channel.

| The Web Log Monitor inspects specified log files of any Web-related server, such as Integrated Web Application Server, Integrated Web Services Server, WebSphere Application Server, and IBM HTTP Server. | The log files are inspected for each keyword that is specified in the rule. If a match is encountered, a notification is sent to the configured notification channel, which can be one of the following channels:

- | • The \*QSYSOPR system message queue.
- | • One or more e-mail addresses.
- | • Both the \*QSYSOPR system message queue and e-mail addresses.

| You can monitor log files for multiple servers under a single monitor.

| Use the IBM Web Administration for i interface to configure the Web Log Monitor to monitor logs of your Web environment.

| **Related information:**

- | “Log formats for HTTP Server” on page 27
- | This topic provides information about log formats and log files.
- | “Setting up logs on HTTP Server” on page 112
- | Set up logs to record events and other information for your IBM HTTP Server for i instance using the
- | IBM Web Administration for i interface.

## Security

The HTTP Server provides many security features that help you control access to data and files.

### Security tips for HTTP Server

This topic provides tips to secure your IBM HTTP Server for i Web server.

Some hints and tips on security issues in setting up the HTTP Server.

- “Permissions on HTTP Server directories”
- “Stopping users from overriding system wide settings for HTTP Server ”
- “Protecting server files by default for HTTP Server”
- “Server Side Includes for HTTP Server ” on page 30

### Permissions on HTTP Server directories

In typical operation, the HTTP Server is started under the IBM i user profile QTMHHTTP and requests coming into the server are run under that user profile. It is possible to start the server and serve requests under different profiles. Refer to the ServerUserID and UserID directives for more information. You must also ensure that all of the resources that can be accessed by a Web client are properly protected. See “User profiles and required authorities for HTTP Server” on page 30 for additional information.

### Stopping users from overriding system wide settings for HTTP Server

You will want to stop users from setting up .htaccess files which can override security features. Here is one example:

```
<Directory />
  AllowOverride None
  Options None
</Directory>
```

This stops all overrides, Includes, and accesses in all directories. You also need to set up directory containers to allow access for specific directories.

### Protecting server files by default for HTTP Server

HTTP Server has a default access feature. To prevent clients from seeing the entire file system, add the following block to the configuration:

```
<Directory />
  Order deny,allow
  Deny from all
</Directory>
```

This forbids default access to file system locations. Add appropriate <Directory> blocks to allow access. For example,

```
<Directory /users/public_html>
  Order deny,allow
  Allow from all
</Directory>
```

Pay particular attention to the interactions of <Location> and <Directory> directives. For example, even if <Directory /> denies access, a <Location /> directive might override it.

## Server Side Includes for HTTP Server

Server side includes (SSI) can be configured so that users can execute programs on the server. To disable that part of SSI use the IncludesNOEXEC option to the Options directive.

## User profiles and required authorities for HTTP Server

This topic provides information about user profiles and required authorities for the IBM HTTP Server for i Web server.

The QTMHHTTP user profile is the default user profile of HTTP Server. This user profile is referred to as the server user profile. The server user profile must have read and execute authority to the directory path of the server root directory. If you are using the **Create New HTTP Server wizard**, the default server root path is `/www/server_name/`, where `server_name` is the name of the HTTP Server.

The server user profile must have read, write, and execute authority to the directory path where the log files are stored. If you are using the **Create New HTTP Server wizard**, the default path is `/www/server_name/logs/`, where `server_name` is the name of the HTTP Server. The log files could include any access, script, or rewrite logs. These logs may or may not be configured to be stored in the `/www/server_name/logs/` directory. Since log files could potentially contain sensitive information, the security of the configuration and log files should be fully considered. The path of the configuration and log files should only be accessible by the appropriate user profiles.

The QTMHHTTP1 user profile is the default user profile that HTTP Server uses when running CGI programs. This user profile must have read and execute authority to the location of any CGI program. User QTMHHTTP requires \*RWX (write) authority to directory `'/tmp'`.

You can optionally specify that the QTMHHTTP or QTMHHTTP1 user profile swap to another user profile as long as that user profile has the required authorities. For more information, see "UserID" on page 234.

- \*RX authority for root directory ("/ ") and directory `"/www"`, including all subdirectories in the path
- \*RWX authority for directory `"/www/server_name/"`

**Note:** Granting \*ALLOBJ authority to any server user profile is not recommended.

### Related tasks:

"Starting Web Administration for i" on page 7

The Web Administration for i allows you to create and manage different types of servers, including Web servers and application servers. Complete the following steps to start the Web Administration for i interface.

## Validation list on HTTP Server

This topic provides information about validation lists for limiting access to your IBM HTTP Server for i Web server.

Your system uses validation lists in conjunction with other resources to limit access to your server resources. Each validation list contains a list of Internet users and their passwords. Each Internet user has one valid password defined for it. An IBM i user profile is never created for the internet users.

A validation list is an IBM i object of type \*VLDL that stores user names and passwords or SSL certificates for use in access control. Validation lists are case-sensitive. Validation lists reside in IBM i libraries and are required when adding a user unless you are adding the user to a group file. If you enter a validation list that does not exist, the system will create it for you.

To create and delete validation lists, you can use the CL commands Create Validation List (CRTVLDL) and the Delete Validation List (DLTVLDL). Validation List APIs are also provided to allow applications to add, change, remove, verify (authenticate), and find entries in a validation list.

Validation list objects are available for all applications to use. For example, if an application requires a password, the application passwords can be stored in a validation list object rather than a database file. The application can use the validation list APIs to verify a user's password, which is encrypted, rather than the application performing the verification itself.

## Kerberos for HTTP Server

Kerberos for network authentication can be used for an IBM HTTP Server for i instance.

Kerberos is a network authentication protocol designed to provide authentication for client or server applications with secret-key cryptography. Kerberos is a ticket-based authentication system that provides an alternative to user and password or X.509 certificate authentication. With the HTTP Server, you can use Kerberos on its own or in conjunction with Enterprise Identity Mapping (EIM) to authenticate Web users to the Web server.

For more information on EIM, see EIM concepts.

### Kerberos requirements

- Supported for IBM i 5.3, or later.
- Check your browser information to ensure it supports Kerberos. Not all browsers support Kerberos and some only support it in their more recent versions.

See the “JKL Toy Company enables single signon for HTTP Server” on page 75 scenario for a complete step-by-step instructions on how to enable Kerberos for your IBM i server.

## Performance


Performance in a Web server environment is influenced by many components. Understanding the components can help you ensure that the HTTP Server is functioning at the highest performance levels.

### File compression for HTTP Server

Information is compressed by the HTTP Server before being sent to the client over the network.

Compressed output is transferred to requesting client browsers at a higher rate of speed than output that is not compressed. This decreases the amount of data that the server needs to send over the network and improves network performance and response times.

Compression and decompression is implemented by the DEFLATE filter, located in “Module mod\_deflate” on page 338. The DEFLATE filter is always inserted after RESOURCE filters like PHP or

SSI. It never touches internal sub-requests. See [Apache HTTP Server Version 2.2 Documentation](#)  for additional information and examples on configuring the Apache server to use compression.

When the DEFLATE filter is used, a LoadModule is required in order to recognize the associated directives.

```
LoadModule deflate_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

### Output compression

Files can be compressed by the server before output to the client. The server can be configured to only compress files which are located in specific containers or globally. Directive SetOutputFilter enables compression for files in the container where it is placed. For example:

```
SetOutputFilter DEFLATE
```

Files being compressed can also be restricted to specific MIME types. In order to configure the server to restrict compression based on MIME types, the `AddOutputFilterByType` directive should be used. For example, to enable compression only for the HTML files located in a specific directory:

```
<Directory "/your-server-root/htdocs">
    AddOutputFilterByType DEFLATE text/html
</Directory>
```

## Input compression

Compressed files require decompression before they can be used. A filter is necessary for decompressing a GZIP compressed request body. The DEFLATE filter is required in the input filter chain and is set by using the `SetInputFilter` or the `AddInputFilter`. For example:

```
<Location /dav-area>
    SetInputFilter DEFLATE
</Location>
```

Requests containing a `Content-Encoding: GZIP` header are automatically decompressed. The `Content-Length` header specifies the length of incoming data from the client, not the byte count of the decompressed data stream. The actual length of the data will be greater than the `Content-Length` header indicates after the decompression has been done.

**Note:** Check your browser to ensure it supports GZIP request bodies.

## Proxy servers

Proxy servers receive a `Vary: Accept-Encoding` HTTP response header to identify that a cached response should be sent only to clients that send the appropriate `Accept-Encoding` request header. The response header prevents compressed content from being sent to a client that cannot support it.

Dependencies on special exclusions, for example, the `User-Agent` header, can be specified with an addition to the `Vary` header. The `Vary` header must be manually configured in order to alert proxies of the additional restrictions. For example, where the addition of the DEFLATE filter depends on the `User-Agent`, the following `Vary` header should be added:

```
Header append Vary User-Agent
```

If compression depends on information other than request headers, set the `Vary` header with a value of `*`. The `*` prevents compliant proxies from caching entirely. For example:

```
Header set Vary *
```

### Related information:

“Compression tasks” on page 108

The IBM HTTP Server for i supports the configuration and management of compression files.

## Fast Response Cache Accelerator (FRCA) for HTTP Server

The Fast Response Cache Accelerator (FRCA) improves the performance and scale of Web and TCP server applications by storing both static and dynamic content in a memory-based cache located in the Licensed Internal Code.

FRCA improves efficiency, scale, and performance by implementing two concepts:

- **Use of a memory-based cache that filters what is stored and what is dismissed.** Requests can process much more quickly when the majority of the requested content is cached within the Licensed Internal Code. The memory-based cache delegates stored information to the Network File Cache which is located in the Licensed Internal Code.
- **Caching functions from within the Licensed Internal Code to reduce request processing time.** Storing cached files within the Licensed Internal Code eliminates overhead and reduces request

processing time by reducing task-switching between the Licensed Internal Code and user application layers. This conserves system resources allowing them to be reallocated towards hosting dynamic content.

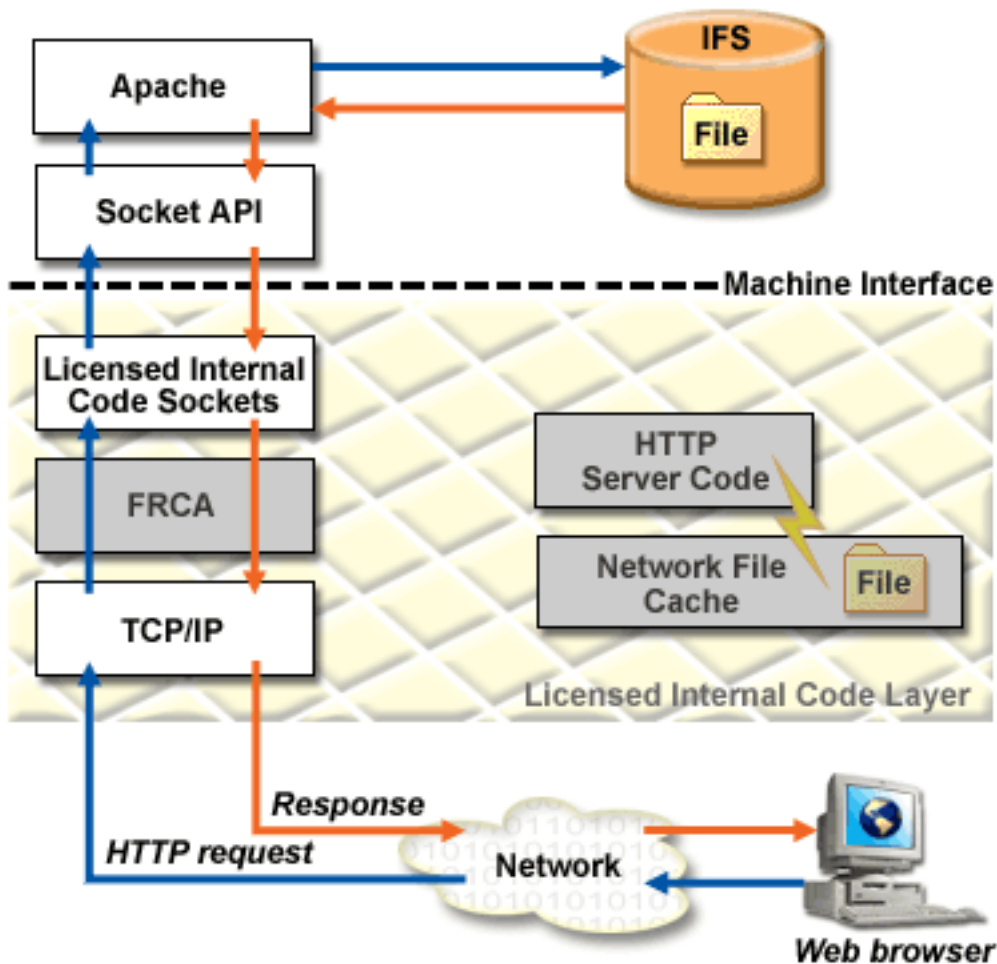
**Note:** The HTTP Server does not check for authorization on content served from FRCA. Use FRCA to cache content that does not need to be secured or accessed through specific validation.

**FRCA improves HTTP Server performance for both static and dynamic content.**

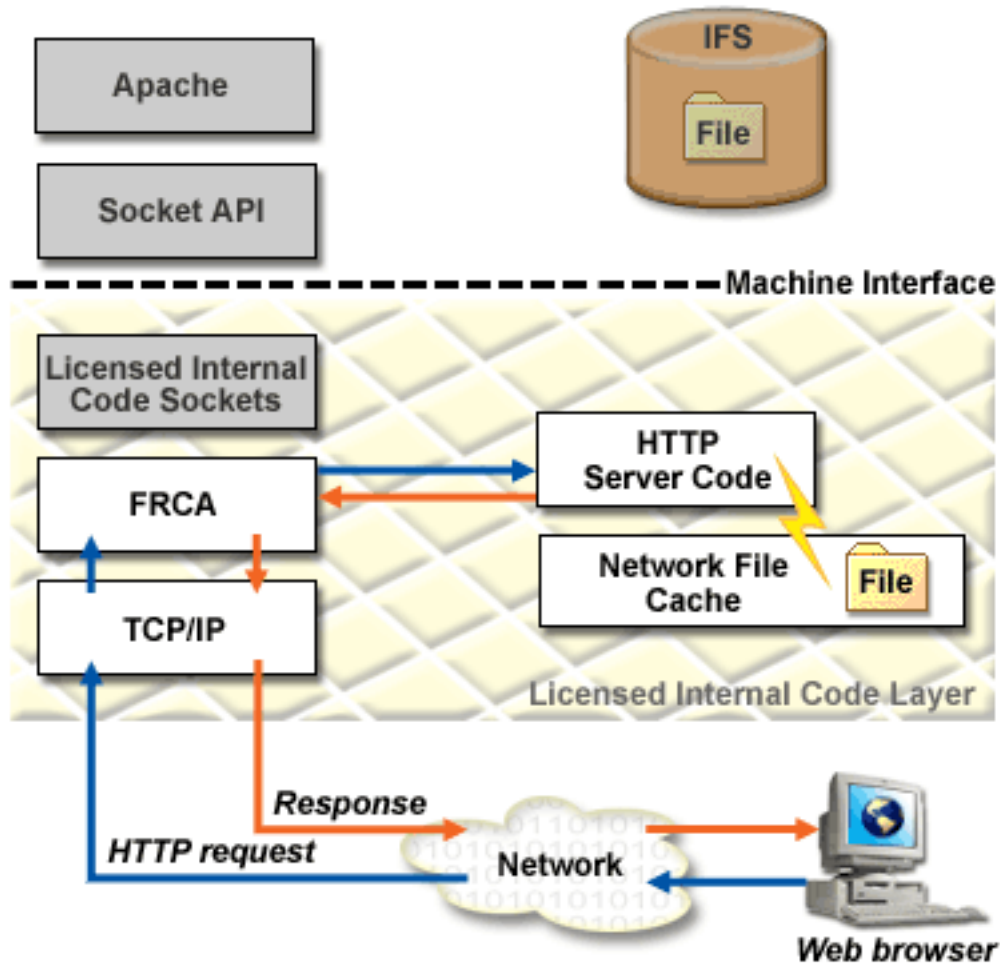
Static content, or content that comes from a file, is stored in Network File Cache and is then served to Licensed Internal Code of HTTP Server which essentially 'short circuits' the normal request processing path so the requested information will reach the user faster.

Dynamic content can be served from a Licensed Internal Code proxy cache or distributed by the Licensed Internal Code reverse proxy to one or more remote servers. A layer-7 router looks at the URL paths to route dynamic requests to the appropriate remote server.

**Static content request and response process without FRCA**



**Static content request and response process with FRCA**



**Related information:**

“Fast Response Cache Accelerator tasks” on page 109

The IBM HTTP Server for i supports the Fast Response Cache Accelerator (FRCA).

**Real time server statistics**

Real time server statistics provide information on IBM HTTP Server for i performance.

Server statistics can be viewed with the **Real Time Server Statistics** tool available through the IBM Web Administration for i interface. Only statistics for running HTTP Servers can be viewed. Data is collected from the primary server job only.

The header information for the active server displays the following:

**Server name**

Displays the name of the active server. The user-defined name was specified during the creation of the server.

**Job** Displays the job name for the active server.

**Server started**

Displays the date and time the server was started.

**Current time**

Displays the date and time of the last manual or automatic refresh of the statistical information.



Statistical information can be refreshed manually by clicking **Refresh** or can be automatically refreshed by selecting a refresh rate from the **Refresh Interval** drop-down menu.

**Note:** Statistical information is cumulative. If a value is greater than  $2^{64}-1$  in any column, the value will reset to 0. All values will reset to 0 if the server is stopped and then started. The type of information displayed is dependent on the activity of the HTTP Server and what functions are enabled. Only statistical information for enabled or active functions are displayed. Each column heading identifies what enabled function or associated server is being surveyed for statistical information.

Each column heading identifies what enabled function or associated server is being surveyed for statistical information. Statistical information is obtained for the following functions:

#### **Server handled**

This column displays the number of completed server transactions by the HTTP Server since the server was started. For example, completed transactions for static HTML pages, HTML pages containing Server Side Include (SSI), and images.

**Proxy** This column displays the number of completed server transactions that used proxy since the server was started. Proxy statistics are only available if proxy is enabled. See “Proxy server types and uses for HTTP Server” on page 24 for more information.

**CGI** This column displays the number of completed server transactions that were handled as Common Gateway Interface (CGI) since the server was started. CGI statistics are only available if CGI is enabled. See “Setting up CGI jobs” on page 135 for more information.

#### **Using SSL**

This column displays the number of completed server transactions that used Secure Sockets Layer (SSL) since the server was started. SSL statistics are only available if SSL is enabled. See “JKL Toy company enables Secure Sockets Layer (SSL) protection on HTTP Server” on page 70 for more information.

#### **WebSphere**

This column displays the number of completed server transactions that used an associated application server since HTTP Server was started. If the associated application server is not running, the information will still be displayed but will equal '0'. WebSphere statistics are only available if a WebSphere Application Server is associated with an HTTP Server.

#### **Customer module**

This column displays the number of completed server transactions that used a customer or third-party module. A customer module is a user module incorporated as a service program into the HTTP Server. See “Apache module programming” on page 198 for more information.

#### **FRCA Stats**

This column displays the number of completed server transactions that used Fast Response Cache Accelerator (FRCA) since the server was started. FRCA statistics are only available if FRCA is enabled. See “Fast Response Cache Accelerator (FRCA) for HTTP Server” on page 32 for more information.

#### **FRCA Proxy**

This column displays the number of completed server transactions that used Fast Response Cache Accelerator (FRCA) proxy since the server was started. FRCA statistics are only available if FRCA is enabled. See “Fast Response Cache Accelerator (FRCA) for HTTP Server” on page 32 for more information.

#### **General**

The general statistical information displays basic information about the active server since the server was started. Statistical information displayed includes the following:

**Active threads**

Displays the number of currently active threads on the server. A thread is an independent unit of work within a job that uses many of the jobs resources to complete work. The difference between jobs and threads is that threads run within the job helping it to finish its work. Every active job has at least one thread, which is called an *initial thread*. The initial thread is created as part of starting the job. The use of threads within a job allows many things to be done at once. For example, while a job is processing, a thread may retrieve and calculate data needed by the job to finish processing.

**Idle threads**

Displays the number of currently idle threads active on the server. An idle thread is a portion of a program that is waiting for either a response or a request before it can continue. Idle threads are most often waiting for an HTTP request to process.

**Normal connections**

Displays the number of total normal (non-secure) connections currently active.

**SSL connections**

Displays the number of total SSL (secure) connections currently active.

**Requests**

Displays the number of total requests to the server since the server was started.

**Responses**

Displays the number of total responses from the server since the server was started.

**Requests rejected**

Displays the number of total rejected requests issued by the server since the server was started.

**Absolute and Delta**

The absolute and delta information displays statistical information about currently enabled functions or associated servers. The *absolute value* is a measurement of the total transactions since the server was started. The *delta value* is a measurement of the total transactions since the server statistics were refreshed. The absolute and delta statistical information may be displayed separately or side by side for comparison. Connections are not the same thing as a request or response transaction. Connection are only recorded for new inbound connections to the server. Each column heading identifies what enabled function or associated server is being surveyed for statistical information. Each row identifies what statistical information is being retrieved. Statistical information displayed for each column includes:

**Requests**

Displays the number of requests to the enabled function or associated server identified at the top of the column.

**Responses**

Displays the number of responses sent by the enabled function or associated server identified at the top of the column.

**Error responses**

Displays the number of error responses sent by the enabled function or associated server identified at the top of the column. An error response example is the 404 "Page Not Found" response.

**Non-cache responses**

Displays the number of non-cached responses sent by the enabled function or associated server identified at the top of the column.

**Cache responses**

Displays the number of local memory cached responses sent by the enabled function or associated server identified at the top of the column.

**Bytes received**

Displays the number of bytes received by the enabled function or associated server identified at the top of the column.

**Bytes sent**

Displays the number of bytes sent by the enabled function or associated server identified at the top of the column.

**Non-cache Processing (seconds)**

Displays the number of seconds of non-cached processing activity completed by the enabled function or associated server identified at the top of the column.

**Cache Processing (seconds)**

Displays the number of seconds of cached processing activity completed by the enabled function or associated server identified at the top of the column.

**Averages**

The server averages information displays the average length of activity, in seconds, completed by the enabled function or associated server identified at the top of the column. Each column heading identifies what enabled function or associated server is being surveyed for statistical information. Each row identifies what statistical information is being retrieved. Averages are not affected by end user response times. Factors such as internet and intranet traffic, firewalls, and connection speeds are not determined. Statistical information displayed for each column includes:

**Total (seconds)**

Displays the total time of activity completed by the enabled function or associated server identified at the top of the column.

**Non-cached (seconds)**

Displays the average length of time of non-cached activity completed by the enabled function or associated server identified at the top of the column.

**Cached (seconds)**

Displays the average length of time of cached activity completed by the enabled function or associated server identified at the top of the column.

**Web Performance Monitor**

The IBM Web Administration for i interface provides the Web Performance Monitor to monitor and evaluate the performance characteristics of your Web environment.

A Web environment is a grouping of related HTTP and application servers that form a Web solution. In a Web environment, you may have to monitor the performance of application servers, HTTP servers, databases, and applications jobs. The Web Performance Monitor provides a single interface to monitor your entire Web environment, to find how your system time and resources are being spent, and to identify potential bottlenecks. Only one Web environment on the system can be monitored at one time.

When performance monitoring is enabled using the Web Performance monitor, your HTTP server instances are stopped and the HTTP server configurations are updated to enable ARM data collection, your application servers are stopped and the application server configurations are updated to enable ARM data collection, ARM data collection is started, and your HTTP server and application servers are restarted to activate the configuration changes. You should only enable Web performance monitoring when it is acceptable to not access the Web environment for a period of time, as your HTTP and application servers are stopped while the configurations are updated. Enabling performance monitoring may take several minutes to complete.

When the Web Performance Monitor is enabled, the **Transactions** tab displays the transaction statistics for jobs that are collecting ARM data in your environment. For each transaction, the average response time

and the number of completed transactions information is shown. Very large results for average response time may indicate a bottleneck and provide a starting point to investigate a performance problem.

You can also choose to look at the transaction times for a single IP Address or hostname, allowing you several creative ways to figure out what is going on within an application. The transaction information shows what jobs the processing requests for each user or IP address has been processed by. This very specific user information can help identify which jobs are slowing things down.

When the Web Performance Monitor is enabled, the **Jobs** tab displays the active servers and jobs in your environment. The Web Performance Monitor tool displays only the jobs on the system that are related to the selected Web server. For example, if the HTTP server is on a remote system, then there will not be any jobs for the HTTP server displayed. For each job, the current user, CPU usage percentages, run priority, number of threads, average transaction response time, and number of complete transactions is provided. Very large results for average response time or CPU usage percentage may indicate a bottleneck and provide a starting point to investigate a performance problem.

When performance monitoring is disabled using the Web Performance Monitor, your HTTP server instances are stopped and HTTP server configurations are updated to disable ARM data collection, your application servers are stopped and the application server configurations are updated to disable ARM data collection, ARM data collection is stopped, and your HTTP server and application servers are restarted to activate the configuration changes. You should only disable Web performance monitoring when it is acceptable to not access the Web environment for a period of time, as your HTTP and application servers are stopped while the configurations are updated. Disabling performance monitoring may take several minutes to complete.

**Related information:**

“Web Performance Monitor” on page 127

The IBM Web Administration for i interface provides the Web Performance Monitor to monitor and evaluate the performance characteristics of your Web environment.

## **Web Performance Advisor**

The Web Performance Advisor provides a way to view, evaluate and modify the attributes that affect the performance of your Web environment. Clear definitions of the attributes are provided along with recommended values. The tool also provides rating for each attribute to help guide the user to acceptable settings.

A Web environment is a grouping of related Web and application servers that form a Web solution. A Web environment is typically made up of a single WebSphere Application Server instance or profile and all the application servers contained within, its corresponding IBM HTTP Server, and any system attributes that could have a direct effect on the performance of the Web environment.

The Web Performance Advisor is made up of multiple components to help you tune the performance of your system and Web environment. These components include an advisor and an export function. These can be launched from the Web Performance Advisor introduction page. On this introduction page, the user is provided a quick, easy-to-read, high-level view of their system and Web environment performance.

The Advisor function allows you to manage system attributes and to manage Web environment attributes. From the manage system and manage Web environment panels, you can view, evaluate, and change each performance attribute. While evaluating each performance attribute, click the attribute's Advise link to learn about the attribute and find the recommended setting. The Web Performance Advisor gathers ratings and recommendations for each of the performance attributes being tuned. From these ratings, icons are displayed to convey whether the attribute is tuned well (green), may need some additional tuning (yellow), or needs immediate attention (red). The ratings that are displayed may vary based on the risk level (conservative or aggressive) you have configured in the General Settings. Conservative means that you do not want to be alerted to those performance attributes that are on the fringe. By using the conservative approach, fewer attributes are changed and drastic performance updates

are not made. Of course, performance may not be tuned as well, but there is much less risk of degrading your machine as a whole. Using the aggressive approach, any attribute that is on the fringe is flagged as needing to be changed. In addition, attributes that would be flagged as good in a conservative mode, might actually be flagged as needing improvement. By doing this, more drastic performance updates are made which may dramatically improve performance. On the downside, the possibility exists that unexpected, unwanted consequences may result from these drastic performance changes.

The export function allows you to save existing performance settings in a performance profile. This profile can be evaluated, compared, or sent to a performance expert for analysis and modification.

When the Web Performance Advisor tool is used to examine a Web related server, a flight recorder performance profile is created to save what all performance attributes are set to prior to any changes being made. Whenever changes are made through the Web Performance Advisor, all the performance attributes are saved (including the new changes) to another flight recorder performance profile file. This is necessary so that you can keep track of all changes made to a Web environment. All flight recorder performance profile files are located in the '/QIBM/UserData/HTTPAdmin/WPA' directory. The Web Performance Advisor tool does not clean up these files; they remain until someone deletes them manually.

Because the attributes affecting performance in a Web environment are located in many places, the Web Performance Advisor combines all of the performance attributes into a performance profile. The profile contains:

- System attribute information made up of the physical and logical resources that have been allocated to the system and partition and selected system values that can have a direct effect on Web performance, TCP/IP settings, and PTF information including the PTF Groups and the individual product PTFs for the products that are used in a Web environment.
- Web attribute information for the WebSphere Application Server instance or profile configured for this Web environment, including all the application servers configured for this particular instance or profile.
- Web performance attributes for each application server being tuned including the WebSphere Application Server JVM settings, system and server resource settings, server JDBC providers and data source resources, and other additional server settings.
- Web attribute information related to your external HTTP server associated with WebSphere Application Server instance or profile.

**Related information:**

“Web Performance Advisor” on page 128

The Web Performance Advisor provides a way to view, evaluate and modify the attributes that affect the performance of your Web environment. Clear definitions of the attributes are provided along with recommended values. The tool also provides rating for each attribute to help guide the user to acceptable settings.

## **Extending HTTP Server functionality**

Maintaining static Hypertext Markup Language (HTML) pages can be easy and inexpensive, but static pages cannot cover all of your Web serving needs. Any time the published content needs to be tailored on data received from a client, the Web page has to be generated on the fly. Serving dynamic data from your IBM HTTP Server for i Web server can be accomplished in several different ways, depending on your needs, your programming skills, and the complexity of the task at hand.

The core functionality of the HTTP Server can be extended to serve dynamic data by Common Gateway Interface (CGI) programs, Apache modules, and server-side includes (SSI). In addition, products available that can be used in the generation of dynamic Web data include applications servers such as WebSphere Application Server, Lotus® Domino®, and Integrated Web Application Server; and server-side scripting languages such as Net.Data and PHP.

## CGI

The Common Gateway Interface (CGI) specification was introduced to enable and standardize the interface between Web servers and external programs. The CGI is a relatively simple, platform and language independent, industry-standard interface for Web application development. Programs that implement the CGI standard are commonly called *CGI programs*.

The purpose of CGI is to extend the capability of an HTTP server by providing framework in which an HTTP server can interface with a program that is specified on a URL. The format of the URL allows parameters to be passed to the CGI program. On the server side, the interface describes how the program is started by the HTTP server and how parameters for the program are passed using a combination of standard-input and environment variables. It also describes how output information (such as HTML elements) are passed back to the HTTP server using standard output. Thus, in its simplest form, a CGI program can be defined as a program that:

1. Can be called as an executable program and run as a child process of the HTTP server.
2. Is able to read from the standard input.
3. Is able to access environment variables.
4. Is able to write to the standard output.
5. Is able to access command- line arguments passed to the program.

The administrator controls which CGI programs the system can run by using the server directives. The server recognizes a URL that contains a request for a CGI program, commonly called a *CGI script*. (Throughout the documentation, we use the terms CGI program and CGI script to mean the same thing.) Depending on the server directives, the server calls that program on behalf of the client.

The server supports CGI programs that are written in C++, REXX, ILE C, ILE RPG, and ILE COBOL. It also supports multiple thread capable CGI programs in all languages that support multiple threads.

CGI programs that are created by compiling source code typically run faster than programs that are written in interpreted languages such as the Net.Data<sup>®</sup> and PHP scripting languages. However, programs that are written in scripting languages tend to be easier to write, maintain, and debug.

The support for CGI by IBM HTTP Server for i includes support for IBM i-unique features that improve the CGI programming model in the areas of performance, high-availability, and support for transactions. The following sections discuss the various features.

## HTTP Server CGI processes

A major concern with CGI performance on other platforms is the fact that a CGI program is started on each Web client request. This includes additional disk and operating system activity to create the new process (job). Quite often, CGI program initialization, such as connecting to a database management system, also takes some time that adds to the response time users experience with such applications.

The IBM HTTP Server for i takes a different approach. The HTTP Server keeps a pool of HTTP server child processes that is used to run CGI programs. The child processes are not ended after a CGI program is run within the process. In addition, child processes are associated with a user profile and only requests for CGI programs that run under the same user profile associated with an existing child process will be run in the process.

Some of the additional features related to CGI processes include:

- The ability to specify how many child processes, and under what user profile, should be pre-started when the Web server starts so that Web clients do not incur the performance hit of starting a new CGI child process.

- The ability to run a CGI request in a pre-started CGI process, enabling the CGI program to be loaded and initialized at server startup. This support is beneficial for programs running in named activation groups. A CGI program running in a named activation group is loaded and initialized one time within a CGI process.

### **Persistent CGI programs**

*Persistent CGI* is an extension to the CGI interface that allows a CGI program to maintain a session with a browser client across multiple browser requests. This allows files to be left open, the state to be maintained, and long running database transactions to be committed or rolled-back based on end-user actions.

### **High availability CGI programs**

High availability CGI programs use APIs to preserve state information. The state information can be accessed by different IBM i servers that are participating as cluster nodes in a clustered environment, even after a failure or switchover of the HTTP Server or IBM i server.

**Note:** Although maintaining CGI program state information across multiple requests is a concept used by both persistent CGI and high availability CGI programs, the mechanisms used by the two types of programs are different and a high availability CGI program should not be confused with a persistent CGI program.

### **Running AIX® CGI programs**

The IBM HTTP Server for i is able to run AIX CGI programs by running the CGI program in the IBM Portable Application Solutions Environment for i.

In addition to running AIX CGI programs, the IBM HTTP Server for i is able to run AIX programs that implement the FastCGI protocol. FastCGI is an interface between Web servers and applications which combines some of the performance characteristics of native Web server modules with the Web server independence of the CGI programming interface. Like AIX CGI programs, AIX FastCGI applications are run in the PASE for i environment.

#### **Related information:**

“CGI programming” on page 182

The IBM HTTP Server for i supports the extension of the functionality of the HTTP Server through the use of Common Gateway Interface (CGI) programs.

“Writing persistent CGI programs” on page 191

*Persistent CGI* is an extension to the CGI interface that allows a CGI program to remain active across multiple browser requests and maintain a session with that browser client. This allows files to be left open, the state to be maintained, and long running database transactions to be committed or rolled-back based on end-user input.

“Writing high availability CGI programs” on page 189

High availability CGI programs use APIs to preserve state information. The state information can be accessed by different IBM i servers that are participating as cluster nodes in a clustered environment, even after a failure or switchover of the HTTP Server or IBM i server.

“Highly available HTTP Server” on page 43

The IBM HTTP Server for i supports Web server clusters, which ensures high availability of your Web site.

“Running CGI programs in IBM PASE for i” on page 197

The IBM HTTP Server for i Web server can run CGI programs created to run in the IBM Portable Application Solutions Environment for i. In addition, the HTTP Server can also run programs that follow the FastCGI protocol.

 [FastCGI Web site](#)

## | **Apache modules**

| Modules are service programs that can be dynamically linked and loaded to extend the nature of the HTTP Server.

| In this way, the Apache modules provide a way to extend the function of a Web server. Functions commonly added by optional modules include:

- | • Authentication
- | • Encryption
- | • Application support
- | • Logging
- | • Support for different content types
- | • Diagnostic support

| A good example of a module that is shipped with the HTTP Server that extends the reach of the core Apache server is:

```
| LoadModule ibm_ssl_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVSSL.SRVPGM
```

| This service program is only loaded, linked, and used when you configure the LoadModule directive because you decided to encrypt your data using Secure Sockets Layer (SSL). The advantage of this is that the core Apache program can stay relatively small and tight until a particular function (as provided by a plug-in module) is needed. Then, with just a LoadModule directive and optionally some configuration directives, you can increase the functionality of your Web server with a corresponding increase in the working set size.

| Apache core functions are those functions available in a standard Apache installation with no nonstandard modules. The HTTP Server supports more than a 250 directives. About 30 percent of those directives are in the core functions. The remainder of the directives are in separate modules. The LoadModule directive must be used to activate the directives in these modules.

| IBM provides Apache modules, typically called plug-ins, in order to extend the functionality of the Web server. The following is a list of the most commonly-used plug-ins:

### | **WebSphere Application Server plug-in**

| Forwards HTTP requests from the Web server to WebSphere Application Server. WebSphere Application Server is the premier application server for Java applications.

### | **Lotus Domino plug-in**

| Forwards HTTP requests from the Web server to Lotus Domino. Lotus Domino provides functions such as e-mail, workflow-based computing, and the integration and management of both structured and unstructured data.

### | **Integrated Web Application Server plug-in**

| Forwards HTTP requests from the Web server to Integrated Web Application Server. Integrated Web Application Server is a lightweight application server for Java applications that is integrated into the IBM i operating system.

| You can also write your own module to extend the core functionality of the HTTP Server.

### | **Related information:**

| “Apache module programming” on page 198

| The IBM HTTP Server for i supports the extension of the functionality of the HTTP Server through the use of third-party Apache modules.



## Service-side includes

Server-side includes (SSI) are the simplest way to add dynamic content to a Web site. A set of directives is embedded in the HTML code and is interpreted by the server before the document is sent to a client. SSI can be used to call a CGI program or return information about documents or the value of environment variables.

In the simplest sense, SSI allows for character substitution from within an HTML document.

SSI also supports the execution of simple conditional statements.

Table 10 lists the SSI commands supported by the HTTP Server.

Table 10. Supported SSI commands

Command	Description
echo	Prints one of the SSI or API variables.
exec	Calls a CGI program.
fsize	Prints the size of the specified file.
flastmod	Prints the last modification date of the specified file.
global	Same as set command.
include	Inserts the text of another file. Included files can be nested.
printenv	Prints all existing environment variables and their values.
set	Sets the value of an environment variable.

### Related information:

“Server-side include commands for HTTP Server” on page 502

This topic provides information about server-side include (SSI) commands for the IBM HTTP Server for i Web server.

## High availability

The IBM HTTP Server for i supports the ability for businesses to withstand Web server outages, including scheduled downtime. The HTTP Server provides high availability by using the IBM i clustering support.

### Related information:

IBM i Cluster technology

## Highly available HTTP Server

The IBM HTTP Server for i supports Web server clusters, which ensures high availability of your Web site.

If Web serving is a critical aspect of your business, you may want high availability and scalability of your Web server environment. High availability and scalability of the Web server environment can be achieved through the use of IBM i clustering.

The Web server cluster solution can provide:

- **Planned downtime:** If a Web server requires planned maintenance, it is possible to transfer the work to another node without visible service interruptions to the client.
- **No unplanned downtime:** If a machine fails, the work is transferred to another node with no human involvement and without visible service interruptions to the client.
- **Scalability:** When employing multiple nodes, it is possible to distribute the Web site workload over the cluster nodes.

Clusters are a collection of complete systems that work together to provide a single, unified computing capability.

A liveness monitor checks the state of the Web server and interacts with the Web server and the clustering resource services in the event that a Web server fails (failover), or a manual switchover takes place (ensures no interruption of Web server services). The clustered hash table (part of the state replication mechanism) can be used to replicate highly available CGI program state data across the cluster nodes so that the state data is available to all nodes in the event that a Web server fails (failover) or is switched-over manually (switchover). To take advantage of this capability, an existing CGI program must be enabled in a highly available Web Server environment. CGI programs write to the CGI APIs to indicate what data is replicated.

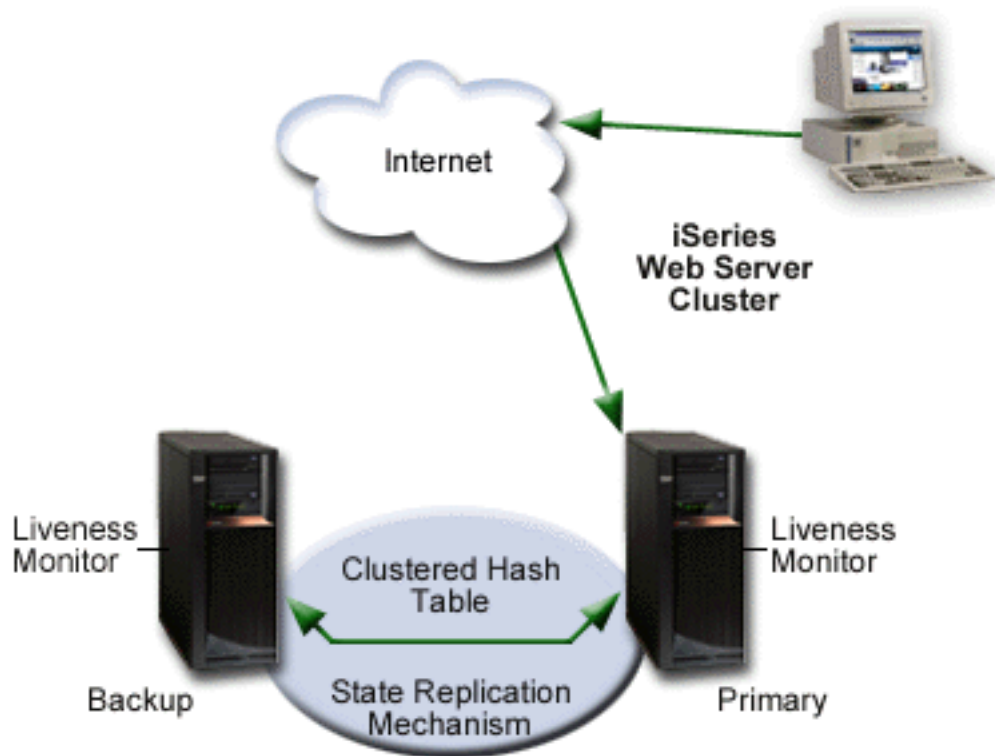
There are three Web server cluster models that are supported:

- “Primary/backup with takeover IP model”
- “Primary/backup with a network dispatcher model” on page 46
- “Peer model” on page 47

### **Primary/backup with takeover IP model**

In this model, the Web server runs on the primary and all backup nodes. The backup node or nodes are in a idle state, ready to become the primary Web server should the primary Web server fail (failover), or a switchover takes place. All client requests are always served by the primary node.

The following diagram illustrates a Primary/backup with takeover IP model.



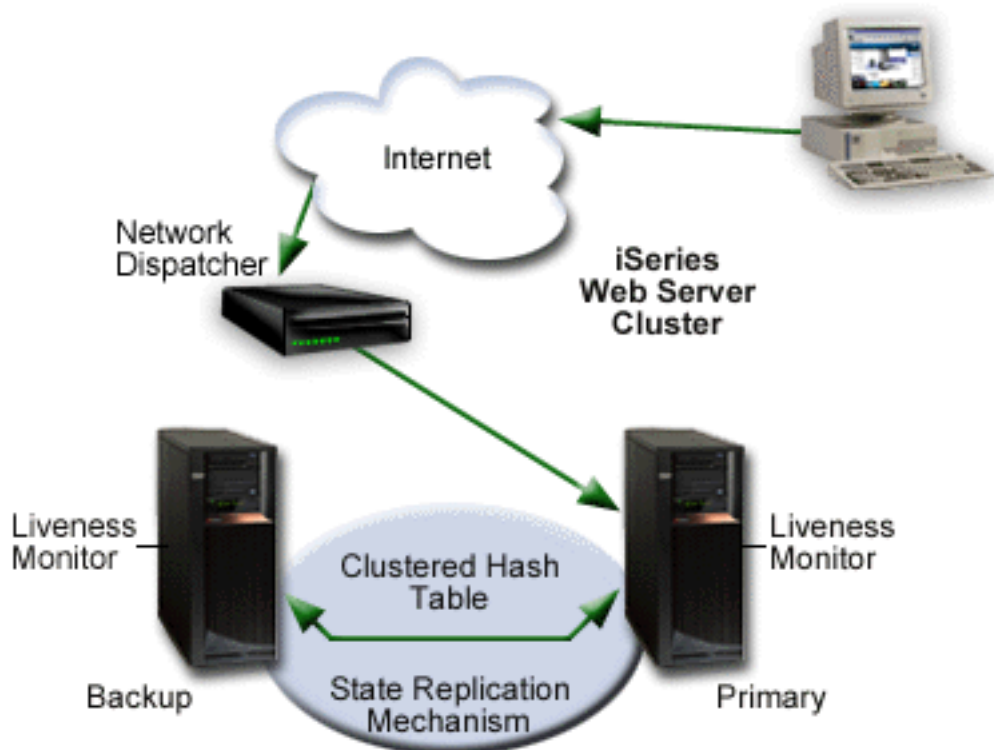
When the primary node fails (failover), or is brought down by the administrator, the failover/switchover process begins. The following steps are performed during failover/switchover:

1. One of the backup servers becomes the primary (the first backup in the switchover order).
2. The client requests are redirected to the new primary node.
3. If the new primary receives a user request that belongs to a long-running-session (a CGI program that has been updated to be a highly available CGI program), the server will restore the request's state. The new primary retrieves that highly available CGI program's state information from the clustered hash table. The clustered hash table is part of the state replication mechanism.
4. After the failed node recovers, the highly available Web server instance can be restarted and it will become the backup system. If the system administrator wants the failed node to become primary again, a manual switchover must be performed (this can be accomplished with the IBM Simple Cluster Management interface available through System i Navigator or a business partner tool).

## Primary/backup with a network dispatcher model

In this model, just like the primary/backup with takeover IP model, the Web server runs on the primary and all backup nodes. The backup nodes are in an idle state and all client requests are served by the primary node. A network dispatcher (for example the IBM WebSphere Edge Server) sends client requests to the Web server.

The following diagram illustrates a Primary/backup with a network dispatcher model.



When the primary node fails (failover), or a switchover takes place, the failover/switchover process begins. The following steps are performed during failover/switchover:

1. One of the backup servers becomes the primary (the first backup in the switchover order).
2. The client requests are sent to the new primary node by the network dispatcher.
3. If the new primary receives a user request that belongs to a long-running-session, the server needs to restore the request's state. The new primary searches for the state either locally or in the clustered hash table. The clustered hash table is part of the state replication mechanism.

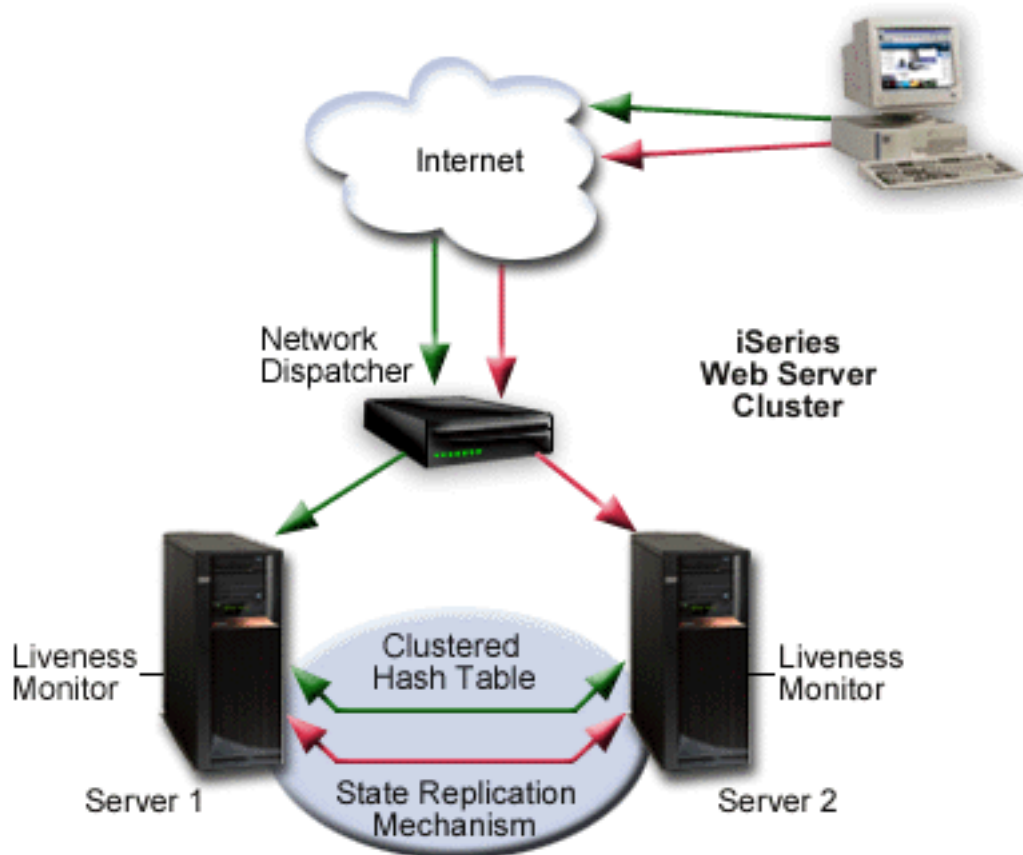
4. After the failed node recovers, the system administrator can restart the Web server instance and it will become a backup Web server. If the system administrator wants the failed node to become primary again, a manual switchover must be performed.

**Note:** A node can join a recovery domain as primary only if the cluster resource group is in inactive mode.

### Peer model

In this model, there is no declared primary node. All nodes are in an active state and serve client requests. A network dispatcher (for example the IBM WebSphere Edge Server) evenly distributes requests to different cluster nodes. This guarantees distribution of resources in case of heavy load. Linear scalability is not guaranteed beyond a small number of nodes. After some number of nodes are added, scalability can disappear, and the cluster performance can deteriorate.

The following diagram illustrates the peer model.



## High availability CGI programs

High availability CGI programs use APIs to preserve state information. The state information can be accessed by different IBM i servers that are participating as cluster nodes in a clustered environment, even after a failure or switchover of the HTTP Server or IBM i server.

See “Writing high availability CGI programs” on page 189 for information about writing high availability CGI programs.

## Web Publishing with the PUT Method

An IBM HTTP Server for i instance can be configured to support the PUT method for Web publishing.

The standard way of uploading files to a Web server using HTTP is through the use of the PUT method. HTTP Server supports the PUT method, but requires additional setup to tell the server how to handle incoming PUT requests. One way to accomplish this is to enable WebDAV, which is provided with HTTP Server through the module `mod_dav`. Another is to provide your own CGI program and configure it for use with HTTP Server. This topic discusses both options, as well as the PUT method in general.

### About the PUT Method

POST and PUT are two methods in the HTTP specification that are used to permanently change files on a Web server. While the POST method is used in conjunction with preestablished content such as Web forms, the PUT method involves manipulating files that do not yet exist on the server. HTTP Server supports the POST and PUT methods in the same way -- that is, it requires a program to tell it how to handle incoming requests.

### WebDAV

Most users will find that the easiest way to implement the PUT method for HTTP Server is to enable WebDAV and use a client program that supports WebDAV (such as Microsoft Web Folders) to upload files. WebDAV is a set of extensions to the HTTP protocol, and is included in HTTP Server through the module `mod_dav`. In addition to the WebDAV extensions, `mod_dav` includes a PUT handler.

For more information on WebDAV, including a list of all the methods included, see “WebDAV for HTTP Server” on page 49 and “Setting up WebDAV for HTTP Server” on page 121.

### CGI programs

Alternatively, you can provide your own CGI program to handle incoming PUT requests, and configure it for use with HTTP Server. A program that handles PUT requests operates much like a program that handles POST requests, but must include additional code for writing (and overwriting) files on the server.

Because a PUT action results in a permanent change on the server, it's important to be aware of the security issues involved in providing your own PUT-handling CGI program. Some of these issues include:

- Ensuring the user making the PUT request is authorized to update files on the server
- Making sure only Web content files are updated
- Only updating content the user is authorized to update

For a more detailed discussion on providing your own PUT-handling CGI program, see the Apache Week article [Publishing Pages with PUT](#) .

Once you have a program capable of handling PUT requests, you can configure it for use with HTTP Server using the `Script` directive. For more information on the `Script` directive, see “Module `mod_actions`” on page 221.

## WebDAV for HTTP Server


This topic provides information about Web-based distributed authoring and versioning (WebDAV) for the IBM HTTP Server for i Web server.

Web-based distributed authoring and versioning (WebDAV) is a set of extensions to the HTTP protocol that allows WebDAV clients (such as Microsoft Web Folders) to collaboratively edit and manage files on remote Web servers. Major features of WebDAV include:

- File locking so that two or more users do not overwrite the same file.
- XML data to store properties data such as author information.
- Copy and move operations so that directory structures can be modified.

WebDAV is a set of extensions to the HTTP protocol. The following table defines the HTTP methods and the WebDAV extensions. Note that two methods, DELETE and PUT, are defined in the HTTP 1.1 specification, but modified by WebDAV.

Method	Specifications	Description
COPY	WebDAV	Copies the resource.
DELETE	HTTP 1.1/WebDAV	Deletes the resource.
GET	HTTP 1.1	Gets the contents of the resource.
HEAD	HTTP 1.1	Returns the message headers from a message sent to the server.
LOCK	WebDAV	Locks the resource.
MKCOL	WebDAV	Creates the collection specified.
MOVE	WebDAV	Moves the resource.
OPTIONS	HTTP 1.1	Performs an option call to the server.
POST	HTTP 1.1	Action defined by the server.
PROPFIND	WebDAV	Performs a property find on the server.
PROPPATCH	WebDAV	Sets or removes properties on the server.
PUT	HTTP 1.1/WebDAV	Puts the contents of the resource to the server in the specified location.
TRACE	HTTP 1.1	Does a trace call to the server.
UNLOCK	WebDAV	Unlocks the resource.

See RFC2518  for more information on WebDAV.

### Related information:

“WebDAV tasks” on page 121

Web-based distributed authoring and versioning (WebDAV) is provided through the IBM HTTP Server for i Web server.

---

## Scenarios: HTTP Server

This topic provides information on how to use the IBM Web Administration for i interface to set up or manage your IBM HTTP Server for i Web server, step-by-step. Each task is specific and includes a usable HTTP Server configuration file when completed.

The JKL Toy Company (JKL), a fictitious company, scenarios will take you through the same processes employees of the JKL Toy Company followed while working with the Web Administration for i interface. Follow the scenario steps, and all prerequisites, to complete the scenario successfully.

The given examples may be used to successfully complete the scenario; however, you may enter your own information at any time. If you are not familiar with the Web Administration for i interface or Web serving, it is suggested that you use the given examples and follow the scenarios closely in the order they are given.

Replace examples in brackets, [...], with your own HTTP Server information. For example,

**http://[Systemi\_name]:[port]**

When instructions are given in the following format, replace the words in the brackets, such as [Systemi\_name], with what is being asked for. For example,

**http://jklserver:2001**

## **JKL Toy Company creates an HTTP Server**

This scenario discusses how to create an IBM HTTP Server for i Web server on an IBM i server.

### **Scenario**

The JKL Toy Company (a fictitious company) wants to run a Web site on their IBM i server. The examples used in this scenario show the **Create New HTTP Server wizard** being used to create an HTTP Server instance called **JKLTEST** which will use **all IP addresses**, port **1975** on an IBM i server designated **JKL\_SERVER**.

### **Prerequisites**

- It is assumed you have read “Scenarios: HTTP Server” on page 49.

### **Start the IBM Web Administration for i interface**

Access the IBM Web Administration for i from your browser. For information about how to access the Web Administration for i interface, see “Starting Web Administration for i” on page 7.

### **Create your HTTP Server**

The Web Administration for i interface allows you to create, set up, and manage multiple servers.

1. Click the **Setup** tab.
2. Expand **Common Tasks and Wizards**.
3. Click **Create HTTP Server**.
4. Enter a descriptive, unique name in the **Server name** field.  
Example: JKLTEST
5. Click **Next**.
6. Accept the default value.  
Example: /www/jkltest
7. Click **Next**.
8. Accept the default value.  
Example: /www/jkltest/htdocs
9. Click **Next**.
10. Accept the default values or replace with your own unique IP address and port.  
Example: IP address All Addresses  
Example: Port 1975
11. Click **Next**.



12. **Optional:** Select **Yes** to use an access log.  
Select **No** if you do not want to create an access log at this time. By default, the log will be created for you.
13. Click **Next**.
14. Accept the default values to specify the length of time to keep the log files or update with your preferences.
15. Click **Next**.
16. Review the displayed information. If any information is incorrect, click **Back** and correct it.
17. Click **Finish** to create your new HTTP Server.

**Note:** If the wizard fails and you receive an error message, check your Webmaster user profile authorities.

## Restart your HTTP Server

Select one of the following methods below:

### Manage one server

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the Server list.
4. Click the **Stop** icon if the server is running.
5. Click the **Start** icon.

### Manage all servers

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select **All Servers** from the Server list.
4. Click the **All HTTP Servers** tab.
5. Select your HTTP Server name in the table.  
Example: JKLTEST
6. Click **Stop** if the server is running.
7. Click **Start**.

**Note:** If your HTTP Server does not start, see “Troubleshooting” on page 202.

## Test your HTTP Server

1. Open a new Web browser.
2. Enter **http://[your\_hostname]:[port]** in the location or URL field .  
Example: http://jkl\_server:1975

Your new HTTP Server will display a generic HTML file provided by the Web Administration for i interface.

## View your HTTP Server configuration

Your configuration will look similar if you used the given example in this and previous examples.

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.

Example: JKLTEST

#### 4. Expand **Tools**.

#### 5. Click **Display Configuration File**.

```
Listen *:1975
DocumentRoot /www/jkltest/htdocs
ServerRoot /www/jkltest
Options -ExecCGI -FollowSymLinks -SymLinksIfOwnerMatch -Includes -IncludesNoExec -Indexes -MultiViews
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%{Cookie}n \"%r\" %t" cookie
LogFormat "%{User-agent}i" agent
LogFormat "%{Referer}i -> %U" referer
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log combined
SetEnvIf "User-Agent" "Mozilla/2" nokeepalive
SetEnvIf "User-Agent" "JDK/1\0" force-response-1.0
SetEnvIf "User-Agent" "Java/1\0" force-response-1.0
SetEnvIf "User-Agent" "RealPlayer 4\0" force-response-1.0
SetEnvIf "User-Agent" "MSIE 4\0b2;" nokeepalive
SetEnvIf "User-Agent" "MSIE 4\0b2;" force-response-1.0
<Directory />
  Order Deny,Allow
  Deny From all
</Directory>
<Directory /www/jkltest/htdocs>
  Order Allow,Deny
  Allow From all
</Directory>
```

## JKL Toy Company adds a new directory to HTTP Server

This scenario discusses how to add a directory to an HTTP Server Web server.

### Scenario

The JKL Toy Company (a fictitious company) has a need to add a directory to their **JKLTEST** configuration. The JKL Web administrator wants to create a directory to keep online employee profile information, such as current projects and contact information. Due to the large number of employees, a separate directory will be created to contain the employee profile information. The new directory will be called **profiles**.

### Prerequisites

- It is assumed you have read “Scenarios: HTTP Server” on page 49.
- It is assumed you have read and completed “JKL Toy Company creates an HTTP Server” on page 50 or you have an existing HTTP Server configuration.

### Start the IBM Web Administration for i interface

Access the IBM Web Administration for i from your browser. For information about how to access the Web Administration for i interface, see “Starting Web Administration for i” on page 7.

### Create a new directory

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.  
Example: JKLTEST
4. Select **Global configuration** from the **Server area** list.
5. Expand **HTTP Tasks and Wizards**.

6. Click **Add a Directory to the Web**.
7. Click **Next**.
8. Select **Static Web pages and files**.
9. Click **Next**.
10. **Optional**: Accept the default or enter a new directory name.  
Example: /www/jkltest/profiles/
11. Click **Next**.
12. **Optional**: Accept the default or enter a new alias name.  
Example: /profiles/
13. Click **Next**.
14. Click **Finish**.

## Restart your HTTP Server

Select one of the following methods below:

### Manage one server

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the Server list.
4. Click the **Stop** icon if the server is running.
5. Click the **Start** icon.

### Manage all servers

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select **All Servers** from the Server list.
4. Click the **All HTTP Servers** tab.
5. Select your HTTP Server name in the table.  
Example: JKLTEST
6. Click **Stop** if the server is running.
7. Click **Start**.

**Note**: If your HTTP Server does not start, see “Troubleshooting” on page 202.

## Test your HTTP Server

1. Open a new Web browser.
2. Enter **http://[i\_hostname]:[port]/[new\_directory\_alias]/** in the location or URL field.  
Example: http://jkl\_server:1975/profiles/

Your new directory will display a generic HTML file provided by the IBM Web Administration for i interface.

## View your HTTP Server configuration

Your configuration will look similar if you used the given example in this and previous examples.

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.

Example: JKLTEST

#### 4. Expand Tools.

#### 5. Click **Display Configuration File**.

```
Alias /profiles/ /www/jkltest/profiles/
Listen *:1975
DocumentRoot /www/jkltest/htdocs
ServerRoot /www/jkltest
Options -ExecCGI -FollowSymLinks -SymLinksIfOwnerMatch -Includes -IncludesNoExec -Indexes -MultiViews
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%{Cookie}n \"%r\" %t" cookie
LogFormat "%{User-agent}i" agent
LogFormat "%{Referer}i -> %U" referer
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log combined
SetEnvIf "User-Agent" "Mozilla/2" nokeepalive
SetEnvIf "User-Agent" "JDK/1\.\0" force-response-1.0
SetEnvIf "User-Agent" "Java/1\.\0" force-response-1.0
SetEnvIf "User-Agent" "RealPlayer 4\.\0" force-response-1.0
SetEnvIf "User-Agent" "MSIE 4\.\0b2;" nokeepalive
SetEnvIf "User-Agent" "MSIE 4\.\0b2;" force-response-1.0
<Directory />
  Order Deny,Allow
  Deny From all
</Directory>
<Directory /www/jkltest/profiles>
  Order Allow,Deny
  Allow From all
</Directory>
<Directory /www/jkltest/htdocs>
  Order Allow,Deny
  Allow From all
</Directory>
```

## JKL Toy Company adds user directories for HTTP Server

This scenario discusses how to add a user directory in an IBM HTTP Server for i Web server.

### Scenario

The JKL Toy Company (a fictitious company) has decided to allow employees to maintain their own personal Web pages. The JKL Web administrator wants the personal Web pages to be stored in a directory of the root file system called **/home** on the **JKLTEST** HTTP Server. The directory **/home** will contain one subdirectory for each employee.

To begin, the JKL Web administrator creates a user profile and user directory for fellow employee Sharon Jones on the IBM i server. The new user profile will be called **SJONES** and the new user directory will be located at **/home/sjones**.

### Prerequisites

- It is assumed you have read “Scenarios: HTTP Server” on page 49.
- It is assumed you have read and completed “JKL Toy Company creates an HTTP Server” on page 50 or you have an existing HTTP Server configuration.
- It is assumed you have read and completed “JKL Toy Company adds a new directory to HTTP Server” on page 52.
- It is assumed you have installed and are familiar with System i Navigator.
- It is assumed you have read “User profiles and required authorities for HTTP Server” on page 30.

## Create a new user profile with System i Navigator

For in-depth information on how to use the System i Navigator, read the System i Navigator help installed with the product.

**Note:** It is not necessary to create a new user profile on your IBM i server if you want to use an existing profile. If using an existing profile, make certain the user profile has the appropriate permissions.

1. Start **System i Navigator**.
2. Expand the IBM i server the HTTP Server is installed on.  
Example: JKL\_SERVER
3. Select **Users and Groups**, or click the Users and Group icon in the toolbar.
4. Click **Create a new user**, or click the Create a New Use icon in the toolbar.
5. Enter a new user name.  
Example: SJONES
6. **Optional:** Enter a description for this new profile.  
Example: This is a test profile.
7. **Optional:** Add a password if necessary for your IBM i server.
8. Click **Capabilities**.
9. Set the system privileges to allow the new user profile to use the HTTP Server.
10. Click **OK**.
11. Click **Add**.

## Create a new user directory with System i Navigator

**Note:** The `/home` directory comes preinstalled on your IBM i server.

1. Start **System i Navigator**.
2. Expand the IBM i server the HTTP Server is installed on.  
Example: JKL\_SERVER
3. Expand **File Systems > Integrated File System > Root**.
4. Right-click directory **home**.
5. Click **New Folder**.
6. Enter the name of your new user profile.  
Example: sjones
7. Click **OK**.

## Copy HTML welcome page to user directory with System i Navigator

The new user directory does not contain any files. Use System i Navigator to copy **index.html**, found in `/www/[server_name]/htdocs` directory of your HTTP Server, to your new user directory.

Example: `/www/jkltest/htdocs`

1. Start **System i Navigator**.
2. Expand the IBM i server the HTTP Server is installed on.  
Example: JKL\_SERVER
3. Expand **File Systems > Integrated File Systems > Root > www > [server\_name] > htdocs**.  
Example: `/www/jkltest/htdocs`
4. Right-click file **index.html**.
5. Click **Copy**.

6. Right-click the new user directory.

Example: sjones

7. Click **Paste**.

**Optional:** Edit file index.html in any way you choose. This is the file the HTTP Server will look for when this directory is requested by the Web browser.

## Start the IBM Web Administration for i interface

Access the IBM Web Administration for i from your browser. For information about how to access the Web Administration for i interface, see “Starting Web Administration for i” on page 7.

### Set up user directories for HTTP Server

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.  
Example: JKLTEST
4. Select **Global configuration** from the **Server area** list.
5. Expand **Server properties**.
6. Click **URL Mapping**.
7. Click the **User Directories** tab in the form.
8. Select **Disable all users except for the following** under **Enable or Disable user directories**.
9. Click **Add** under the **Enabled users** table.
10. Enter the name of your new user profile.  
Example: sjones
11. Click **Continue**.
12. Click **Add** under the **Current user directories** table.
13. Enter **/home** in the User directories column.

**Note:** The order in which the user directories are listed determines which directory the HTTP Server will use first. If a match is not found in the first (top) user directory, the next user directory listed will be used. This continues until a match is found.

14. Click **Continue**.
15. Click **OK**.

### Set up /home directory for HTTP Server

After creating the user directory, you must set up your HTTP Server to provide access to directory **/home**.

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select **Global configuration** from the **Server area** list.
4. Expand **Server Properties**.
5. Click **Container Management**.
6. Click the **Directories** tab in the form.
7. Click **Add** under **Directory/Directory Match containers** table.
8. Select **Directory** from the list in the **Type** column.
9. Enter **/home** in the **Directory path or expression** column.
10. Click **Continue**.

11. Click **OK**.
12. Select **Directory /home** from the **Server area** list.
13. Click **Security**.
14. Click the **Control Access** tab in the form.
15. Select **Deny then allow** from the **Order for evaluating access** list under **Control access based on where the request is coming from**.
16. Select **Allow access to all, except the following** under **Control access based on where the request is coming from**.

**Note:** Do not add restrictions at this time. Return to this form at the end of the scenario to add restrictions.

17. Click **OK**.

## Restart your HTTP Server

Select one of the following methods below:

### Manage one server

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the Server list.
4. Click the **Stop** icon if the server is running.
5. Click the **Start** icon.

### Manage all servers

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select **All Servers** from the Server list.
4. Click the **All HTTP Servers** tab.
5. Select your HTTP Server name in the table.  
Example: JKLTEST
6. Click **Stop** if the server is running.
7. Click **Start**.

**Note:** If your HTTP Server does not start, see “Troubleshooting” on page 202.

## Test your HTTP Server

1. Open a new Web browser.
2. Enter **http://[i\_hostname]:[port]/~[user\_directory]** in the location or URL field .  
Example: `http://jkl_server:1975/~sjones`

Your new user directory will display the generic HTML file copied from directory /htdocs.

## View your HTTP Server configuration

Your configuration will look similar if you used the given example in this and previous examples.

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.

Example: JKLTEST

#### 4. Expand Tools.

#### 5. Click Display Configuration File.

```
Listen *:1975
DocumentRoot /www/jkltest/htdocs
ServerRoot /www/jkltest
Options -ExecCGI -FollowSymLinks -SymLinksIfOwnerMatch -Includes -IncludesNoExec -Indexes -MultiViews
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%{Cookie}n \"%r\" %t" cookie
LogFormat "%{User-agent}i" agent
LogFormat "%{Referer}i -> %U" referer
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log combined
SetEnvIf "User-Agent" "Mozilla/2" nokeepalive
SetEnvIf "User-Agent" "JDK/1\." force-response-1.0
SetEnvIf "User-Agent" "Java/1\." force-response-1.0
SetEnvIf "User-Agent" "RealPlayer 4\." force-response-1.0
SetEnvIf "User-Agent" "MSIE 4\." nokeepalive
SetEnvIf "User-Agent" "MSIE 4\." force-response-1.0
UserDir Disable
UserDir Enable Sjones
UserDir /home
<Directory />
  Order Deny,Allow
  Deny From all
</Directory>
<Directory /www/jkltest/htdocs>
  Order Allow,Deny
  Allow From all
</Directory>
<Directory /home>
  Order Deny, Allow
  Allow From all
</Directory>
```

## JKL Toy Company enables cookie tracking on HTTP Server

This scenario discusses how to enable cookie tracking for an IBM HTTP Server for i Web server.

### Scenario

The JKL Toy Company (a fictitious company) wants to be able to measure Web site visitor activity and trends. The JKL Web administrator would like to try to measure how many new and unique users visit the intranet Web site. Requiring users to obtain a userid and password is the most accurate way track users; however, this method has the disadvantage of forcing the intranet Web users to register for a userid and password.

Analyzing the data in the log file by IP address could be used to track users. Two disadvantages to this method are:

- Some ISPs use dynamic IP addressing, assigning random IP addresses to all users.
- Some ISPs send all traffic through a proxy server, creating a log entry for the IP address of the proxy server only.

Setting a unique number in a cookie in the user's browser the first time that they visit the Web site combined with using a log that records cookies could be used to track users. This log can be analyzed to show how many new cookies have been set and how many old cookies have returned. In addition, the log can also be used to show the sequence of URLs that a particular cookie used to navigate through the Web site. A downside of this method is that users can shut off the browsers ability to record cookies.



The JKL Web administrator has decided to use the cookie method. The JKL Web administrator will store cookie information in a new log called **JKLCOOKIE\_LOG** using a new cookie called **JKLCOOKIE**.

## Prerequisites

- It is assumed you have read “Scenarios: HTTP Server” on page 49.
- It is assumed you have read and completed “JKL Toy Company creates an HTTP Server” on page 50 or you have an existing HTTP Server configuration.

## Start the IBM Web Administration for i interface

Access the IBM Web Administration for i from your browser. For information about how to access the Web Administration for i interface, see “Starting Web Administration for i” on page 7.

## Create a cookie for HTTP Server

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.  
Example: JKLTEST
4. Select **Global configuration** from the **Server area** list.
5. Expand **Server Properties**.
6. Click **Logging**.
7. Click the **User Tracking (Cookies)** tab in the form.
8. Select **Enabled** from the **Track user requests in a cookie** list.
9. Enter a name for the cookie in the **Cookie name** field or use the default.  
Example: JKLCOOKIE
10. Enter a value in the **Expiration period** field.  
Example: 1
11. Select a time period from the **Expiration period** list.  
Example: Years
12. Click **OK**.

## Set up the cookie log for HTTP Server

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select **Global configuration** from the **Server area** list.
4. Expand **Server Properties**.
5. Click **Logging**.
6. Click the **Custom Logs** tab in the form.
7. Click **Add** under the **Custom logs** table.
8. Enter **logs/[log\_name]** in the **Log** column.  
Example: logs/jklcookie\_log
9. Select **cookie** from the **Log format** list.
10. Enter a value in the **Expiration** field.  
Example: 364
11. Select a time period from the **Expiration** list.  
Example: Days
12. Click **Continue**.

13. Click **OK**.

**Note:** The rest of the fields on this form are optional.

## Restart your HTTP Server

Select one of the following methods below:

### Manage one server

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the Server list.
4. Click the **Stop** icon if the server is running.
5. Click the **Start** icon.

### Manage all servers

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select **All Servers** from the Server list.
4. Click the **All HTTP Servers** tab.
5. Select your HTTP Server name in the table.  
Example: JKLTEST
6. Click **Stop** if the server is running.
7. Click **Start**.

**Note:** If your HTTP Server does not start, see "Troubleshooting" on page 202.

## Test your HTTP Server

1. Open a new Web browser.
2. Turn cookie alerts on in your browser. Consult the Web browser's help documentation for details on enabling cookie alerts.
3. Enter **http://[i\_hostname]:[port]** in the location or URL field.  
Example: `http://jkl_server:1975`

## View your HTTP Server configuration

Your configuration will look similar if you used the given example in this and previous examples.

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.  
Example: JKLTEST
4. Expand **Tools**.
5. Click **Display Configuration File**.

```
Listen *:1975
DocumentRoot /www/jkltest/htdocs
ServerRoot /www/jkltest
Options -ExecCGI -FollowSymLinks -SymLinksIfOwnerMatch -Includes -IncludesNoExec -Indexes -MultiViews
LogMaint logs/jklcookie_log 364 0
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%{Cookie}n \"%r\" %t" cookie
LogFormat "%{User-agent}i" agent
```

```

LogFormat "%{Referer}i -> %U" referer
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log combined
CustomLog logs/jklcookie_log cookie
SetEnvIf "User-Agent" "Mozilla/2" nokeepalive
SetEnvIf "User-Agent" "JDK/1\0" force-response-1.0
SetEnvIf "User-Agent" "Java/1\0" force-response-1.0
SetEnvIf "User-Agent" "RealPlayer 4\0" force-response-1.0
SetEnvIf "User-Agent" "MSIE 4\0b2;" nokeepalive
SetEnvIf "User-Agent" "MSIE 4\0b2;" force-response-1.0
CookieTracking On
CookieName JKLCOOKIE
CookieExpires 31536000
<Directory />
  Order Deny,Allow
  Deny From all
</Directory>
<Directory /www/jkltest/htdocs>
  Order Allow,Deny
  Allow From all
</Directory>

```

## JKL Toy Company creates virtual hosts on HTTP Server

This scenario discusses how to create virtual hosts in an IBM HTTP Server for i Web server.

### Scenario

The JKL Toy Company (a fictitious company) wants to serve two domain names from one IP address. This can be done using virtual hosts.

The JKL Web administrator has decided to use the name-based virtual host for HTTP Server JKLTEST. The ISP has configured the Domain Name Server to route requests for JKLINFO to IP address 9.5.61.228, port 78.

### Prerequisites

- It is assumed you have read “Scenarios: HTTP Server” on page 49.
- It is assumed you have read and completed “JKL Toy Company creates an HTTP Server” on page 50 or you have an existing HTTP Server configuration.
- It is assumed you are familiar with Domain Name Servers (DNS).

### Start the IBM Web Administration for i interface

Access the IBM Web Administration for i from your browser. For information about how to access the Web Administration for i interface, see “Starting Web Administration for i” on page 7.

### Set up a name-based virtual host

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.  
Example: JKLTEST
4. Select **Global configuration** from the **Server area** list.
5. Expand **Server Properties**.
6. Click **Virtual Hosts**.
7. Click the **Name-based** tab in the form.
8. Click **Add** under the **Named virtual hosts** table.
9. Select or enter an IP address in the **IP address** column.

Example: 9.5.61.228

**Note:** The IP address 9.5.61.288 used in this scenario is associated with JKL Toy Company's IBM i hostname **JKLINFO** and registered by a Domain Name Server (DNS). You will need to choose a different IP address and hostname. The IBM Web Administration for i interface provides the IP addresses used by your IBM i system in the IP Address list; however, you will need to provide the hostname associated with the address you choose.

10. Enter a port number in the **Port** column.

Example: 78

11. Click **Add** under the **Virtual host containers** table in the **Named host** column.
12. Enter the fully qualified server hostname for the virtual host in the **Server name** column.  
Example: JKLINFO.com

**Note:** Make sure the server hostname you enter is fully qualified and associated with the IP address you selected.

13. Enter a document root for the virtual host index file or welcome file in the **Document root** column.  
Example: /www/jkltest/companyinfo/

**Note:** You are specifying a document root using a directory that will be added below in the *Set up the virtual host directories* section.

14. Click **Continue**.
15. Click **OK**.

### Set up Listen directive for virtual host

1. Expand **Server Properties**.
2. Click **General Server Configuration**.
3. Click the **General Settings** tab in the form.
4. Click **Add** under the **Server IP addresses and ports to listen on** table.
5. Select the IP address you entered for the virtual host in the **IP address** column.  
Example: 9.5.61.288
6. Enter the port number you entered for the virtual host in the **Port** column.  
Example: 78
7. Accept **Disabled** default for FRCA.
8. Click **Continue**.
9. Accept the default values for the remainder of the form.
10. Click **OK**.

### Set up the virtual host directories

1. Select the virtual host from the **Server area** list.
2. Expand **HTTP Tasks and Wizards**.
3. Click **Add a Directory to the Web**.
4. Click **Next**.
5. Select **Static web pages and files**.
6. Click **Next**.
7. Enter a directory name for the virtual host in the **Name** field.  
Example: /www/jkltest/companyinfo/
8. Click **Next**.
9. Enter an alias for the virtual host in the **Alias** field.

Example: /companyinfo/

10. Click **Next**.
11. Click **Finish**.

The document root and directory for the virtual host has been created.

## Restart your HTTP Server

Select one of the following methods below:

### Manage one server

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the Server list.
4. Click the **Stop** icon if the server is running.
5. Click the **Start** icon.

### Manage all servers

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select **All Servers** from the Server list.
4. Click the **All HTTP Servers** tab.
5. Select your HTTP Server name in the table.  
Example: JKLTEST
6. Click **Stop** if the server is running.
7. Click **Start**.

**Note:** If your HTTP Server does not start, see “Troubleshooting” on page 202.

## Test your HTTP Server

1. Start a new Web browser.
2. Enter **http://[virtual\_hostname\_name]:[port]** in the location or URL field.  
Example: http://JKLINFO:78

## View your HTTP Server configuration

Your configuration will look similar if you used the given example in this and previous examples.

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.  
Example: JKLTEST
4. Expand **Tools**.
5. Click **Display Configuration File**.

```
Listen *:1975
Listen 9.5.61.228:78
DocumentRoot /www/jkltest/htdocs
ServerRoot /www/jkltest
Options -ExecCGI -FollowSymLinks -SymLinksIfOwnerMatch -Includes -IncludesNoExec -Indexes -MultiViews
NameVirtualHost 9.5.61.228:78
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%{Cookie}n \"%r\" %t" cookie
```

```

LogFormat "%{User-agent}i" agent
LogFormat "%{Referer}i -> %U" referer
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log combined
SetEnvIf "User-Agent" "Mozilla/2" nokeepalive
SetEnvIf "User-Agent" "JDK/1\0" force-response-1.0
SetEnvIf "User-Agent" "Java/1\0" force-response-1.0
SetEnvIf "User-Agent" "RealPlayer 4\0" force-response-1.0
SetEnvIf "User-Agent" "MSIE 4\0b2;" nokeepalive
SetEnvIf "User-Agent" "MSIE 4\0b2;" force-response-1.0
<Directory />
  Order Deny,Allow
  Deny From all
</Directory>
<Directory /www/jkltest/htdocs>
  Order Allow,Deny
  Allow From all
</Directory>
<VirtualHost 9.5.61.228:78>
  ServerName JKLINFO.com
  DocumentRoot /www/jkltest/companyinfo/
  <Directory /www/jkltest/companyinfo>
    Order Allow,Deny
    Allow From all
  </Directory>
  Alias /companyinfo/ /www/jkltest/companyinfo/
</VirtualHost>

```

#### Related information:

“Virtual hosts on HTTP Server” on page 23

This topic provides information about virtual host types on the IBM HTTP Server for i Web server.

“Virtual host tasks” on page 132

This topic provides step-by-step tasks for configuring virtual hosts in the IBM HTTP Server for i Web server.

## JKL Toy Company adds password protection for HTTP Server

This scenario discusses how to add password protection to an IBM HTTP Server for i Web server.

### Scenario

The JKL Toy Company (a fictitious company) wants to protect a set of Web pages on its Web site so that they can only be viewed by visitors that have a password. In order to add password protection, JKL needs to decide what type of authentication method to use:

- Internet user - requires an entry in a validation list.
- User profile - requires an IBM i server user profile.
- LDAP - requires an LDAP server.

JKL Toy Company chooses to use Internet users for the following reasons:

- User profiles are not desirable since JKL does not want to create a user profile for each authenticated visitor to the Web site.
- Since JKL only wants to implement authentication on one IBM i server, validation lists will be used. LDAP is a better solution for multiple systems.

The Web page content to be protected is in the preexisting directory `/www/jkltest/profiles/`. The visitor's user name and passwords will be stored in a new validation list called `users` in library `PROFILES`. The first user name that we will enter is `sjones` with a password of `dragon102`.

### Prerequisites

- It is assumed you have read “Scenarios: HTTP Server” on page 49.

- It is assumed you have read and completed “JKL Toy Company creates an HTTP Server” on page 50 or you have an existing HTTP Server configuration.
- It is assumed you have read and completed “JKL Toy Company adds a new directory to HTTP Server” on page 52.
- It is assumed you have access to or the correct authority to create an IBM i library.

## Create a library for validation lists on your IBM i server

Skip the following steps if you will be using an existing library on your IBM i server for your validation list.

1. Start a 5250 session on your system.
2. Enter **CRTL** on the command line.
3. Type the **F4** key to prompt for additional parameters.
4. Enter a name for your library in the **Library** field.  
Example: PROFILES
5. **Optional:** Edit the remaining fields as necessary or accept the default values.
6. Type the **Enter** key (or equivalent) to create your library.

Make sure the proper authorities and restrictions you want on the library are active before continuing.

## Start the IBM Web Administration for i interface

Access the IBM Web Administration for i from your browser. For information about how to access the Web Administration for i interface, see “Starting Web Administration for i” on page 7.

## Set up password protection for a directory on HTTP Server

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.  
Example: JKLTEST
4. Select **Directory /www/[server\_name]/[new\_directory]/** from the **Server area** list.  
Example: /www/jkltest/profiles/

**Note:** The new directory was created with the “JKL Toy Company adds a new directory to HTTP Server” on page 52 scenario.

5. Expand **Server Properties**.
6. Click **Security**.
7. Click the **Authentication** tab in the form.
8. Select **Use Internet users in validation lists**.
9. Enter a descriptive name in the **Authentication name or realm** field.  
Example: JKL Employee Profiles

**Note:** When users attempt to access a password protected resource, they are challenged for a username and password. The **Authentication name or realm** value is displayed in the login window, and should provide information regarding the resource the user is attempting to access.

10. Click **Add** under **Validation lists** table.
11. Enter **[library]/[validation\_list\_name]**.  
Example: profiles/users

**Note:** In the above example, **profiles** is the name of the IBM i library and **users** is the name of the validation list.

12. Click **Continue**.
13. Select **Default server profile** from the **IBM i user profile to process requests** list under **Related information**. When selected, the value **%%SERVER%%** will be placed in the field.
14. Click **Apply**.
15. Click the **Control Access** tab in the form.
16. Select **All authenticated users (valid user name and password)** under **Control access based on who is making the requests**.
17. Click **OK**.

## Create a validation list for HTTP Server

1. Click the **Advanced** tab.
2. Click the **Internet Users and Groups** subtab.
3. Expand **Internet Users and Groups**.
4. Click **Add Internet User**.
5. Enter **[username]** into the **User name** field.  
Example: sjones
6. Enter **[password]** into the **Password** field.  
Example: dragon102
7. Enter the same password in the **Confirm password** field.
8. **Optional:** Enter comments for this Internet user.
9. Enter **[library]/[validation\_list\_name]** in the **Validation list** field.  
Example: profiles/users

**Note:** In the above example, **profiles** is the name of the library and **users** is the name of the validation list.

10. Click **Apply**.

## Restart your HTTP Server

Select one of the following methods below:

### Manage one server

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the Server list.
4. Click the **Stop** icon if the server is running.
5. Click the **Start** icon.

### Manage all servers

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select **All Servers** from the Server list.
4. Click the **All HTTP Servers** tab.
5. Select your HTTP Server name in the table.  
Example: JKLTEST
6. Click **Stop** if the server is running.



7. Click **Start**.

**Note:** If your HTTP Server does not start, see “Troubleshooting” on page 202.

## Test your HTTP Server

1. Open a new Web browser.
2. Enter `http://[i_hostname]:[port]/[new_directory_alias]/` in the location or URL field.  
Example: `http://jkl_server:1975/profiles/`
3. Enter the username and password you created.

You will be asked to provide a valid username and password. Enter the username and password you entered in the validation list. It is suggested you limit \*PUBLIC authority, but allow authority to the Web administrator user authority and QTMHHTTP.

## View your HTTP Server configuration

Your configuration will look similar if you used the given example in this and previous examples.

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.  
Example: JKLTEST
4. Expand **Tools**.
5. Click **Display Configuration File**.

```
Alias /profiles/ /www/jkltest/profiles/
Listen *:1975
DocumentRoot /www/jkltest/htdocs
ServerRoot /www/jkltest
Options -ExecCGI -FollowSymLinks -SymLinksIfOwnerMatch -Includes -IncludesNoExec -Indexes -MultiViews
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%{Cookie}n \"%r\" %t" cookie
LogFormat "%{User-agent}i" agent
LogFormat "%{Referer}i -> %U" referer
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log combined
SetEnvIf "User-Agent" "Mozilla/2" nokeepalive
SetEnvIf "User-Agent" "JDK/1\.\.0" force-response-1.0
SetEnvIf "User-Agent" "Java/1\.\.0" force-response-1.0
SetEnvIf "User-Agent" "RealPlayer 4\.\.0" force-response-1.0
SetEnvIf "User-Agent" "MSIE 4\.\.0b2;" nokeepalive
SetEnvIf "User-Agent" "MSIE 4\.\.0b2;" force-response-1.0
<Directory />
  Order Deny,Allow
  Deny From all
</Directory>
<Directory /www/jkltest/profiles>
  Order Allow,Deny
  Allow From all
  Require valid-user
  PasswdFile profiles/users
  UserID %SERVER%
  AuthType Basic
  AuthName "JKL Employee Profiles"
</Directory>
<Directory /www/jkltest/htdocs>
  Order Allow,Deny
  Allow From all
</Directory>
```

## JKL Toy Company adds dynamic content with server-side includes for HTTP Server

This scenario discusses how to add dynamic content to an IBM HTTP Server for i Web server using server-side includes.

### Scenario

The JKL Toy company (a fictitious company) wants to add some dynamic content to their index file (or welcome page) on their Web site. The welcome Web page is located in /www/jkltest/htdocs. The JKL Web administrator will add the current server time to display on their Web page.

**Note:** Server-side includes (SSI) create dynamic Web pages by adding content to a Web page before it is sent to the browser. Server performance may be impacted when processing SSIs.

### Prerequisites

- It is assumed you have read “Scenarios: HTTP Server” on page 49.
- It is assumed you have read and completed “JKL Toy Company creates an HTTP Server” on page 50 or you have an existing HTTP Server configuration.
- It is assumed you have installed and are familiar with System i Navigator.

### Edit the index file (or welcome page) with System i Navigator

For in-depth information on how to use the System i Navigator, read the System i Navigator help installed with the product.

1. Start **System i Navigator**.
2. Expand the IBM i server the HTTP Server is installed on.  
Example: JKL\_SERVER
3. Expand **File Systems > Integrated File System > Root > www > [server\_name]**.  
Example: File Systems > Integrated File System > Root > www > jkltest
4. Click **htdocs**.  
The directory **htdocs** is the default name of your document root provided by the Create New HTTP Server wizard.
5. Right-click **index.html**.
6. Click **Rename**.
7. Rename the file **index.shtml**.
8. Right-click **index.shtml**.
9. Click **Edit**.
10. Enter the following lines below the <BODY> tag and before the </BODY> tag:  

```
<p>The current server time is:  
<!--#config timefmt="%T" -->  
<!--#echo var="DATE_LOCAL" -->  
</p>
```
11. Save and close the file.

See “Server-side include commands for HTTP Server” on page 502 for more information about SSI commands.

### Start the IBM Web Administration for i interface

Access the IBM Web Administration for i from your browser. For information about how to access the Web Administration for i interface, see “Starting Web Administration for i” on page 7.

## Set up server-side includes for HTTP Server

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.  
Example: JKLTEST
4. Select **Global configuration** from the **Server area** list.
5. Expand **Server properties**.
6. Click **Container Management**.
7. Click the **Files** tab in the form.
8. Click **Add** under the **Files/Files Match containers** table.
9. Select **Files Match** from the list in the **Type** column.
10. Enter `\.shtml(\..+)?$` in the **File name or expression** column.
11. Click **Continue**.
12. Click **OK**.
13. Select **Files Match \.shtml(\..+)?\$** from the **Server area** list.
14. Expand **Server Properties**.
15. Click **Dynamic Content and CGI**.
16. Click the **Server Side Includes** tab in the form.
17. Select **Allow server side files without CGI** under **Server side includes**.
18. Click **OK**.
19. Select **Global configuration** from the **Server area** list.
20. Expand **Server Properties**.
21. Click **General Server Configuration**.
22. Click the **Welcome Pages** tab in the form.
23. Select **index.html** in the **Welcome/index file names** table.
24. Rename the file **index.shtml** in the **File name** column.
25. Click **Continue**.
26. Click **OK**.

## Restart your HTTP Server

Select one of the following methods below:

### Manage one server

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Click the **Stop** icon if the server is running.
5. Click the **Start** icon.

### Manage all servers

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select **All Servers** from the **Server** list.
4. Click the **All HTTP Servers** tab.
5. Select your HTTP Server name in the table.

Example: JKLTEST

6. Click **Stop** if the server is running.
7. Click **Start**.

**Note:** If your HTTP Server does not start, see “Troubleshooting” on page 202.

## Test your HTTP Server

1. Start a new Web browser.
2. Enter **http://[i\_hostname]:[port]** in the location or URL field.

Example: `http://jkl_server:1975`

The Web page now displays the current server time.

## View your HTTP Server configuration

Your configuration will look similar if you used the given example in this and previous examples.

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.

Example: JKLTEST

4. Expand **Tools**.
5. Click **Display Configuration File**.

```
Listen *:1975
DocumentRoot /www/jkltest/htdocs
ServerRoot /www/jkltest
Options -ExecCGI -FollowSymLinks -SymLinksIfOwnerMatch -Includes -IncludesNoExec -Indexes -MultiViews
AccessFileName .htaccess
LogFormat "%h %l %u %t \"%r\" %|s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%{Cookie}n \"%r\" %t" cookie
LogFormat "%{User-agent}i" agent
LogFormat "%{Referer}i -| %U" referer
LogFormat "%h %l %u %t \"%r\" %|s %b" common
CustomLog logs/access_log combined
SetEnvIf "User-Agent" "Mozilla/2" nokeepalive
SetEnvIf "User-Agent" "JDK/1\0" force-response-1.0
SetEnvIf "User-Agent" "Java/1\0" force-response-1.0
SetEnvIf "User-Agent" "RealPlayer 4\0" force-response-1.0
SetEnvIf "User-Agent" "MSIE 4\0b2;" nokeepalive
SetEnvIf "User-Agent" "MSIE 4\0b2;" force-response-1.0
DirectoryIndex index.shtml
<Directory />
  Order Deny,Allow
  Deny From all
</Directory>
<Directory /www/jkltest/htdocs>
  Order Allow,Deny
  Allow From all
</Directory>
<FilesMatch \.shtml(\..+)?>
  Options +IncludesNoExec
  AddOutputFilter INCLUDES .shtml
</FilesMatch>
```

## JKL Toy company enables Secure Sockets Layer (SSL) protection on HTTP Server

This scenario discusses how to enable SSL protection for an IBM HTTP Server for i Web server.

## Scenario

The JKL Toy company (a fictitious company) wants to enable Secure Sockets Layer (SSL) protection for a specific directory on their HTTP Server. The secured directory will contain confidential corporate earnings information that only a select group of employees and business associates will be able to access. The JKL Web administrator has decided not to create and deploy user certificates to client browsers, but rather use SSL so that all data exchanged with the browser is encrypted. The JKL Web administrator will use a server certificate, basic password protection (based upon existing IBM i user accounts), and standard SSL encryption to provide access to the secured information.

**Note:** Although JKL chooses not to implement digital certificates, they must still register their HTTP Server with the IBM i Digital Certificate Manager.

## Prerequisites

- It is assumed you have read “Scenarios: HTTP Server” on page 49.
- It is assumed you have read and completed “JKL Toy Company creates an HTTP Server” on page 50 or you have an existing HTTP Server configuration.
- It is assumed that a certificate authority (and certificate store) is already established for the Digital Certificate Manager.
- It is assumed you are familiar with Domain Name Servers (DNS).

## Start the IBM Web Administration for i interface

Access the IBM Web Administration for i from your browser. For information about how to access the Web Administration for i interface, see “Starting Web Administration for i” on page 7.

## Set up a name-based virtual host

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.  
Example: JKLTEST
4. Select **Global configuration** from the **Server area** list.
5. Expand **Server Properties**.
6. Click **Virtual Hosts**.
7. Click the **Name-based** tab in the form.
8. Click **Add** under the **Named virtual hosts** table.
9. Select or enter an IP address in the **IP address** column.

Example: 9.5.61.228

**Note:** The IP address 9.5.61.228 used in this scenario is associated with JKL Toy Company's IBM i hostname **JKLEARNINGS** and registered by a Domain Name Server (DNS). You will need to choose a different IP address and hostname. The IBM Web Administration for i interface provides the IP addresses used by your IBM i server in the IP Address list; however, you will need to provide the hostname associated with the address you choose.

10. Enter a port number in the **Port** column.

Example: 443

**Note:** Specify a port number other than the one currently being used for your HTTP Server to maintain an SSL and non-SSL Web site.

11. Click **Add** under the **Virtual host containers** table in the **Named host** column.

**Note:** This is a table within the **Named virtual hosts** table in the **Named host** column.

12. Enter the fully qualified server hostname for the virtual host in the **Server name** column.

Example: www.JKLEARNINGS.org

**Note:** Make sure the server hostname you enter is fully qualified and associated with the IP address you selected.

13. Enter a document root for the virtual host index file or welcome file in the **Document root** column.

Example: /www/jkltest/earnings/

**Note:** You are specifying a document root that will be created below. Remember the document root you have entered; you will be asked to enter the document root again when creating a new directory.

14. Click **Continue**.

15. Click **OK**.

### Set up Listen directive for virtual host

1. Expand **Server Properties**.

2. Click **General Server Configuration**.

3. Click the **General Settings** tab in the form.

4. Click **Add** under the **Server IP addresses and ports to listen** on table.

5. Select the IP address you entered for the virtual host in the **IP address** column.

Example: 9.5.61.288

6. Enter the port number you entered for the virtual host in the **Port** column.

Example: 443

7. Click **Continue**.

8. Click **OK**.

### Set up the virtual host directories

1. Select the virtual host from the **Server area** list.

2. Expand **HTTP Tasks and Wizards**.

3. Click **Add a Directory to the Web**.

4. Click **Next**.

5. Select **Static web pages and files**.

6. Click **Next**.

7. Enter a directory name for the virtual host in the **Name** field.

Example: /www/jkltest/earnings/

8. Click **Next**.

9. Enter an alias for the virtual host in the **Alias** field.

Example: /earnings/

10. Click **Next**.

11. Click **Finish**.

The document root and directory for the virtual host has been created.

### Set up password protection via authentication

1. Select the directory under the virtual host from the **Sever area** list.

Example: Directory /www/jkltest/earnings

2. Expand **Server Properties**.

3. Click **Security**.
4. Click the **Authentication** tab in the form.
5. Select **Use IBM i profile of client** under **User authentication method**.
6. Enter **Projected Earnings** in the **Authentication name or realm** field.
7. Select **Default server profile** from the **IBM i user profile to process requests** list under **Related information**. When selected, the value **%%SERVER%%** will be placed in the field.
8. Click **Apply**.
9. Click the **Control Access** tab in the form.
10. Click **All authenticated users (valid user name and password)** under **Control access based on who is making the request**.
11. Click **OK**.

### Enable SSL for the virtual host

1. Select the virtual host from the **Sever area** list.  
Example: Virtual Host \*:443
2. Expand **Server Properties**.
3. Click **Security**.
4. Click the **SSL with Certificate Authentication** tab in the form.
5. Select **Enable SSL** under **SSL**.
6. Select **QIBM\_HTTP\_SERVER\_[server\_name]** from the **Server certificate application name** list.  
Example: QIBM\_HTTP\_SERVER\_JKLTEST

**Note:** Remember the name of the server certificate. You will need to select it again in the Digital Certificate Manager.

7. Select **Do not request client certificate for connection** under **Client certificates when establishing the connection**.
8. Click **OK**.

The **HTTPS\_PORT** provides a specific environment variable value that is passed to CGI programs . This field is not used in this scenario.

### Associate system certificate with HTTP Server

The application name (created during the SSL process) is assigned a system certificate via the Digital Certificate Manager (DCM). During the process of enabling SSL for a virtual host, an IBM i server certificate must be assigned to the application name used when configuring SSL. This task is accomplished via the Digital Certificate Manager interface (accessed from the IBM i Tasks screen). See IBM i Digital Certificate Manager for more information.

**Note:** The following steps will require a user profile with higher levels of authority than those documented for the Webmaster profile. Web browsers will need to be restarted using the higher authority profile to authenticate.

1. Click the **Related Links** tab.
2. Click **Digital Certificate Manager**.
3. Click **Select a Certificate Store**.
4. Select **\*SYSTEM**.
5. Click **Continue**.
6. Enter a password in the Certificate store password field.
7. Click **Continue**.

8. Click **Manage Applications**.
9. Select **Update certificate assignment**.
10. Click **Continue**.
11. Select **Server**.
12. Click **Continue**.
13. Select the appropriate application name.

**Note:** Select the application name created while enabling SSL for the virtual host directory.

Example: QIBM\_HTTP\_SERVER\_JKLTEST

14. Click **Update Certificate Assignment**.
15. Select the appropriate certificate.
16. Click **Assign New Certificate**. This assigns the certificate to the application name selected in the previous step.

## Restart your HTTP Server

Select one of the following methods below:

### Manage one server

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the Server list.
4. Click the **Stop** icon if the server is running.
5. Click the **Start** icon.

### Manage all servers

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select **All Servers** from the Server list.
4. Click the **All HTTP Servers** tab.
5. Select your HTTP Server name in the table.  
Example: JKLTEST
6. Click **Stop** if the server is running.
7. Click **Start**.

**Note:** If your HTTP Server does not start, see “Troubleshooting” on page 202.

## Test your HTTP Server

1. Start a new Web browser.
2. Enter **https://[virtual\_hostname\_name]:[port]** in the location or URL field.  
Example: **https://www.JKLEARNINGS.org:443**

You will be challenged for a user name and password. After entering an appropriate IBM i user name and password, you will see a sample homepage (created by the Serve New Directory wizard) with the browser's security padlock icon enabled. The padlock indicates that SSL is enabled.



## View your HTTP Server configuration

Your configuration will look similar if you used the given example in this and previous examples.

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.

Example: JKLTEST

4. Expand **Tools**.
5. Click **Display Configuration File**.

```
LoadModule ibm_ssl_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVSSL.SRVPGM
Listen *:1975
Listen 9.5.61.228:443
DocumentRoot /www/jkltest/htdocs
ServerRoot /www/jkltest
Options -ExecCGI -FollowSymLinks -SymLinksIfOwnerMatch -Includes -IncludesNoExec -Indexes -MultiViews
NameVirtualHost 9.5.61.228:443
AccessFileName .htaccess
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%{Cookie}n \"%r\" %t" cookie
LogFormat "%{User-agent}i" agent
LogFormat "%{Referer}i -> %U" referer
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log combined
SetEnvIf "User-Agent" "Mozilla/2" nokeepalive
SetEnvIf "User-Agent" "JDK/1\0" force-response-1.0
SetEnvIf "User-Agent" "Java/1\0" force-response-1.0
SetEnvIf "User-Agent" "RealPlayer 4\0" force-response-1.0
SetEnvIf "User-Agent" "MSIE 4\0b2;" nokeepalive
SetEnvIf "User-Agent" "MSIE 4\0b2;" force-response-1.0
DirectoryIndex index.html
<Directory />
  Order Deny,Allow
  Deny From all
</Directory>
<Directory /www/jkltest/htdocs>
  Order Allow,Deny
  Allow From all
</Directory>
<VirtualHost 9.5.61.228:443>
  ServerName www.JKLEARNINGS.org
  DocumentRoot /www/jkltest/earnings/
  SSLEnable
  SSLAppName QIBM_HTTP_SERVER_JKLTEST
  SSLClientAuth None
  <Directory /www/jkltest/earnings>
    Order Allow,Deny
    Allow From all
    Require valid-user
    PasswdFile %%SYSTEM%%
    UserID %%SERVER%%
    AuthType Basic
    AuthName "Projected Earnings"
  </Directory>
  Alias /earnings/ /www/jkltest/earnings/
</VirtualHost>
```

## JKL Toy Company enables single signon for HTTP Server

This scenario discusses how to enable single signon for security for an IBM HTTP Server for i Web server.

To learn more about Kerberos and network security on IBM i servers, see Network authentication service.

## Scenario

The JKL Web administrator, John Day, wants to enable single signon for the JKL Toy Company network. The network consists of several IBM i systems and a Windows 2000 server, where the users are registered in Microsoft Windows Active Directory. Based on John Day's research, he knows that Microsoft Active Directory uses the Kerberos protocol to authenticate Windows users. John Day also knows that IBM i provides a single signon solution based on an implementation of Kerberos authentication, called network authentication service, in conjunction with Enterprise Identity Mapping (EIM).

While excited about the benefits of a single signon environment, John Day wants to thoroughly understand single signon configuration and usage before using it across the entire enterprise. Consequently, John Day decides to configure a test environment first.

After considering the various groups in the company, John Day decides to create the test environment for the MYCO Order Receiving department, a subsidiary of JKL Toys. The employees in the Order Receiving department use multiple applications, including HTTP Server, on one IBM i system to handle incoming customer orders. John Day uses the Order Receiving department as a testing area to create a single signon test environment that can be used to better understand how single signon works and how to plan a single signon implementation across the JKL enterprise.

### This scenario has the following advantages:

- Allows you to see some of the benefits of single signon on a small scale to better understand how you can take full advantage of it before you create a large-scale, single signon environment.
- Provides you with a better understanding of the planning process required to successfully and quickly implement a single signon environment across your entire enterprise.

As the network administrator at JKL Toy Company, John Day wants to create a small single signon test environment that includes a small number of users and a single IBM i server, *Systemi A*. John Day wants to perform thorough testing to ensure that user identities are correctly mapped within the test environment. The first step is to enable a single signon environment for the IBM i server and applications on *Systemi A*, including the HTTP Server. After implementing the configuration successfully, John Day eventually wants to expand the test environment to include the other systems and users in the JKL enterprise.

### The objectives of this scenario are as follows:

- The IBM i system, known as Systemi A, must be able to use Kerberos within the MYCO.COM realm to authenticate the users and services that are participating in this single signon test environment. To enable the system to use Kerberos, Systemi A must be configured for network authentication service.
- The directory server on Systemi A must function as the domain controller for the new EIM domain.

**Note:** Two types of domains play key roles in the single signon environment: an EIM domain and a Windows 2000 domain. Although both of these terms contain the word *domain*, these entities have very different definitions.

Use the following descriptions to understand the differences between these two types of domains. For more information about these terms, see the EIM and Network authentication service topics.

#### EIM domain

An EIM domain is a collection of data, which includes the EIM identifiers, EIM associations, and EIM user registry definitions that are defined in that domain. This data is stored in a Lightweight Directory Access Protocol (LDAP) server, such as the IBM Tivoli® Directory Server for i5/OS, which can run on any system in the network defined in that domain. Administrators can configure systems (EIM clients), such as IBM i, to participate in the domain so that systems and applications can use domain data for EIM lookup operations and identity mapping. To find out more about an EIM domain, see EIM.

#### Windows 2000 domain

In the context of single signon, a Windows 2000 domain is a Windows network that contains

several systems that operate as clients and servers, as well as a variety of services and applications that the systems use. The following are some of the components pertinent to single signon that you may find within a Windows 2000 domain:

– **Realm**

A realm is a collection of machines and services. The main purpose of a realm is to authenticate clients and services. Each realm uses a single Kerberos server to manage the principals for that particular realm.

– **Kerberos server**

A Kerberos server, also known as a key distribution center (KDC), is a network service that resides on the Windows 2000 server and provides tickets and temporary session keys for network authentication service. The Kerberos server maintains a database of principals (users and services) and their associated secret keys. It is composed of the authentication server and the ticket granting server. A Kerberos server uses Microsoft Windows Active Directory to store and manage the information in a Kerberos user registry.

**Note:** These servers should be in the same subnet to ensure that the tokens can be validated.

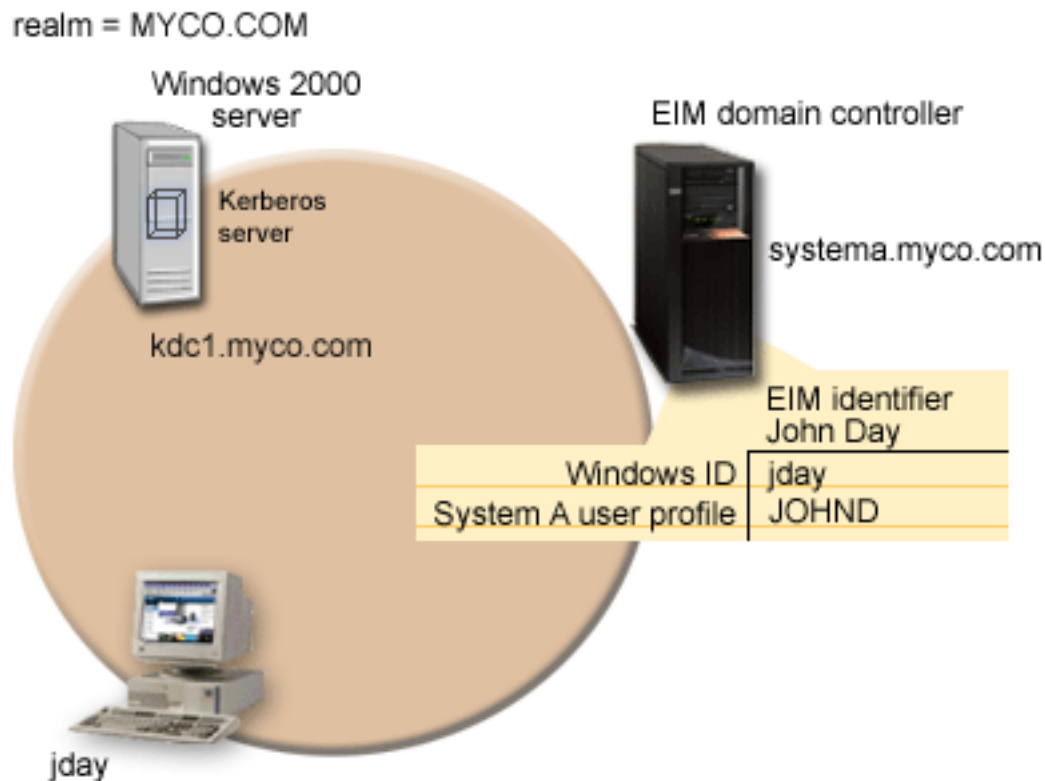
– **Microsoft Windows Active Directory**

Microsoft Windows Active Directory is an LDAP server that resides on the Windows 2000 server along with the Kerberos server. The Active Directory is used to store and manage the information in a Kerberos user registry. Microsoft Windows Active Directory uses Kerberos authentication as its default security mechanism. Therefore, if you are using Microsoft Active Directory to manage your users, you are already using Kerberos technology.

- One user profile on *Systemi A* and one Kerberos principal must each be mapped to a single EIM identifier.
- A Kerberos service principal must be used to authenticate the user to the IBM HTTP Server for i.

## Details

The following figure illustrates the network environment for this scenario:



The figure illustrates the following points relevant to this scenario.

### EIM domain data defined for the enterprise

- An EIM domain called *MyCoEimDomain*.
- An EIM registry definition for *Systemi A* called *SystemiA.MYCO.COM*.
- An EIM registry definition for the Kerberos registry called *MYCO.COM*.
- An EIM identifier called John Day. This identifier uniquely identifies John Day, the administrator for *MyCo*.
- A source association for the *jday* Kerberos principal on the Windows 2000 server.
- A target association for the *JOHND* user profile on *Systemi A* to access HTTP Server.

### Windows 2000 server

- Acts as the Kerberos server (*kdc1.myco.com*), also known as a key distribution center (KDC), for the network.
- The default realm for the Kerberos server is *MYCO.COM*.
- A Kerberos principal of *jday* is registered with the Kerberos server on the Windows 2000 server. This principal will be used to create a source association to the EIM identifier, John Day.

### Systemi A

- Runs IBM i 5.4, or later, with the following options and licensed products installed:
  - IBM i Host Servers
  - Qshell Interpreter

- IBM i Access for Windows
- Network Authentication Enablement
- The IBM Tivoli Directory Server for i5/OS (LDAP) on *Systemi A* will be configured to be the EIM domain controller for the new EIM domain, *MyCoEimDomain*. *Systemi A* participates in the EIM domain, *MyCoEimDomain*.
- The principal name for *Systemi A* is *krbsvr400/Systemia.myco.com@MYCO.COM*.
- The principal name for the HTTP Server on *Systemi A* is *HTTP/Systemia.myco.com@MYCO.COM*.
- The user profile of *JOHND* exists on *Systemi A*. You create a target association between this user profile and the EIM identifier, *John Day*.
- The home directory for the IBM i user profile, *JOHND*, (*/home/JOHND*) is defined on *Systemi A*.

#### Client PC used for single signon administration

- Runs Microsoft Windows 2000 operating system.
- Runs IBM i Access for Windows V5R4, or later.
- Runs System i Navigator with the following subcomponents installed:
  - Network
  - Security
- Serves as the primary logon system for administrator John Day.
- Configured to be part of the *MYCO.COM* realm (Windows domain).

#### Prerequisites

Successful implementation of this scenario requires that the following assumptions and prerequisites are met:

1. It is assumed you have read “Scenarios: HTTP Server” on page 49.
2. All system requirements, including software and operating system installation, have been verified. Ensure that all the necessary licensed programs are installed. To verify that the licensed programs have been installed, complete the following:
  - a. In System i Navigator, expand your **system** > **Configuration and Service** > **Software** > **Installed Products**.
3. All necessary hardware planning and setup is complete.
4. TCP/IP and basic system security are configured and tested on each system.
5. The directory server and EIM are not previously configured on *Systemi A*.

**Note:** Instructions in this scenario are based on the assumption that the directory server has not been previously configured on *Systemi A*. However, if you have previously configured the directory server, you can still use these instructions with only slight differences. These differences are noted in the appropriate places within the configuration steps.

6. A single DNS server is used for host name resolution for the network. Host tables are not used for host name resolution.

**Note:** The use of host tables with Kerberos authentication may result in name resolution errors or other problems.

#### Configuration steps:

**Note:** Before you implement this scenario, you need to thoroughly understand the concepts related to single signon, including network authentication service and Enterprise Identity Mapping (EIM). See the following information to learn about the terms and concepts related to single signon:

- Enterprise Identity Mapping (EIM)
- Network authentication service

These are the configuration steps John Day completed. Follow these configuration steps to enable a single signon environment for your IBM i system.

“Step 1: Planning work sheet”

“Step 2: Create a basic single signon configuration for *Systemi A*” on page 83

“Step 3: Add principal names to the KDC” on page 85

“Step 4: Add Kerberos keytab” on page 86

“Step 5: Create home directory for *John Day* on *Systemi A*” on page 86

“Step 6: Test network authentication service configuration on *Systemi A*” on page 86

“Step 7: Create EIM identifier for *John Day*” on page 87

“Step 8: Create a source association and target association for the new EIM identifier” on page 87

“Step 9: Configure IBM i Access for Windows applications to use Kerberos authentication” on page 88

“Step 10: Add *Systemi A* to an existing EIM domain” on page 88

“Step 11: Configure HTTP Server for single signon” on page 89

“Step 12: (Optional) Post configuration considerations” on page 90

### Step 1: Planning work sheet:

The following planning work sheets are tailored to fit this scenario. These planning work sheets demonstrate the information that you need to gather and the decisions you need to make to prepare the single signon implementation described by this scenario. To ensure a successful implementation, you must be able to answer **Yes** to all prerequisite items in the work sheet and be able to gather all the information necessary to complete the work sheets before you perform any configuration tasks.

Table 11. Single signon prerequisite work sheet


Prerequisite work sheet	Answers
Are you running IBM i 5.4 or later?	Yes
Are the following options and licensed products installed on <i>Systemi A</i> ? <ul style="list-style-type: none"> <li>• IBM i Host Servers</li> <li>• Qshell Interpreter</li> <li>• IBM i Access for Windows</li> <li>• Network Authentication Enablement</li> </ul>	Yes
Have you installed an application that is enabled for single signon on each of the PCs that will participate in the single signon environment? <b>Note:</b> For this scenario, all of the participating PCs have IBM i Access for Windows installed and <i>Systemis A</i> has the IBM HTTP Server for i installed.	Yes
Is System i Navigator installed on the administrator's PC? <ul style="list-style-type: none"> <li>• Is the Security subcomponent of System i Navigator installed on the administrator's PC?</li> <li>• Is the Network subcomponent of System i Navigator installed on the administrator's PC?</li> </ul>	Yes
Have you installed the latest IBM i Access for Windows service pack? See System i Access  for the latest service pack.	Yes
Do you, the administrator, have *SECADM, *ALLOBJ, and *IOSYSCFG special authorities?	Yes

Table 11. Single signon prerequisite work sheet (continued)

Prerequisite work sheet	Answers
<p>Do you have one of the following systems in the network acting as the Kerberos server (also known as the KDC)? If yes, specify which system.</p> <ol style="list-style-type: none"> <li>Windows 2000 Server <b>Note:</b> Microsoft Windows 2000 Server uses Kerberos authentication as its default security mechanism.</li> <li>Windows Server 2003</li> <li>IBM Portable Application Solutions Environment for i</li> <li>AIX server</li> <li>System z®</li> </ol>	Yes, Windows 2000 Server
Are all your PCs in your network configured in a Windows (R) 2000 domain?	Yes
Have you applied the latest program temporary fixes (PTFs)?	Yes
Is the IBM i system time within 5 minutes of the system time on the Kerberos server? If not see Synchronize system times.	Yes

You need this information to configure EIM and network authentication service to create a single signon test environment.

Table 12. Single signon configuration planning work sheet for Systemi A.

Use the following information to complete the EIM Configuration wizard. The information in this work sheet correlates with the information you need to supply for each page in the wizard:

Configuration planning work sheet for Systemi A	Answers
<p>How do you want to configure EIM for your system?</p> <ul style="list-style-type: none"> <li>Join an existing domain</li> <li>Create and join a new domain <b>Note:</b> This option allows you to configure the current system's directory server as the EIM domain controller when the directory server is not already configured as the EIM domain controller.</li> </ul>	<p>Create and join a new domain <b>Note:</b> This will configure the directory server on the same system on which you are currently configuring EIM.</p>
<p>Do you want to configure network authentication service?</p> <p><b>Note:</b> You must configure network authentication service to configure single signon.</p>	Yes
<p>The Network Authentication Service wizard launches from the EIM Configuration wizard. Use the following information to complete the Network Authentication Service wizard: <b>Note:</b> You can launch the Network Authentication Service wizard independently of the EIM Configuration wizard.</p>	
<p>What is the name of the Kerberos default realm to which your system belongs?</p> <p><b>Note:</b> A Windows 2000 domain is similar to a Kerberos realm. Microsoft Windows Active Directory uses Kerberos authentication as its default security mechanism.</p>	MYCO.COM
Are you using Microsoft Active Directory?	Yes

Table 12. Single signon configuration planning work sheet for Systemi A (continued).

Use the following information to complete the EIM Configuration wizard. The information in this work sheet correlates with the information you need to supply for each page in the wizard:

Configuration planning work sheet for Systemi A	Answers
What is the Kerberos server, also known as a key distribution center (KDC), for this Kerberos default realm? What is the port on which the Kerberos server listens?	<b>KDC:</b> <i>kdc1.myco.com</i> <b>Port:</b> <i>88</i> <b>Note:</b> This is the default port for the Kerberos server.
Do you want to configure a password server for this default realm? If yes, answer the following questions:  What is name of the password server for this Kerberos server? What is the port on which the password server listens?	Yes <b>Password server:</b> <i>kdc1.myco.com</i> <b>Port:</b> <i>464</i> <b>Note:</b> This is the default port for the Kerberos server.
For which services do you want to create keytab entries? • IBM i Kerberos Authentication • LDAP • IBM HTTP Server for i • IBM i NetServer	IBM i Kerberos Authentication <b>Note:</b> A keytab entry for HTTP Server must be done manually as described later in the configuration steps.
What is the password for your service principal or principals?	<i>Systemisa123</i> <b>Note:</b> Any and all passwords specified in this scenario are for example purposes only. To prevent a compromise to your system or network security, never use these passwords as part of your own configuration.
Do you want to create a batch file to automate adding the service principals for Systemi A to the Kerberos registry?	Yes
Do you want to include passwords with the IBM i service principals in the batch file?	Yes
As you exit the Network Authentication Service wizard, you will return to the EIM Configuration wizard. Use the following information to complete the EIM Configuration wizard:	
Specify user information for the wizard to use when configuring the directory server. This is the connection user. You must specify the port number, administrator distinguished name, and a password for the administrator. <b>Note:</b> Specify the LDAP administrator's distinguished name (DN) and password to ensure the wizard has enough authority to administer the EIM domain and the objects in it.	<b>Port:</b> <i>389</i> <b>Distinguished name:</b> <i>cn=administrator</i> <b>Password:</b> <i>mycopwd</i> <b>Note:</b> Any and all passwords specified in this scenario are for example purposes only. To prevent a compromise to your system or network security, do not use these passwords as part of your own configuration.
What is the name of the EIM domain that you want to create?	<i>MyCoEimDomain</i>
Do you want to specify a parent DN for the EIM domain?	No
Which user registries do you want to add to the EIM domain?	Local IBM i-- <i>SystemiA.MYCO.COM</i> Kerberos-- <i>MYCO.COM</i> <b>Note:</b> The Kerberos principals stored on the Windows 2000 server are not case sensitive; therefore do not select <b>Kerberos user identities are case sensitive.</b>



Table 12. Single signon configuration planning work sheet for Systemi A (continued).

Use the following information to complete the EIM Configuration wizard. The information in this work sheet correlates with the information you need to supply for each page in the wizard:

Configuration planning work sheet for Systemi A	Answers
Which EIM user do you want Systemi A to use when performing EIM operations? This is the system user <b>Note:</b> If you have not configured the directory server prior to configuring single signon, the only distinguished name (DN) you can provide for the system user is the LDAP administrator's DN and password.	<b>User type:</b> Distinguished name and password <b>User:</b> <i>cn=administrator</i> <b>Password:</b> <i>mycopwd</i> <b>Note:</b> Any and all passwords specified in this scenario are for example purposes only. To prevent a compromise to your system or network security, never use these passwords as part of your own configuration.
After you complete the EIM Configuration wizard, use the following information to complete the remaining steps required for configuring single signon:	
What is the IBM i user profile name for the user?	<i>JOHND</i>
What is the name of the EIM identifier that you want to create?	<i>John Day</i>
What kinds of associations do you want to create?	<b>Source association:</b> Kerberos principal <i>jday</i> <b>Target association:</b> IBM i user profile <i>JOHND</i>
What is the name of the user registry that contains the Kerberos principal for which you are creating the source association?	<i>MYCO.COM</i>
What is the name of the user registry that contains the IBM i user profile for which you are creating the target association?	<i>SystemiA.MYCO.COM</i>

## Step 2: Create a basic single signon configuration for Systemi A

You need to create a basic single signon configuration using the System i Navigator. The EIM configuration wizard will assist in the configuration process. Use the information from your planning work sheets to configure EIM and network authentication service on *Systemi A*.

**Note:** For more information about EIM, see the EIM concepts topic.

1. Start System i Navigator.
2. Expand **Systemi A > Network > Enterprise Identity Mapping**.
3. Right-click **Configuration** and select **Configure** to start the EIM Configuration wizard.
4. On the **Welcome** page, select **Create and join a new domain**. Click **Next**.
5. On the **Specify EIM Domain Location** page, select **On the local Directory server**.
6. Click **Next** and the **Network Authentication Service** wizard is displayed.

**Note:** The Network Authentication Service wizard only displays when the system determines that you need to enter additional information to configure network authentication service for the single signon implementation.

7. Complete these tasks to configure network authentication service:
  - a. On the **Configure Network Authentication Service** page, select **Yes**.

**Note:** This launches the Network Authentication Service wizard. With this wizard, you can configure several IBM i interfaces and services to participate in the Kerberos realm.

- b. On the **Specify Realm Information** page, enter *MYCO.COM* in the **Default realm** field and select **Microsoft Active Directory is used for Kerberos authentication**. Click **Next**.

- c. On the **Specify KDC Information** page, enter *kdc1.myco.com* in the **KDC** field and enter *88* in the **Port** field. Click **Next**.
- d. On the **Specify Password Server Information** page, select **Yes**. Enter *kdc1.myco.com* in the **Password server** field and *464* in the **Port** field. Click **Next**.
- e. On the **Select Keytab Entries** page, select **IBM i Kerberos Authentication**. Click **Next**.
- f. On the **Create OS/400 Keytab Entry** page, enter and confirm a password, and click **Next**. For example, *Systemi A123*. This password will be used when *Systemi A* is added to the Kerberos server.

**Note:** Any and all passwords specified in this scenario are for example purposes only. To prevent a compromise to your system or network security, never use these passwords as part of your own configuration

- g. On the **Create Batch File** page, select **Yes**, specify the following information, and click **Next**:
  - **Batch file:** Add the text *Systemi A* to the end of the default batch file name. For example, *C:\Documents and Settings\All Users\Documents\IBM\Client Access\NASConfigSeries A.bat*.
  - **Select Include password:** This ensures that all passwords associated with the IBM i service principal are included in the batch file. It is important to note that passwords are displayed in clear text and can be read by anyone with read access to the batch file. Therefore, it is recommended that you delete the batch file from the Kerberos server and from your PC immediately after use.

**Note:** If you do not include the password, you will be prompted for the password when the batch file is run.

**Note:** You must have **ktpass** and **SETSPN** (set service principal name) installed on your Windows 2000 server before running this bat file. The **ktpass** tool is provided in the Service Tools folder on the Windows 2000 Server installation CD. The **SETSPN** tool is included in the Microsoft Windows 2000 Resource Kit and can be downloaded from the Microsoft website.

- h. On the **Summary** page, review the network authentication service configuration details. Click **Finish** to complete the Network Authentication Service wizard and return to the EIM Configuration wizard.

8. On the **Configure Directory Server** page, enter the following information, and click **Next**:

**Note:** If you configured the directory server before you started this scenario, you will see the **Specify User for Connection** page instead of the **Configure Directory Server** page. In that case, you must specify the distinguished name and password for the LDAP administrator.

- Port: *389*
- Distinguished name: *cn=administrator*
- Password: *mycopwd*

**Note:** Any and all passwords specified in this scenario are for example purposes only. To prevent a compromise to your system or network security, never use these passwords as part of your own configuration.

9. On the **Specify Domain** page, enter the name of the domain in the **Domain** field, and click **Next**. For example, *MyCoEimDomain*.
10. On the **Specify Parent DN for Domain** page, select **No**, and click **Next**.

**Note:** If the directory server is active, a message is displayed that indicates you need to end and restart the directory server for the changes to take effect. Click **Yes** to restart the directory server.

11. On the **Registry Information** page, select **Local OS/400 and Kerberos**, and click **Next**.

**Note:**

- Registry names must be unique to the domain.
- You can enter a specific registry definition name for the user registry if you want to use a specific registry definition naming plan. However, for this scenario you can accept the default values.

12. On the **Specify EIM System User** page, select the user for the operating system to use when performing EIM operations on behalf of operating system functions, and click **Next**:

**Note:** Because you did not configure the directory server prior to performing the steps in this scenario, the only distinguished name (DN) that you can choose is the LDAP administrator's DN.

- User type: *Distinguished name and password*
- Distinguished name: `cn=administrator`
- Password: *mycopwd*

**Note:** Any and all passwords specified in this scenario are for example purposes only. To prevent a compromise to your system or network security, never use these passwords as part of your own configuration.

13. On the **Summary** page, confirm the EIM configuration information. Click **Finish**.

### Step 3: Add principal names to the KDC

To add the system to the Windows 2000 KDC, use the documentation for your KDC that describes the process of adding principals. By convention, the IBM i system name can be used as the username. Add the following principal names to the KDC:

```
krbsvr400/Systemia.ordept.myco.com@ORDEPT.MYCO.COM  
HTTP/Systemia.myco.com@MYCO.COM
```

On a Windows 2000 server, follow these steps:

1. Use the Active Directory Management tool to create a user account for the IBM i system (select the **Users** folder, right-click, select **New**, then select **User**.) Specify *Systemia* as the Active Directory user and *HTTPSystemia* as the service principal for HTTP.
2. Access the properties on the Active Directory user *Systemia* and the service principal *HTTPSystemia*. From the **Account** tab, select the **Account is trusted for delegation**. This will allow the *HTTPSystemia* service principal to access other services on behalf of a signed-in user.
3. Map the user account to the principal by using the **ktpass** command. This needs to be done twice, once for *Systemia* and once for *HTTPSystemia*. The **ktpass** tool is provided in the Service Tools folder on the Windows 2000 Server installation CD. To map the user account, open the **ktpass** command window and enter the following:

```
ktpass -princ krbsvr400/Systemia.ordept.myco.com@ORDEPT.MYCO.COM -mapuser Systemia -pass Systemia123
```

Then add the HTTP Server to the KDC:

```
ktpass -princ HTTP/Systemia.myco.com@MYCO.COM -mapuser Systemia -pass Systemia123
```

For HTTP, an additional step (**setspn** - set service principal name) is required after the **ktpass** is done:

```
SETSPN -A HTTP/Systemia.myco.com@MYCO.COM HTTPSystemia
```

**Note:** The **SETSPN** tool is included in the Microsoft Windows 2000 Resource Kit and can be downloaded from the Microsoft website.

**Note:** The value *Systemia123* is the password that you specified when you configured network authentication service. Any and all passwords used within this scenario are for example purposes only. Do not use the passwords during an actual configuration.

## Step 4: Add Kerberos keytab

You need keytab entries for authentication purposes as well as for generating the authorization identity. The network authentication service (the IBM i implementation of the Kerberos protocol) wizard creates a keytab entry for *SystemiA*, however a keytab for HTTP must be manually created. The wizard is only able to create keytab entries for the system and certain applications that the code is aware are Kerberos-enabled. The network authentication service wizard configures network authentication service (Kerberos) for you. The wizard is called by the EIM wizard if you have not already configure network authentication service on the system or if your network authentication service configuration is not complete.

The **kinit** command is used to initiate Kerberos authentication. A Kerberos ticket-granting ticket (TGT) is obtained and cached for the HTTP Server principal. Use **kinit** to perform the ticket exchange for the HTTP Server principal. The ticket is cached for reuse.

1. Start a 5250 session on *Systemi A*.
2. Type QSH.
3. Type `keytab add HTTP/Systemia.myco.com`.
4. Type *Systemi123* for the password.
5. Type *Systemi123* again to confirm the password.
6. Type `keytab list`.

**Note:** The **keytab list** command lists the keytab information on your IBM i system.

7. Now test the password entered in the keytab to make sure it matches the password used for this service principal on the KDC. Do this with the following command: `kinit -k HTTP/Systemia.myco.com`  
The `-k` option tells the `kinit` command not to prompt for a password; only use the password that is in the keytab. If the `kinit` command fails, it is likely that different passwords were used on either the `ktpass` command done on the Windows Domain controller or on the keytab command entered in QSH.
8. Now test the IBM i Kerberos authentication to make sure the keytab password is the same as the password stored in the KDC. Do this with the following command: `kinit -k krbsvr400/Systemia.myco.com`

**Note:** The Network Authentication Service wizard created this keytab entry.

9. Type `klist`.

**Note:** If the `kinit` command returns without errors, then `klist` will show your ticket cache.

## Step 5: Create home directory for John Day on Systemi A

You need to create a directory in the `/home` directory to store your Kerberos credentials cache. To create a home directory, complete the following:

1. Start a 5250 session on *Systemi A*.
2. Type QSH.
3. On a command line, enter: `CRTDIR '/home/user profile'` where *user profile* is your IBM i user profile name. For example: `CRTDIR '/home/JOHND'`.

## Step 6: Test network authentication service configuration on Systemi A

Now that you have completed the network authentication service configuration tasks for *Systemi A*, you need to test that your configuration. You can do this by requesting a ticket-granting ticket for the HTTP principal name, *HTTP/Systemia.myco.com*.

To test the network authentication service configuration, complete these steps:

**Note:** Ensure that you have created a home directory for your IBM i user profile before performing this procedure.

1. On a command line, enter QSH to start the Qshell Interpreter.

2. Enter `keytab list` to display a list of principals registered in the keytab file. In this scenario, `HTTP/Systemia.myco.com@MYCO.COM` displays as the principal name for *Systemi A*.
3. Enter `kinit -k HTTP/Systemia.myco.com@MYCO.COM`. If this is successful, then the `kinit` command is displayed without errors.
4. Enter `klist` to verify that the default principal is `HTTP/Systemia.myco.com@MYCO.COM`.

### Step 7: Create EIM identifier for *John Day*

Now that you have performed the initial steps to create a basic single signon configuration, you can begin to add information to this configuration to complete your single signon test environment. You need to create the EIM identifier that you specified in “Step 1: Planning work sheet” on page 80. In this scenario, this EIM identifier is a name that uniquely identifies *John Day* in the enterprise.

To create an EIM identifier, follow these steps:

1. Start System i Navigator.
2. Expand **Systemi A > Network > Enterprise Identity Mapping > Domain Management > MyCoEimDomain**

**Note:** If the domain is not listed under Domain Management, you may need to add the domain. You may be prompted to connect to the domain controller. In that case, the **Connect to EIM Domain Controller** dialog is displayed. You must connect to the domain before you can perform actions in it. To connect to the domain controller, provide the following information and click **OK**:

- **User type:** Distinguished name
- **Distinguished name:** `cn=administrator`
- **Password:** `mycopwd`

**Note:** Any and all passwords specified in this scenario are for example purposes only. To prevent a compromise to your system or network security, never use these passwords as part of your own configuration.

3. Right-click **Identifiers** and select **New Identifier...**
4. On the **New EIM Identifier** dialog, enter a name for the new identifier in the **Identifier** field, and click **OK**. For example, *John Day*.

### Step 8: Create a source association and target association for the new EIM identifier

You must create the appropriate associations between the EIM identifier and the user identities that the person represented by the identifier uses. These identifier associations, when properly configured, enable the user to participate in a single signon environment.

In this scenario, you need to create two identifier associations for the *John Day* identifier:

- A source association for the *jday* Kerberos principal, which is the user identity that *John Day*, the person, uses to log in to Windows and the network. The source association allows the Kerberos principal to be mapped to another user identity as defined in a corresponding target association.
- A target association for the *JOHND* IBM i user profile, which is the user identity that *John Day*, the person, uses to log in to System i Navigator and other IBM i applications on *Systemi A*. The target association specifies that a mapping lookup operation can map to this user identity from another one as defined in a source association for the same identifier.

Now that you have created the *John Day* identifier, you need to create both a source association and a target association for it.

To create a source association between the Kerberos principal *jday* identifier, follow these steps:

1. Start System i Navigator.

2. Expand **Systemi A > Enterprise Identity Mapping > Domain Management > MyCoEimDomain > Identifiers**
3. Right-click *John Day*, and select **Properties**.
4. On the **Associations** page, click **Add**.
5. In the **Add Association** dialog, specify or click **Browse...** to select the following information, and click **OK**:
  - **Registry:** *MYCO.COM*
  - **User:** *jday*
  - **Association type:** Source
6. Click **OK** to close the **Add Association** dialog.  
To create a target association between the IBM i user profile and the *John Day* identifier, follow these steps:
7. On the **Associations** page, click **Add**.
8. On the **Add Association** dialog, specify or **Browse...** to select the following information, and click **OK**:
  - **Registry:** *SystemiA.MYCO.COM*
  - **User:** *JOHND*
  - **Association type:** Target
9. Click **OK** to close the **Add Association** dialog.
10. Click **OK** to close the **Properties** dialog.

## Step 9: Configure IBM i Access for Windows applications to use Kerberos authentication

You must use Kerberos to authenticate before you can use System i Navigator to access *Systemi A*. Therefore, from your PC, you need to configure IBM i Access for Windows to use Kerberos authentication. Jay Day will use IBM i Access for Windows to monitor the status of the HTTP Server and monitor the other activities on the IBM i system.

To configure IBM i Access for Windows applications to use Kerberos authentication, complete the following steps:

1. Log on to the Windows 2000 domain by logging on to your PC.
2. In System i Navigator on your PC, right-click *Systemi A* and select **Properties**.
3. On the **Connection** page, select **Use Kerberos principal name, no prompting**. This allows IBM i Access for Windows connections to use the Kerberos principal name and password for authentication.
4. A message is displayed that indicates you need to close and restart all applications that are currently running for the changes to the connection settings to take effect. Click **OK**. Then, end and restart System i Navigator.

## Step 10: Add *Systemi A* to an existing EIM domain:

The IBM i does not require mapping, per the EIM configuration, as it is not a signon-type entity. You do, however, have to add the system to an existing EIM domain.

**Note:** IF EIM resides on the same IBM i system as the HTTP Server, then skip this step.

1. Start System i Navigator.
2. Expand **Systemi A > Enterprise Identity Mapping > Configuration**.
3. Click **Configure system for EIM**.
4. Click **Join an existing domain**. Click **Next**.
5. Type *Systemia.myco.com* in the **Domain controller name** field.
6. Type 389 in the **Port** field. Click **Next**.
7. Select **Distinguished name and password** from the **User type** field.

8. Type *cn=administrator* in the **Distinguished name** field.
9. Type *mycopwd* in the **Password** field.
10. Type *mycopwd* in the **Confirm password** field. Click **Next**.
11. Select *MyCoEimDomain* from the **Domain** column. Click **Next**.
12. Select *Systemia.myco.com* for **Local OS/400** and *kdc1.myco.com* for **Kerberos**.
13. Select **Kerberos user identities are case sensitive**. Click **Next**.
14. Select **Distinguished name and password** from the **User type** list.
15. Type *cn=administrator* in the **Distinguished name** field.
16. Type *mycopwd* in the **Password** field.
17. Type *mycopwd* in the **Confirm password** field. Click **Next**.
18. Review the information and click **Finish**.

### Step 11: Configure HTTP Server for single signon

After the basic test environment is working, John Day configures the HTTP Server to participate in the single signon environment. Once single signon is enabled, John Day can access the HTTP Server without being prompted for a user ID and password after signing on to the Windows environment

To set up Kerberos for your HTTP Server, complete the following steps:

1. Start the Web Administration for i interface.
2. Click the **Manage** tab.
3. Click the **HTTP Servers** subtab.
4. Select the HTTP Server you want to work with from the **Server** list.
5. Select the resource from the server area (a directory or a file) you want to work with from the **Server area** list.
6. Expand **Server Properties**.
7. Click **Security**.
8. Click the **Authentication** tab.
9. Select **Kerberos** under **User authentication method**.
10. Select **enable** or **disable** to match the source user identity (user ID) associated with the server ticket with an IBM i system profile defined in a target association. If enabled when Kerberos is specified for the AuthType directive, the server will use EIM to attempt to match the user ID associated with the server ticket with an IBM i system profile. If there is no appropriate target association for an IBM i system profile, the HTTP request will fail.
11. Click **Apply**.

Restart the HTTP Server instance to use your new Kerberos settings.

Your configuration file will now include new code for the Kerberos options you selected.

**Note:** These examples are used as reference only. Your configuration file may differ from what is shown.

Processing requests using client's authority is **Disable**:

```
<Directory />
  Order Deny,Allow
  Deny From all
  Require valid-user
  PasswdFile %%KERBEROS%%
  AuthType Kerberos
</Directory>
```

Processing requests using client's authority is **Enabled**:

```

<Directory />
  Order Deny,Allow
  Deny From all
  Require valid-user
  PasswdFile %%KERBEROS%%
  UserID %%CLIENT%%
  AuthType Kerberos
</Directory>

```

**Note:** If your Directory or File server area does not contain any control access restrictions, perform the following steps:

1. Start the Web Administration for i interface.
2. Click the **Manage** tab.
3. Click the **HTTP Servers** subtab.
4. Select your HTTP Server from the **Server** list.
5. Select the server area you want to work with from the **Server area** list.
6. Expand **Server Properties**.
7. Click **Security**.
8. Click the **Control Access** tab.
9. Select **Deny then allow** from the **Order for evaluating access** list.
10. Select **Deny access to all, except the following**.
11. Click **Add** under the **Specific allowed client hosts** table.
12. Type *\*.jkl.com* under the **Domain name or IP address** column to allow clients in the JKL domain to access the resource.

**Note:** You should type the domain name or IP address of your server. If you do not, no client is allowed access to the resources.

13. Click **Continue**.
14. Click **OK**.

### **Step 12: (Optional) Post configuration considerations:**

Now that you finished this scenario, the only EIM user you have defined that EIM can use is the Distinguished Name (DN) for the LDAP administrator. The LDAP administrator DN that you specified for the system user on *Systemi A* has a high level of authority to all data on the directory server. Therefore, you might consider creating one or more DN's as additional users that have more appropriate and limited access control for EIM data. The number of additional EIM users that you define depends on your security policy's emphasis on the separation of security duties and responsibilities. Typically, you might create at least the two following types of DN's:

- A user that has EIM administrator access control  
This EIM administrator DN provides the appropriate level of authority for an administrator who is responsible for managing the EIM domain. This EIM administrator DN could be used to connect to the domain controller when managing all aspects of the EIM domain by means of System i Navigator.
- At least one user that has all of the following access controls:
  - Identifier administrator
  - Registry administrator
  - EIM mapping operations
 This user provides the appropriate level of access control required for the system user that performs EIM operations on behalf of the operating system.

**Note:** To use the new DN for the system user instead of the LDAP administrator DN, you must change the EIM configuration properties for the system user on each system.



To use Microsoft Internet Explorer to access a Kerberos protected resource, the Integrated Windows Authentication option must be enabled. To enable it, from Internet Explorer go to **Tools > Internet options > Advanced tab and Enable Integrated Windows Authentication**.

## JKL Toy Company monitors Web server activity with logs on HTTP Server

This scenario discusses how to monitor IBM HTTP Server for i Web server activity with logs.

### Scenario

The JKL Toy Company (a fictitious company) wants to know who is visiting their Web site. The **JKLTEST** server is already using a combined access log, but the JKL Web administrator wants to create a new access log that can be altered without affecting the data in the default access log file. By using this method, the JKL Web administrator will have two logs that can be formatted to log specific information.

The JKL Web administrator found that enabling the logging function has some advantages and some disadvantages. Enabling the logging function does cause a small performance hit on the server, but a wide range of information about who is visiting the Web site can be obtained. After reading the information on log formats, the JKL Web administrator has decided to use the Combined, or NCSA Extended, log format.

See Module `mod_log_config` for HTTP Server for advanced information.

### Prerequisites

- It is assumed you have read “Scenarios: HTTP Server” on page 49.
- It is assumed you have read and completed “JKL Toy Company creates an HTTP Server” on page 50 or you have an existing HTTP Server configuration.

### Start the IBM Web Administration for i interface

Access the IBM Web Administration for i from your browser. For information about how to access the Web Administration for i interface, see “Starting Web Administration for i” on page 7.

### Set up a log file

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server instance from the **Server** list.  
Example: JKLTEST
4. Select **Global configuration** from the **Server area** list.
5. Expand **Server Properties**.
6. Click **Logging**.
7. Click the **Custom Logs** tab in the form.
8. Click **Add** under the **Custom logs** table.
9. Enter a name for the new log in the **Log** column.  
Example: `logs/server_monitor`

**Note:** The above example creates a log file named `server_monitor` in the `/logs` directory.

10. Select **combined** from the **Log format** list in the **Attributes** column.
11. **Optional:** Accept the default Environment variable condition or enter a new value.
12. **Optional:** Accept the default expiration of the log or enter a new value.
13. **Optional:** Accept the default maximum cumulative seize or enter a new value.

14. Click **Continue**.
15. **Optional:** Click **Log identity of client**. This may significantly degrade performance of the web server. under **Client identity logging**.

**Note:** The option to **Log identity of client** will impact server performance by requiring a Domain Name Server (DNS) lookup every time a new client is logged. If you do not log the identity of the client the IP address of the client will be logged instead of the domain name. Some log analysis tools can perform DNS lookup, allowing identity of clients without impacting your performance.

16. Click **OK**.

## Restart your HTTP Server

Select one of the following methods below:


### Manage one server

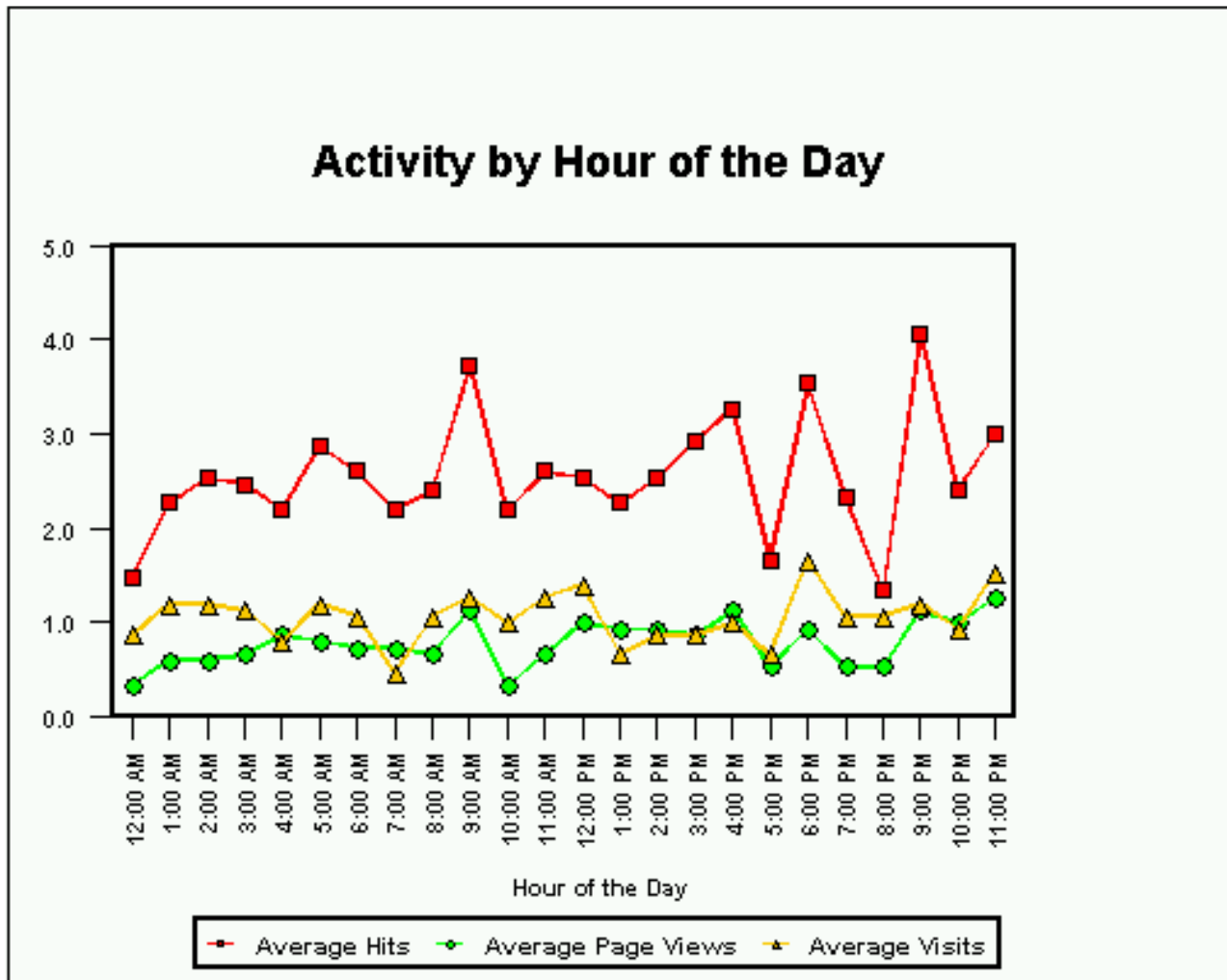
1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the Server list.
4. Click the **Stop** icon if the server is running.
5. Click the **Start** icon.

### Manage all servers

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select **All Servers** from the Server list.
4. Click the **All HTTP Servers** tab.
5. Select your HTTP Server name in the table.  
Example: JKLTEST
6. Click **Stop** if the server is running.
7. Click **Start**.

**Note:** If your HTTP Server does not start, see “Troubleshooting” on page 202.

Logging will begin when the HTTP Server instance has started. The JKL Web administrator has decided to use the IBM Tivoli Web Response Monitor  to generate usage reports. This product can read the log file and generate detailed reports that contain information such as the following:



### Test your HTTP Server

1. Open a new Web browser.
2. Enter `http://[i_hostname]:[port]` in the location or URL field.  
Example: `http://jkl_server:1975`

Review your log for HTTP Server activity.

### View your HTTP Server configuration

Your configuration will look similar if you used the given example in this and previous examples.

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.  
Example: JKLTEST
4. Expand **Tools**.
5. Click **Display Configuration File**.

```
Listen *:1975
DocumentRoot /www/jkltest/htdocs
ServerRoot /www/jkltest
Options -ExecCGI -FollowSymLinks -SymLinksIfOwnerMatch -Includes -IncludesNoExec -Indexes
LogFormat "%h %l %u %t \"%r\" %s %b \"%{Referer}i\" \"%{User-Agent}i\" combined
```

```

LogFormat "%{Cookie}n \"%r\" %t" cookie
LogFormat "%{User-agent}i" agent
LogFormat "%{Referer}i -> %U" referer
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log combined
CustomLog logs/server_monitor combined
SetEnvIf "User-Agent" "Mozilla/2" nokeepalive
SetEnvIf "User-Agent" "JDK/1\." force-response-1.0
SetEnvIf "User-Agent" "Java/1\." force-response-1.0
SetEnvIf "User-Agent" "RealPlayer 4\." force-response-1.0
SetEnvIf "User-Agent" "MSIE 4\." nokeepalive
SetEnvIf "User-Agent" "MSIE 4\." force-response-1.0
<Directory />
  Order Deny,Allow
  Deny From all
</Directory>
<Directory /www/jkltest/htdocs>
  Order Allow,Deny
  Allow From all
</Directory>

```

---

## Tasks

This topic provides step-by-step instructions for administration and management tasks with the IBM Web Administration for i interface.

### Getting started with the IBM Web Administration for i interface

The IBM Web Administration for i interface is used to create and configure IBM HTTP Server for i Web servers.

#### Step 1: Install

Ensure that IBM HTTP Server for i is installed on your server and is functioning correctly. For more information on installing the product, see "Installing HTTP Server" on page 2.

#### Step 2: Create an HTTP Server instance

Use the **Create HTTP Server wizard** to quickly create a working HTTP Server configuration.

1. Access the IBM Web Administration for i from your browser. For information about how to access the Web Administration for i interface, see "Starting Web Administration for i" on page 7.
2. Click the **Setup** tab.
3. Expand **Common Tasks and Wizards**.

**Note:** By default, all lists are expanded. If you collapse any list, the Web Administration for i interface displays the list as collapsed the next time you view it.

4. Click **Create HTTP Server**.
5. Enter a name to identify your HTTP Server. This name is used later to configure and administer your server. Enter a server description to help identify your server.
6. Click **Next**.
7. Enter the server root. The server root is the base directory for your HTTP Server. Within this directory, the wizard creates subdirectories for your logs, and configuration information. If the server root does not exist, the **Create HTTP Server wizard** creates one for you.
8. Click **Next**.
9. Enter the document root. The document root is the directory from which your documents are served by your HTTP Server. If the directory root does not exist, the **Create HTTP Server wizard** creates one for you.

10. Click **Next**.
11. Leave the IP address list as **All addresses**. You may select a specific IP address if you so choose.
12. Enter a port number. By default, the port is 80. This is the port your Web site runs (or "listens on"). It is suggested you enter a different port other than 80 because a port can only be used by one server at any time.
13. Click **Next**.
14. Select **Yes** or **No** for the **Create HTTP Server wizards** to create an access log. The access log contains information about requests made to your HTTP Server. This information is useful for analyzing who is accessing your Web site and how many requests have been made during a specific period of time.
15. Click **Next**.
16. Specify how long you want to keep the error and access log files. Select **Keep, do not delete** or **Delete based upon age**.
17. Click **Next**.
18. The **Create HTTP Server wizard** displays a summary of HTTP Server configuration it creates. If you want to change an entry, simply click **Back**.
19. Click **Finish** and HTTP Server is created.

For more information on the Web Administration for i interface, see "Overview of IBM Web Administration for i" on page 4.

### Step 3: Start and test your HTTP Server

After using the **Create HTTP Server wizard**, it is time to start your Web server and go live.

1. Click the **Start** icon next to the **Server list**.
2. Click the Refresh icon and check if the server status is still shown as "Running".  
If your HTTP Server does not start, see "Troubleshooting" on page 202.
3. Open another Web browser and go to `http://your.server.name:port/` where *your.server.name* is the host name of your IBM i server and *port* is the port number you entered in the **Create HTTP Server wizard**.

The supplied HTML example welcome page is displayed.

When you have finished this preliminary work with the Web Administration for i interface, expand your HTTP server capabilities. See the "Scenarios: HTTP Server" on page 49 for more information.

## HTTP Server tasks

This topic provides step-by-step tasks for an IBM HTTP Server for i Web server.

### Related information:

"HTTP Server Concepts" on page 13

This topic provides conceptual information of the various functions and features of IBM HTTP Server for i.

## Setting up MIME types on HTTP Server

Set up MIME types for your IBM HTTP Server for i instance using the IBM Web Administration for i interface.

Multipurpose Internet Mail Extensions (MIME) types associate file contents and file extensions with the way the server and the client handle files. To change the MIME settings for the server, do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server list**.

4. Select the context you want to work with from the **Server area** list.
5. Expand **Server Properties**.
6. Click **Content Settings**.
7. Click the **MIME** tab in the form.
8. Edit the default content-type, content-language, and character set values as necessary.
9. **Optional:** If necessary, select **File extensions are case sensitive** to distinguish between uppercase and lowercase letters when comparing file extensions.
10. **Optional:** If necessary, select **Force content-type for all files** to force the mapping of all files in this context to a specified MIME type.
11. Click **Add** under the **Specify individual Meta (MIME) information for file extensions** table.
12. Enter file extensions in the **File extension** column.
13. If available, select **Add** from the list in the **Action** column.
14. Select the file type from the list in the **Type** column.
15. Enter or select additional MIME types, encoding, languages, or browser types in the **Value** column.
16. Click **Continue**.
17. Click **OK**.

### Setting up content and language negotiation for HTTP Server

Content negotiation for an HTTP Server instance can be set up using the IBM Web Administration for i interface. Content negotiation is defined as the process where the client provides a set of preferences (such as language) to the server, and the server finds the best resource match to those the client prefers.

To configure content and language negotiation, do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Select the context you want to work with from the **Server area** list.
5. Expand **Server Properties**.
6. Click **Content Settings**.
7. Click the **Content Negotiation** tab in the form.
8. **Optional:** If necessary, select **Allow content-negotiated documents to be cached**.
9. Click **Add** under the **Language priority (highest to lowest priority)** table.
10. Enter or select from the list a content-language in the **Content-language** column.
11. Click **Continue**.
12. **Optional:** If necessary, select language priority to force from the **Force language priority** list.
13. Click **OK**.

#### Related information:

“Content negotiation for HTTP Server” on page 17

The IBM HTTP Server for i supports content negotiation, type-map files, MultiViews, negotiation methods, dimensions of negotiation,, negotiation algorithm, media types, and wildcards.

### Setting up customized error messages on HTTP Server

Customized error messages for an HTTP Server instance can be set up using the IBM Web Administration for i interface.

The server has default messages that are displayed to the user when an error occurs. You can change these messages to better suit your particular needs. For example, you can change a message to include more information about the cause of the problem and suggest possible solutions for it. For internal networks, you might provide a contact person for your users to call.

To customize your messages, do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Select the context you want to work with from the **Server area** list.
5. Expand **Server Properties**.
6. Click **HTTP Responses**.
7. Click the **Error Message Customization** tab in the form.
8. Select how you want to append the generated footer onto error messages. Depending on the server area you select, you may, optionally, select **Inherit**.
9. Select an error code from the **Custom messages from error codes** table or click **Add** to add a new error message.
10. Click **OK**.

If your messages are displayed in the Microsoft Internet Explorer browser, see “Symptom: Web browser problems with HTTP Server” on page 208.

## Setting up directory indexing and directory listing on HTTP Server

Directory index and directory listing for an IBM HTTP Server for i instance can be set up using the IBM Web Administration for i interface.

A directory index or directory listing shows files and subdirectories that are contained in the directory. The server shows each subdirectory item or each file on a separate line along with information about each item. Use caution when configuring directory listing function, since it allows others to view your directory structure.

To enable directory listings do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Select the context you want to work with from the **Server area** list.
5. Expand **Server Properties**.
6. Click **Directory Handling**.
7. Click the **General Settings** tab in the form.
8. Select **Enabled** for your HTTP Server to always search for a welcome or index file name.
9. Select **Display directory listings for all directories**.
10. Click **Apply**.

Once directory listings are enabled, you can customize the appearance of the directory list (also called fancy indexing). Directory listing is optional.

To customize the appearance of your directory list, do the following:

1. Click the **Appearance** tab in the form.
2. Select the options for your directory listing. View the help text for specific field values.
3. Click **OK**.

## Setting up environment variables on HTTP Server

Set up environment variables for CGI programs running in an HTTP Server instance using the IBM Web Administration for i interface.

When the server runs a CGI program, it uses environment variables to pass information about the request and the server. Configuring environment variables allows you to specify which variables the CGI programs inherits.

To specify environment variables, do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Select the context you want to work with from the **Server area** list.
5. Expand **Server Properties**.
6. Click **Request Processing**.
7. Click the **Custom Environment Variables** tab in the form.
8. Click **Add** under the **Environment variables based on a conditional attribute** table.

**Note:** Select an environment variable from the table to redefine or remove an existing environment variable.

9. Enter the environment variable name in the **Variable** column.
10. Enter the environment variable value in the **Value** column.
11. Enter the environment variable attribute in the **Attribute** column.
12. Enter the environment variable attribute value in the **Attribute value** column.
13. **Optional:** Select to make the environment variable case sensitive in the **Case sensitive** column.
14. Click **Continue**.
15. Click **OK**.

See “Environment variables set by HTTP Server” on page 493 for a list of environment variables.

## Setting up of a highly available HTTP Server

Set up and administer highly available IBM HTTP Server for i instances using the IBM Web Administration for i interface.

All required programs (HTTP Server, WebSphere, Servlets, Net.Data, and Clustering support) must already be installed on all nodes. See “Highly available HTTP Server” on page 43 for more information.

### Step 1 - Configure the IBM i Cluster

For each node, configure your cluster. See Configuring clusters for more information. Then continue to step 2.

### Step 2 - Configure IP addresses

For each IBM i node in the cluster that a highly available Web server will be running on, configure the IP address that the Web server will be using. This can be done using the **CFGTCP CL** command. You should configure one IP address for each unique Web server. Each Web server is configured to a dedicated TCP/IP line interface. When using the Network Dispatcher model or comparable IP director with either HAModel IPTakeoverWithDispatcher or PurePeer model, the IP Line interface should be typed **\*VIRTUALIP**. See TCP/IP for more information.

1. Start the Web Administration for i interface.
2. Click the **Manage** tab.
3. Click the **HTTP Servers** subtab.
4. Select your HTTP Server from the **Server** list.
5. Select the context you want to work with from the **Server area** list.



6. Expand **Server Properties**.
7. Click **General Server Configuration**.
8. Click the **General Settings** tab in the form.
9. Click **Add** under the **Server IP addresses and ports to listen** on table.

**Note:** Directive HotBackup will be set to *off* and ignored if currently configured for your HTTP Server.

You may want to perform the next steps on one IBM i server and copy (for example using FTP or NetServer) the HTTP Server configuration and instance files to each IBM i server where the highly available HTTP Server will be running in the cluster. The files that must be copied are:

- /www/server\_name/conf/server\_name.conf
- /QSYS.LIB/QUSRSYS.LIB/QATMINSTC.FILE/instance\_name.MBR

10. Add the IP address the highly available Web server will be running on.
11. Click **OK**.
12. Continue to step 3.

### **Step 3- Configure the highly available HTTP Server**

1. Start the Web Administration for i interface.
2. Click the **Manage** tab.
3. Click the **HTTP Servers** subtab.
4. Select your HTTP Server from the **Server** list.
5. Select the context you want to work with from the **Server area** list.
6. Expand **Server Properties**.
7. Click **System Resources**.
8. Click the **Highly Available Server** tab in the form.
9. Specify one specific server IP address to listen on.
10. Click **Enable HTTP server** to be highly available.
11. Select a highly available model.

**Note:** If you are implementing the primary/backup with network dispatcher model or the peer model, configure the network dispatcher according to the existing cluster nodes and the configured Web server.

12. **Optional:** Click **Enable highly available CGI program**.
13. Enter your liveness monitor settings. The LMUrlCheck directive is required. The other LM directives have defaults.
14. Click **OK**.
15. Continue to step 4.

### **Step 4- Start the highly available HTTP Server**

Start your highly available HTTP Server.

1. Start a 5250 session on the IBM i server that will contain a highly available HTTP Server instance.
2. Use **STRTCPSVR** CL command on the appropriate node.
3. Continue to step 5.

**Note:** In the case of the primary/backup model, the first highly available server to be started will automatically assume the role of the primary. The second highly available server to be started will automatically assume the role of the backup.

## Step 5- Manage your highly available HTTP Server

Use the **ENDTCPSVR** CL command on the appropriate node or use the IBM Simple Cluster Management interfaces to stop or end your highly available HTTP Server. In the case of primary/backup model depending on which server you are ending this may or may not force a fail over. Ending the primary server with a backup server running will force a fail over from primary to backup to occur. Ending the backup will only affect the backup server. Ending the primary server with no backup will end the primary server. In the case of PurePeer model only the server you are ending will be affected as any other peer servers will continue to process client requests.

**Note:** In the case of primary/backup model, it is possible to determine which highly available Web server is the primary or backup server. The QBATCH subsystem will have a job running named **QZHBEXPG** on the primary node only. For the client data it is suggested that you set up a method to automatically publish static files to each Web server. Static files include HTML and highly available CGI programs.

### Setting up a welcome or index page on HTTP Server

Set up a welcome or index page on your IBM HTTP Server for i instance using the IBM Web Administration for i interface.

You can configure your server to display a specific Web page known as a welcome page for client requests that do not include a specific file name. The server determines which file to serve by matching the list of welcome pages to the files in the directory. The first match it finds is the file it will return. To configure welcome page settings, do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Select the context you want to work with from the **Server area** list.
5. Expand **Server Properties**.
6. Click **General Server Configuration**.
7. Click the **Welcome Pages** tab in the form.
8. Select **Enabled** to have the Welcome page displayed.
9. Select the default action the server will take if the welcome file or index file does not exist.
10. Click **Add** under the **Welcome/index file names** table.

**Note:** You may also use the existing file in the Welcome/index file names table.

11. Click **Browse** and select the HTML file you want to use as a Welcome page.
12. Click **Continue**.
13. Click **OK**.

### Manually editing HTTP Server

Manually edit your IBM HTTP Server for i Web server configuration using the IBM Web Administration for i interface.

**Attention:** Improper modifications to your configuration file could make your HTTP Server unusable. Modifications to the configuration file manually should only be performed by advanced users.

The Web Administration for i interface has been designed to modify the HTTP Server configuration file by applying changes made to the various forms and wizards supplied by the Web Administration for i interface. Use of the forms and wizards greatly decreases the potential for user error and helps maintain an error-free configuration file.

Optionally, the configuration file may be edited manually. When the configuration file has been modified manually, the Web Administration for i interface does not perform the usual error checking that is done when using the Web Administration for i interface. Any changes made to the configuration file directly should be done with caution.

As a precaution, do the following before you modify the configuration file manually:

- Save a backup of your configuration file before manually editing. See “Managing backup files for HTTP Server” on page 103 for more information.
- Keep track of any changes you make to your configuration file.

In addition, after each modification, test your configuration by stopping and starting your HTTP Server. Verify the directives you manually configured have the desired effect.

To modify the HTTP Server configuration manually, do the following:

1. Start the Web Administration for i interface.
2. Click the **Manage** tab.
3. Click the **HTTP Servers** subtab.
4. Select your HTTP Server from the **Server** list.
5. Expand **Tools**.
6. Click **Edit Configuration File**.

**Note:** The line mode editor functions as a simple text editor only and does not error check any changes to the configuration file.

Click **OK** when you have finished modifying the configuration file. Stop and start the server.

## Managing HTTP Servers

Manage your IBM HTTP Server for i Web server using the IBM Web Administration for i interface.

As your client base grows and changes, and you add or move Web content, you need to redefine your list of servers. After successfully creating an HTTP Server there are a number of basic tasks that you will need to know in order to manage your servers successfully.

- “Starting and stopping the ADMIN server”
- “Checking status of a server” on page 102
- “Starting and stopping a server” on page 102
- “Renaming a server” on page 102
- “Deleting a server” on page 102

## Starting and stopping the ADMIN server

The ADMIN server runs on port 2001 (or 2010 for a secure connection) and serves the IBM i Task Page.

You can start the ADMIN server by doing one of the following:

- In System i Navigator click **Network** -> **Servers** -> **TCP/IP** and right-click **HTTP Administration**. Then click **Start Instance** -> **ADMIN**.
- On an IBM i command line type **STRTCPSVR SERVER(\*HTTP) HTTPSVR(\*ADMIN)**.

You can stop the ADMIN server by doing one of the following:

- In System i Navigator click **Network** -> **Servers** -> **TCP/IP** and right-click **HTTP Administration**. Then click **Stop Instance** -> **ADMIN**.
- On an IBM i command line type **ENDTCPSVR SERVER(\*HTTP) HTTPSVR(\*ADMIN)**.

## Checking status of a server

To determine the status of your server, do the following using the Web Administration for i interface:

1. Click the **Manage** tab.
2. Click the **All Servers** subtab.

**Note:** Items listed as "Unknown" are servers the Web master user profile does not have authority to.

## Starting and stopping a server

Select one of the following methods below using the Web Administration for i interface:

### Manage one server

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your server from the **Server** list.
4. Click the **Stop** icon if the server is running.
5. Click the **Start** icon.

### Manage all servers

1. Click the **Manage** tab.
2. Click the **All Servers** subtab.
3. Click the **All HTTP Servers** tab.
4. Select your server name in the table.
5. Click **Stop** if the server is running.
6. Click **Start**.

**Note:** When stopping or starting a server, it may take several seconds for the jobs to end or begin. Click **Refresh** to view the server's current status. If your HTTP Server does not start, see "Troubleshooting" on page 202.

## Renaming a server

To rename a server, do the following using the Web Administration for i interface:

1. Click the **Manage** tab.
2. Click the **All Servers** subtab.
3. Click the **All HTTP Servers** tab.
4. Select the server you want to rename.
5. Click **Rename**.
6. Enter the new name.
7. Click **OK**.

You will receive a message that indicates whether or not the task completed successfully.

**Note:** This does not change the document root or the server root. Only the instance name is changed.

## Deleting a server

Once you delete a server, you cannot retrieve it. You must create a server to replace the deleted server. If the server you selected is running, it stops before the system deletes it. The system does not delete the server configuration that is associated with this server or the directory and its contents.

To delete a server, do the following using the Web Administration for i interface:

1. Click the **Manage** tab.
2. Click the **All Servers** subtab.
3. Click the **All HTTP Servers** tab.
4. Select the server you want to delete.
5. Click **Stop** if the server is running.
6. Click **Delete**.

You will receive a message that indicates whether or not the task completed successfully.

## Managing addresses and ports for HTTP Server

This topic provides information about how to manage addresses and ports for your IBM HTTP Server for i with the IBM Web Administration for i interface.

Most browsers make HTTP requests on ports 80 and 443 by default. Typically, the default configuration option is for servers to listen on all IP addresses on port 80. Multiple servers cannot listen on the same port and IP numbers. Multiple servers may listen on the same IP address, but require a unique port, or they may listen on the same port, but require a unique IP address. If you want each server to listen on port 80, then you should configure each server to listen on a specific unique IP address. In addition, if you add another Web server product such as LotusDomino on the same IBM i server, it cannot listen on the same IP address and the same port as the HTTP Server.

You can change the IP address or port for your server by doing the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server name from the **Server** list.
4. Expand **Server Properties**.
5. Click **General Server Configuration**.
6. Click the **General Settings** tab in the form.
7. Do one of the following:
  - Select an existing IP address and port from the **Server IP address and port to listen on** table to modify or delete.
  - Click **Add** under the **Server IP address and ports to listen on table** to add a new IP address and port.
8. Click **Enabled** or **Disabled** in the FRCA column. Only select **Enabled** if you are using or will be using FRCA.
9. Click **Continue**.
10. Click **OK**.

## Managing backup files for HTTP Server

In the IBM HTTP Server for i, there are several files that should be backup up for later recovery.

Make sure that the following objects are included in your periodic backup activity:

### Instance files

- QUSRSYS/QATMHINSTA
- QUSRSYS/QATMHINSTC

### Configuration files

## HTTP Server

Save the *conf* file located in the */www/[server\_name]/conf/* directory, where *Server\_Name* is the name of your HTTP Server instance.

**Note:** This describes the default location of the configuration file. If your configuration files are located in another directory, you must save the configuration file in your location.

For more information about backup and recovery of files on the IBM i server, see Systems management.

## Managing directories for HTTP Server

You can manage directories for a IBM HTTP Server for i instance with the IBM Web Administration for i interface.

The following explains how to add a directory and how to remove a directory from your HTTP Server configuration.

### Add a directory

The HTTP Server uses directories to serve Web pages and content. The Web Administration for i interface has an Add a Directory to the Web wizard that will create a new directory to serve Web content and CGIs.

To add a new directory, do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Select **Global configuration** from the **Server area** list.
5. Expand **HTTP Tasks and Wizards**.
6. Click **Add a Directory to the Web**.

When the wizard is finished, it will display a summary of the directory you just created.

### Remove a directory

When removing a directory, the Web Administration for i interface removes all references to the directory from your configuration file only. The physical directory and content within the directory are not removed from the file system.

To remove a directory and subdirectories, do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Select **Global configuration** from the **Server area** list.
5. Expand **Server Properties**.
6. Click **Container Management**.
7. Click the **Directories** tab in the form.
8. Select the directory you want to delete from the **Directory/Directory Match container** table.
9. Click **Remove**.
10. Click **OK** when the message box appears.
11. Click **OK**.

## Managing HTTP Server performance

Performance in a IBM HTTP Server for i Web server environment is influenced by many components. Understanding the components can help you to react quickly when a performance problem occurs at a crucial time.

There are several things that can affect your server's performance. Consider the following performance related topics:

- "Local cache"
- "Files to cache when server has started"
- "Threads" on page 106
- "DNS lookups" on page 106
- "Server-side includes" on page 107
- "Content negotiation" on page 107
- "Document tree" on page 107
- ".htaccess files" on page 107
- "Virtual host log files" on page 107
- "KeepAlive and KeepAliveTimeout" on page 107
- "Logging" on page 107
- "CGI programs" on page 108
- "TCP/IP settings" on page 108
- "Network" on page 108

### Local cache

Enabling the HTTP Server's local cache can result in better performance and system throughput by caching (in memory) frequently accessed files. You can configure several settings associated with the local cache.

To configure the local cache settings, do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Select the **Global configuration** from the **Server area** list.
5. Expand **Server Properties**.
6. Click **System Resources**.
7. Click the **Caching** tab in the form.

Enter or select options from this form. After you are finished, click **OK**.

### Files to cache when server has started

Including file names in **Files to cache when server is started** causes the files to be loaded into the server's memory when the server is started.

- **Copy into memory** specifies the names of files that you want to load into the server's memory each time you start the server. By keeping your most frequently requested files loaded in the server's memory, you can improve your server's response time for those files. For example, if you load your server's welcome page into memory at startup, the server can handle requests for the page much more quickly than if it had to read the file from the file system.
- **Keep file descriptor open** specifies the names of ASCII stream files whose descriptors are cached at the server startup. By keeping your most frequently requested files opened at server startup, you can

improve your server's response time for those files. For example, if you open your server's welcome page files at startup, the server can handle requests for the page much more quickly than if it had to open the files each time they are requested. The advantage of using this option over Copy into memory is it does not cache the content of the file and therefore does not allocate large amount of memory, yet provides similar performance. The disadvantage of using this option over Copy into memory is it only caches the file descriptors of ASCII stream files and it keeps the file open (share read) while the server is active.

- **Memory map of file** option is the same as Copy into memory except it uses memory address pointers, instead of simply using a chunk of server memory, to specify the names of files that you want to map into the server's memory each time that you start the server.

**What to cache** allows you to specify what information is included in the cache.

- **Dynamically cache files based on file usage** allows dynamic caching. The default value is off (or disabled).
- **Update cache when files are modified** updates the cache whenever its original file content changes. The default value is on (or enabled).

Enter or select options from this form. After you are finished, click **OK**.

## Threads

Each time your server receives a client request, the server first checks to see if any threads are available and then uses available threads to process the request. If no threads are available, it holds the request until threads become available. When a request ends, the server threads become idle (at which point they are available for the server to use again).

**Note:** The HTTP Server performance may increase by increasing the number of threads, but not the IBM i system performance.

Setting the maximum number of active threads too high can cause a decrease in system performance. You can experiment with lowering the maximum number of active threads until you see no affect on system performance. A good starting point would be half of the previous setting. For example, if you had the maximum number of active threads set to 100, try setting it to 50. Lowering the maximum number of active threads directive might result in an increased number of rejected connections when the server reaches its capacity.

To change the number of threads to process requests, do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Select the **Global configuration** from the **Server area** list.
5. Expand **Server Properties**.
6. Click **System Resources**.
7. Click the **Advanced** tab in the form.

Enter or select options from this form. After you are finished, click **OK**.

## DNS lookups

Every time the server needs to request a DNS lookup, there may be a delay while the DNS server is contacted. Limit the use of DNS lookups. Consider logging IP addresses and using a log analysis tool that does DNS lookups.



## Server-side includes

Server performance can be impacted when server-side includes are processed. Limit the use of server-side includes except where needed.

## Content negotiation

Restrict content negotiation to those contexts where it is needed.

## Document tree

Try to organize your document tree into a flat broad tree structure rather than a narrow deep tree structure. The fewer directory levels the better.

For better performance, store static and Net.Data files in the root (or /) file system. Avoid placing static and Net.Data files in the QSYS and QDLS file systems.

## .htaccess files

Server performance is impacted if the server must look for and open .htaccess files. If the AllowOverride directive is set to None, the server does not look for .htaccess files. If AllowOverride is set to All, there is a significant performance impact as the server looks for .htaccess files in every directory.

## Virtual host log files

If you create separate log files for each virtual host, you should consider that a file descriptor is opened for each log file. Opening too many file descriptors can impact system performance.

## KeepAlive and KeepAliveTimeout

The connection time-out determines the number of seconds the server waits for a subsequent request before closing a persistent connection. Enabling persistent connections increases the throughput of your server. Consider decreasing the connection time-out if you have simple pages without images.

To set this value, do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Select the **Global configuration** from the **Server area** list.
5. Expand **Server Properties**.
6. Click **System Resources**.
7. Click the **HTTP Connections** tab in the form.
8. Enter a value for **Connection time-out**, or make a selection from the list.
9. Enter a value for **Maximum pending connections**, or make a selection from the list.
10. Select **Enabled** for **Allow persistent connections**.
11. Enter a value for **Time to wait between requests**, or make a selection from the list.
12. Enter a value for **Maximum requests per connection**, or make a selection from the list.
13. Click **OK**.

## Logging

Logging server activity does impact server performance. Try to do as little error and access logging as required.

## CGI programs

CGI programs should be run in a named activation group to get the best performance. Also determine what CGI jobs your server generally uses. Use the `StartCGI` and `StartThreadedCGI` directives to start those jobs when the server starts. Use the `QTMHHTTP1` user profile to run CGI requests. If you must use a different user profile, use a "dummy" user profile (a user profile that is not allowed to sign-on) instead of `%%CLIENT%%`.

## TCP/IP settings

See TCP/IP applications, protocols, and services for more information on TCP/IP settings.

## Network

Consider that the performance of the network that your data flows across can also affect the perception of your server's performance.

## Compression tasks

The IBM HTTP Server for i supports the configuration and management of compression files.

### Related information:

"File compression for HTTP Server" on page 31

Information is compressed by the HTTP Server before being sent to the client over the network.

## Setting up input decompression for HTTP Server

This topic provides information about how to set up input decompression for GZIP compressed input bodies for IBM HTTP Server for i Web server using the IBM Web Administration for i interface.

In order to set up input decompression, a filter must be inserted in the input filter chain. This is done using the `SetInputFilter` directive. WebDAV makes frequent use of compression, and as such, input decompression is used primarily with WebDAV. The range of usefulness of input decompression is also determined by Web browser support. Most Web browsers do not support compressed data or have limited support. Compressed information is only accessible with Web browsers with HTTP/1.1 support. See "File compression for HTTP Server" on page 31 for more information.

To set up input decompression, do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Select the context you want to work with from the **Server area** list.
5. Expand **Server Properties**.
6. Click **Compression**.
7. Click the **Input Filter** tab.
8. Click **Add** under the **Set input filter** table.
9. Specify DEFLATE for the filter name under the **Filter name** column.
10. Click **Apply**.

You should now have something like the below example in your configuration.

### Example

```
SetInputFilter DEFLATE
```

See "Setting up output compression for HTTP Server" on page 109 for more information.

## Setting up output compression for HTTP Server

This topic provides information about how to set up output compression for an IBM HTTP Server for i Web server using the IBM Web Administration for i interface.

In order to set up output compression, a filter must be inserted in the output filter chain. This is done using the `SetOutputFilter` directive. See “File compression for HTTP Server” on page 31 for more information.

To set up output compression, do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Select the context you want to work with from the **Server area** list.
5. Expand **Server Properties**.
6. Click **Compression**.
7. Click the **Output Filters** tab.
8. Add an output filter:  
There are two types of output filters. The first output filter type only requires a file extension and a filter name. The second output filter type requires a MIME type and filter name. For both output filter types, click **Add** under the appropriate table and specify the file extension, MIME type, and filter name.
9. Click **Continue**.
10. Click **Add** under the **Set output filter** table.
11. Specify DEFLATE for the filter name.
12. Click **Apply**.

You should now have something like the below example in your configuration.

### Example

```
AddOutputFilterByType DEFLATE text/html
AddOutputFilter DEFLATE .html
SetOutputFilter DEFLATE
```

See “Setting up input decompression for HTTP Server” on page 108 for more information.

## Fast Response Cache Accelerator tasks

The IBM HTTP Server for i supports the Fast Response Cache Accelerator (FRCA).

### Related concepts:

“Fast Response Cache Accelerator (FRCA) for HTTP Server” on page 32

The Fast Response Cache Accelerator (FRCA) improves the performance and scale of Web and TCP server applications by storing both static and dynamic content in a memory-based cache located in the Licensed Internal Code.

## Setting up Fast Response Cache Accelerator (FRCA) for HTTP Server

This topic provides information about how to set up Fast Response Cache Accelerator for your IBM HTTP Server for i Web server using the IBM Web Administration for i interface.

FRCA is a Web cache architecture that is tightly integrated with the TCP/IP stack. FRCA moves performance critical TCP Application functions into a fast response cache that improves HTTP Server performance. The following explains how to enable FRCA, FRCA logging, and FRCA file caching.

- “Enabling FRCA” on page 110
- “Enabling FRCA logging” on page 110

- “Enabling FRCA file caching” on page 111
- “Enabling FRCA reverse proxy caching” on page 111

## Enabling FRCA

**Note:** The following information may be used to enable FRCA for the first time or enable FRCA for a different Server area.

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Select the context you want to work with from the **Server area** list.
5. Expand **Server Properties**.
6. Click **FRCA**.
7. Click the **General Settings** tab in the form.
8. Click **Add** under the **Server IP addresses and ports to listen on** table.
9. Enter an IP address and port number or select an existing IP address and port. FRCA will listen on the IP address and port you specify.
10. Select **Enabled** from the list under the **FRCA** column.
11. Click **Continue**.
12. Click **Apply**.
13. Stop and restart your server.

FRCA is now enabled. After enabling FRCA, you can set up logs and file caching.

## Enabling FRCA logging

FRCA logging information allows you to track and generate reports on your HTTP Server's activity. You may specify various log attributes, such as the format for the information in the log file, rules for excluding entries from the log file, and client side information logging. Each server configuration file contains information about the type of log files the server will create. You must enable FRCA before FRCA logging can be set up. See Set up logs on HTTP Server for more information.

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Select the context you want to work with from the **Server area** list.
5. Expand **Server Properties**.
6. Click **FRCA**.
7. Click the **FRCA Logs** tab in the form.
8. Click **Add** under the **FRCA logs** table.
9. Enter the name of the log file you want to use.
10. Enter the log attributes under the **Attributes** column.
11. Click **Continue**.
12. Click **OK**.
13. Stop and restart your server.

## Enabling FRCA file caching

FRCA provides file caching support. You may specify the maximum cache size, the maximum file size to cache, the files to cache during server startup, and the directories to dynamically cache files from. You must enable FRCA before FRCA file caching can be set up.

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Select the context you want to work with from the **Server area** list.
5. Expand **Server Properties**.
6. Click **FRCA**.
7. Click the **FRCA File Cache** tab in the form.
8. Select **Enabled** from the **FRCA file cache capabilities** list.
9. Enter a new value for **Maximum cache size** and select corresponding size unit, or keep the default value.
10. Enter a new value for **Maximum file size to cache** and select the corresponding size unit, or keep the default value.
11. Click **Add** under the **Files to cache during server startup** table to add file types or specific files to cache at HTTP Server startup.
12. Click **Continue** when finished adding files to table.
13. Click **Add** under the **Files to cache during server runtime** table to add file types or specific files to cache during HTTP Server runtime.
14. Click **Continue** when finished adding files to table.
15. Click **OK**.
16. Stop and restart your server.

## Enabling FRCA reverse proxy caching

FRCA provides reverse proxy caching support. You may specify the maximum proxy cache size and the maximum proxy response size to cache. In addition, you may provide options for controlling which documents are cached based on expiration criteria, specify remote servers for proxy requests, and establish document retention policies. You must enable FRCA before FRCA Reverse Proxy caching can be set up.

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Select the context you want to work with from the **Server area** list.
5. Expand **Server Properties**.
6. Click **FRCA**.
7. Click the **FRCA Reverse Proxy Cache** tab in the form.
8. Select **Enabled** from the **FRCA reverse proxy cache capabilities** list.
9. Enter a new value for **Maximum proxy cache size** and select its corresponding size unit, or keep the default value.
10. Enter a new value for **Maximum proxy response size to cache** and select its corresponding size unit, or keep the default value.
11. Enter a new value for **Document retention period** and select its time unit, or keep the default value.
12. Click **Add** under the **Proxy requests to remote servers** table.
13. Enter a virtual path under the **Local virtual path** column.

14. Enter a remote server URL under the **Remote server URL** column.
15. Click **Continue**.
16. Click **Add** under the **Document refresh policies** table.
17. Enter a full or partial URL under the **Match URL** column.
18. Enter a value under the **Period** column and select its corresponding time unit.
19. Click **Continue**.
20. Click **OK**.
21. Stop and restart your server.

## Log and log file tasks

The IBM HTTP Server for i supports numerous log and log file tasks.

### Related information:

“Log formats for HTTP Server” on page 27

This topic provides information about log formats and log files.

## Setting up logs on HTTP Server

Set up logs to record events and other information for your IBM HTTP Server for i instance using the IBM Web Administration for i interface.

Your HTTP Server can generate a record of events commonly referred to as a Log. Logs can contain error messages, information on what is being accessed on your HTTP Server, who is accessing your HTTP Server, script logs, and FRCA logs.

The following topics discuss general log settings required for all logs, Access logs, Error logs, Script logs, FRCA logs, where to find the HTTP Server job log, and how to run a trace.

## General log settings

Before creating a specific log type, the general settings for all logs must be applied to your HTTP Server configuration. To configure the general settings for all logs, do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Select the context you want to work with from the **Server area** list.
5. Expand **Server Properties**.
6. Click **Logging**.
7. Click the **General Settings** tab in the form.

The General Settings allow you to specify log entry time (local or Greenwich Mean Time), the log cycle, and maximum log file size.

8. Click **Apply**.

After you complete the general settings for all logs, you can specify what type of logs you want to create.

## Access Logs

Access logs contain a record of requests to the HTTP Server. The access log itself can be configured to record specific information that you will want to review later. To configure an access log, do the following:

1. See General log settings.
2. Click the **Manage** tab.

3. Click the **HTTP Servers** subtab.
4. Select your HTTP Server from the **Server** list.
5. Select the context you want to work with from the **Server area** list.
6. Expand **Server Properties**.
7. Click **Logging**.
8. Click the **Custom Log** tab in the form.

You can specify various types of information that can be logged in the Access log by specifying a customized log format. For more information how to specify a customized log format see Log Format.

9. Click **Apply**.

## Error Logs

Error Logs contain records of errors that are encountered by visitors to the server. You can specify what types of errors that are logged. To configure error logs, do the following:

1. See General log settings.
2. Click the **Manage** tab.
3. Click the **HTTP Servers** subtab.
4. Select your HTTP Server from the **Server** list.
5. Select the context you want to work with from the **Server area** list.
6. Expand **Server Properties**.
7. Click **Logging**.
8. Click the **Error Logs** tab in the form.

You must first enable error logging to edit what errors will be logged. Once enabled, do the following:

9. Enter the path and name of the error log.
10. Enter an expiration date.
11. The value defines how long the error log will be maintained before information is rolled over.
12. Enter a maximum cumulative size.  
The value defines how large your error log can be before old log entries are deleted.
13. Select logging level.  
From the **Logging level** list, select the level of information you want entered in the error log.
14. Click **Apply**.

## Script Logs

Script Logs contain errors generated by CGI programs running on the server. Generally you should only enable these logs when you are debugging programs on the server. To configure script logs, do the following:

**Note:** Set up a script log only if you are running CGI programs.

1. See General log settings.
2. Click the **Manage** tab.
3. Click the **HTTP Servers** subtab.
4. Select your HTTP Server from the **Server** list.
5. Select the context you want to work with from the **Server area** list.
6. Expand **Server Properties**.
7. Click **Logging**.
8. Click the **Script Logs** tab in the form.

You must first enable script logging to edit what script errors will be logged. Once enabled, do the following:

9. Enter the path and name of the script error log.
10. Enter a maximum log file size.  
The value defines the size of the script error log.
11. Enter a maximum log entry size.  
The value defines the size of the script error log entry.
12. Click **Apply**.

## FRCA Logs

Fast Response Cache Accelerator ( FRCA) is an extension to the HTTP Server that enables caching and serving of data in Licensed Internal Code.

1. See General log settings.
2. Click the **Manage** tab.
3. Click the **HTTP Servers** subtab.
4. Select your HTTP Server from the **Server** list.
5. Select the context you want to work with from the **Server area** list.
6. Expand **Server Properties**.
7. Click **Logging**.
8. Click the **FRCA Logs** tab in the form.

FRCA must be enabled before information is written to the FRCA log. Once enabled, do the following:

9. Click **Add** under the **FRCA logs** table.
10. Enter the path and name of the FRCA log.
11. Enter the log format.

**Note:** For more information how to specify a customized log format see “Log formats for HTTP Server” on page 27.

12. Enter the environment variable conditions.
13. Enter an expiration date.  
The value defines how long the FRCA log will be maintained before information is rolled over.
14. Enter the maximum cumulative size of the FRCA log file.  
The value defines how large your FRCA log can be before old log entries are deleted.
15. Click **Continue**.
16. Click **Apply**.

## HTTP server job logs

The HTTP Server job logs contain messages or exceptions. The HTTP Server job log is maintained in the QHTTSPVR subsystem, listed with a job name matching the name of your HTTP Server instance.

## Run a trace

The HTTP Server trace allows you to view various levels of trace information related to a specific server. You will need to have a 5250 session on the IBM i server your HTTP Server is currently running on.

1. Start a 5250 session.
2. Start the server with a parameter of the STRTCPSVR command. Use the following:
  - -ve (error) for a trace that contains records for all error return codes or exception conditions.



- -vi (information) for a trace that contains -ve level trace records as well as trace records for entry and exit points from application level API's and API parameters.
- -vv (verbose) for a trace that contains -vi level trace records as well as trace records for debugging control flow or data corruption.

For example `STRTCPSVR *HTTP HTTPSVR(JKLSERVER '-vv')`.

### 3. There are three ways to get output from the trace:

- ENDTCPFSVR - When the server is ended the trace data is placed into a spool file. There is a spool file for each job that is running on the server. If a server ends abnormally, trace data is placed into spool files even if tracing is not active at the time of the error.
- DMPUSRTRC - This command dumps the trace data for a specific job to the display or to a physical file member in the QTEMP library. For example:
  - a. Use the WRKACTJOB command to find the server job number. For example `WRKACTJOB SBS(QHTTPSVR)`.
  - b. Dump the user trace to a file in QTEMP. For example `DMPUSRTRC JOB(nnnnnn/QTMHHTTP/MYSERVER)`, where nnnnnn is the job number and MYSERVER is the server.
  - c. Use the DSPPFM command to view the contents of the trace. For example `DSPPFM QTEMP/QAP0ZDMP MBR(QP0Znnnnn)`.
- TRCTCPAPP - You can use the TRCTCPAPP command to initiate a trace after the server is started and to end a trace. To use the TRCTCPAPP command, the server must have been started with the STRTCPSVR command.

**Note:** If you started the trace with the STRTCPSVR and one of the trace startup parameters (-ve, -vi, or -vv), then you must do the following to end the trace:

- a. Enter the TRCTCPAPP SET (\*ON) command to synchronize it with the STRTCPSVR command. For example: `TRCTCPAPP APP(HTTP) SET(*ON) HTTPSVR(JKLSERVER) TRCLVL(*VERBOSE)`.
- b. Enter the TRCTCPAPP SET (\*OFF) command. For example: `TRCTCPAPP APP(*HTTP) SET (*OFF) TITLE('My title')`.

## Proxy tasks

The IBM HTTP Server for i supports proxy tasks.

### Related information:

“Proxy server types and uses for HTTP Server” on page 24

This topic provides information about proxy server types and uses for the IBM HTTP Server for i Web server.

## Setting up forward proxy for HTTP Server

Set up forward proxy for an IBM HTTP Server for i instance using the IBM Web Administration for i interface.

Configure your HTTP Server for forward proxy using the Web Administration for i interface. Only the steps necessary to configure a forward proxy are discussed.

To configure your HTTP Server for forward proxy, do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Select **Global configuration** from the **Server area** list.

**Note:** To configure a forward proxy for a virtual host, select the virtual host from the **Server area** list. See JKL Toy Company creates virtual hosts on HTTP Server for more information.

5. Expand **Server Properties**.

6. Click **Proxy**.
7. Click the **Forward Proxy** tab in the form.
8. Select **Enabled** from the **Forward proxy capabilities** list.
9. Enter the domain default in the **Default domain for unqualified requests** field. The default domain is used if a request does not contain a domain name. For example, `http:\\www`, does not contain a domain name.

**Note:** The remaining fields are not required to set up forward proxy for your HTTP Server. Edit the default values now or return to this form at a later time.

10. Click **OK**.

## Setting up reverse proxy for HTTP Server

This topic provides information about how to set up a reverse proxy for your IBM HTTP Server for i with the IBM Web Administration for i interface.

Configure your HTTP Server for reverse proxy using the Web Administration for i. Only the tabs necessary to configure reverse proxy are discussed.

To configure your HTTP Server for reverse proxy, do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Select **Global configuration** from the **Server area** list.

**Note:** If you want to configure a reverse proxy for a virtual host, click the virtual host from the **Server area** menu. See JKL Toy Company creates virtual hosts on HTTP Server for more information.

5. Expand **Server Properties**.
6. Click **Proxy**.
7. Click the **Reverse Proxy** tab in form.
8. Select **Enabled** from the **Reverse proxy capabilities** list.
9. Click **Add** under the **Proxy request to remote servers** table.

**Note:** This table defines what requests will be mapped into the space of the server. The local server does not act as a proxy in the conventional sense, but appears as a mirror of the remote server.

10. Select **Client requests** from the **Request Type** list.

When this option is used, non-proxy requests matching the URL specified in the **Local virtual path** column are transformed into proxy requests for the URL specified in the **Remote server URL** column. The proxy then handles the transformed request and returns any document (or error messages) the remote server provides. Clients remain unaware of any transformation.

11. Enter the local virtual path in the **Local virtual path** column.

If a non-proxy requests matches the path specified in this column, the non-proxy request will be transformed into a proxy request for the URL specified in the **Remote server URL** column.

12. Select **Specify URL** from the list in the **Remote server URL** column.
13. Enter the remote server URL in the **Remote server URL** column.
14. Click **Add** under the **Proxy requests to remote servers** table.
15. Select **Redirect requests** from the **Request Type** list.

When this option is used for *redirected requests*, headers in response documents are adjusted in the event that a "Redirect" is issued by the remote server. This allows clients to remain unaware of any transformation of the requests even if remote servers redirect the proxy.

16. Enter the path in the **Local virtual path** column.  
If your server is given a non-proxy request and the request matches the URL specified in the **Local virtual path** column, the URL request will be transformed into a proxy request for the URL specified in the **Remote server URL** column.
17. Enter the remote server URL in the **Remoter server URL** column.  
If a non-proxy request matches a URL in the **Local virtual path** column, the request will be transformed in the URL specified in the **Remote server URL** column. The client will be directed to the remote server URL without being aware of the redirect.
18. Click **Continue**.
19. Click **OK**.

All other options for reverse proxy are optional and allow you to modify specific reverse proxy capabilities.

After configuring your HTTP Server for reverse proxy, you can configure your server for a proxy chain.

### Set up proxy chaining for HTTP Server

This topic provides information about how to set up a proxy chain with your HTTP Server and other proxy servers with the IBM Web Administration for i interface.

Configure your HTTP Server for proxy chaining using the Web Administration for i interface. Only the steps necessary to configure a proxy chain are discussed. Before you can configure your HTTP Server for a proxy chain, you must configure your HTTP Server for forward proxy or reverse proxy.

To configure your HTTP Server for a proxy chain, do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Select **Global configuration** from the **Server area** list.

**Note:** If you want to configure a proxy chain for a virtual host, select the virtual host from the **Server area** list. See “JKL Toy Company creates virtual hosts on HTTP Server” on page 61 for more information.

5. Expand **Server Properties**.
6. Click **Proxy**.
7. Click the **Proxy Chaining** tab in the form.
8. Click **Add** under the **Remote proxies** table.
9. Enter the URL of the remote proxy in the **Remote proxy URL** column.

**Note:** If you are not using the default port 80, include the port number with the remote proxy URL.  
Example: `http://www.myserver.com:1975`

**Note:** For the remote server, only the HTTP protocol is supported. HTTPS and FTP are not supported in the remote proxy URL field.

10. Enter the full or partial URL in the **Match requests to forward** column.
11. Click **Continue**.
12. Click **OK**.

### Security tasks

This topic provides step-by-step tasks for security with the IBM HTTP Server for i Web server.

**Related information:**

“Security tips for HTTP Server” on page 29

This topic provides tips to secure your IBM HTTP Server for i Web server.

## Setting up password protection on HTTP Server

Set up password protection for resources on your IBM HTTP Server for i instance using the IBM Web Administration for i interface.

You can protect Web resources by asking the user for a userid and password to gain access to these resources. Group files can be used to classify users into groups (for example: users and administrators). This allows you to limit access to those users that are defined in a group. If the user is listed in the group, then the userid and password are validated in one of the following ways:

- Internet users in a validation list - This requires you to create a validation list that contains Internet users. You can create a validation list and Internet users through the Web Administration for i.
- User profiles password protection - This requires that each user must have a system user profile.
- LDAP password protection - This requires that you configure a LDAP server with the user entries.

### Group file password protection

The following steps explain how to add password protection (using groups) to a directory context.

1. Create a group file with the following format:

```
groupname: user1[, user2[, user3...]]
```

**groupname**

Any name you want to use to identify the group you are defining. This name can be used on subsequent group definitions within the same server group file.

**user1[, user2[, user3...]]**

This can be any combination of user names and group names. Separate each item with a comma.

For example:

```
ducks: webfoot, billface, swandude  
geese: gooseegg, bage1  
flock: ducks, geese
```

In the above example, notice that once the groups named ducks and geese are defined, they can be included as part of the group named flock.

2. Click the **Manage** tab.
3. Click the **HTTP Servers** subtab.
4. Select your HTTP Server from the **Server** list.
5. Select the context you want to work with from the **Server area** list.

**Note:** Do not select Global configuration or Virtual Host. If the Authentication tab cannot be selected, select a different context to work with from the Server area list.

6. Expand **Server Properties**.
7. Click **Security**.
8. Click the **Authentication** tab in the form.
9. Select **Use Internet users in validation list** or **Use IBM i profile of client** under **User authentication method**.

**Note:** Your selection should be based off of the incoming traffic your HTTP Server will receive. If incoming traffic is from outside of your local access network, using Internet users in a validation list would be more beneficial than using IBM i profiles. If incoming traffic is from a local access network, using IBM i profiles would be more beneficial than using Internet users in a validation list.

10. Enter an authentication name or realm. The realm name is displayed on the login prompt.
11. Add a user authentication method if necessary.
12. Click **OK**.

After configuring authentication, you must configure control access.

1. Select the same context you work with previously from the Server area list.
2. Expand **Server Properties**.
3. Click **Security**.
4. Click the **Control Access** tab in the form.
5. Select **Specific users and groups**.
6. Click **Add** under the **User and Group names** table.
7. Select **Group** from the list in the **Type** column.
8. Enter the name of the group in the **Name** column.
9. Enter the path/filename of the group file used above.
10. Click **OK**.

Note that changes to existing group files take effect after the HTTP Server is restarted.

### **User profiles password protection**

You can protect Web resources by asking the user for a userid and password to gain access to these resources. An IBM i user profile can be used to authenticate users.

To configure password protection using a user profile, do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Select the context you want to work with from the Server area list.
5. Expand **Server Properties**.
6. Click **Security**.
7. Click the **Authentication** tab in the form.

**Note:** If the Authentication tab cannot be selected, select a different context to work with from the **Server area** list.

8. Select **Use IBM i profile of client** under **User authentication method**.
9. Enter an authentication name or realm. The realm name is displayed on the login prompt.
10. Choose one of the two methods below:  
Enter a user name in the **IBM i user profile to process requests** field.  
Select a user name under **IBM i user profile to process requests**. Select **Default server profile** to allow the HTTP Server profile (QTMHHTTP) to process requests.
11. Click **OK**.

After configuring authentication, you must configure control access.

1. Select the same context you work with previously from the **Server area** list.
2. Expand **Server Properties**.
3. Click **Security**.
4. Click the **Control Access** tab in the form.

5. Select **All authenticated users (valid user name and password)** under **Control access based on who is making requests**.
6. Click **OK**.

## LDAP password protection

You can protect Web resources by asking the user for a userid and password (to gain access to these resources). A Lightweight Directory Access Protocol (LDAP) server can be used to authenticate users.

LDAP is a directory service protocol that runs over TCP/IP, using non-secure or Secure Sockets Layer (SSL). The LDAP directory service follows a client/server model, where one or more LDAP servers contain the directory data. This allows any LDAP-enabled application to store information once (such as user authentication information). Other applications using the LDAP server are then able to request the stored information. The HTTP server can act as a LDAP client, making requests for information.

One of the advantages of using the LDAP server for authentication is that it allows the information to be shared by multiple LDAP clients, and stores the information in a platform independent fashion. This can help prevent information from being duplicated within a network.

The following steps explain how to add password protection (using LDAP) to a directory context.

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Select the context you want to work with from the **Server area** list.
5. Expand **Server Properties**.
6. Click **Security**.
7. Click the **Authentication** tab in the form.

**Note:** If the Authentication tab cannot be selected, select a different context to work with from the **Server area** list.

8. Select **Use user entries in LDAP server** under **User authentication method**.
9. Enter an authentication name or realm. The realm name is displayed on the login prompt.
10. Enter an LDAP configuration file.
11. Enter an LDAP group name or filter.
12. Click **OK**.

After configuring authentication, you must configure control access.

1. Select the same context you work with previously from the **Server area** list.
2. Expand **Server Properties**.
3. Click **Security**.
4. Click the **Control Access** tab in the form.
5. Select one of the options for who can access this resource.
6. Select one of the options for who can access this resource under **Users and groups who can access this resource**.
7. Select **Allow access to all, except the following** under **Control access based on where the request is coming from**.
8. Enter any domain names or IP address you do not want to allow access to.
9. Click **OK**.

## Setting up to secure against a Telnet denial-of-service attack

This topic provides information about how to secure your IBM HTTP Server for i Web server against a Telnet denial-of-service (DoS) attack using the IBM Web Administration for i interface.

The HTTP Server configuration to protect against Telnet DoS attacks has default settings, but you may want to change them to suit your individual needs.

Your HTTP Server can detect a DoS attack by measuring the time-out and frequency, or the number of time-outs of certain clients' requests. If the HTTP Server does not receive a request from the client, then your HTTP Server determines that a Telnet DoS attack is in progress. This occurs after making the initial client connection to your HTTP Server.

The HTTP Server's default is to perform attack detection and penalization. However, this default may not be right for your environment. If all access to your HTTP Server is through a firewall or proxy server or Internet Service Provider (ISP), then the Telnet DoS protection is built into each of these entities. You should turn off the Telnet DoS protection for this HTTP Server instance so that the HTTP Server does not falsely detect a DoS condition.

To secure against a Telnet DoS attack perform the following steps:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Expand **Server Properties**.
5. Click **System Resources**.
6. Click the **HTTP Connections** tab in the form.

**Note:** The values provided are the current HTTP connections settings used by your Web server. Continue only if you want to change the default values.

7. Enter new values for the provided fields.
8. Click **Apply**.
9. Click the **Denial of Service** tab in the form.

**Note:** The values provided are the current denial-of-service settings used by your Web server. Continue only if you want to change the default values.

10. Enter new values for the provided fields.
11. Click **OK**.

See "User profiles and required authorities for HTTP Server" on page 30 for more information if you encounter authority problems.

## WebDAV tasks

Web-based distributed authoring and versioning (WebDAV) is provided through the IBM HTTP Server for i Web server.

### Related information:

"WebDAV for HTTP Server" on page 49

This topic provides information about Web-based distributed authoring and versioning (WebDAV) for the IBM HTTP Server for i Web server.

## Setting up WebDAV for HTTP Server

Set up WebDAV for your IBM HTTP Server for i instance using the Web Administration for i interface.

Web-based distributed authoring and versioning (WebDAV) is a set of extensions to the HTTP protocol that allows WebDAV clients (such as Microsoft Web Folders) to collaboratively edit and manage files on remote Web servers. See “WebDAV for HTTP Server” on page 49 for more information.

To configure WebDAV on your server, do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Select **Global configuration** from the **Server area** list.
5. Expand **Server Properties**.
6. Click **Request Processing**.
7. Click the **WebDAV** tab in the form.
8. Specify one of the following under **WebDAV lock databases**:
  - **Full path name for locking stream files**: - the full path of the DAV lock database for the Root (/) or QOpenSys streaming file system.
  - **Library/name for locking QSYS objects**: - the library and file name of the DAV lock database for QSYS objects.
9. Click **OK**.
10. Select the context you want to work with from the **Server area** list. The server area you select will be WebDAV enabled.
11. Click **Request Processing**.
12. Click the **WebDAV** tab in the form.
13. Select **Enabled** to Enable WebDAV.
14. Enter the appropriate file system under **Repository provider**.
15. Enter any **WebDAV restrictions** you want enabled for this server area.
16. Click **OK**.

## Web tasks

This topic provides step-by-step tasks for accessing Web applications with the IBM HTTP Server for i Web server.

### Integrated Web application server

The integrated Web application server provides a Web container for dynamic Web applications that uses minimal system resources, is easy to configure, and is imbedded into IBM i.

#### Details

The integrated Web application server is ready for use without having to install any additional products. This Web container is capable of running Servlet or JSP applications. The integrated Web application server is an ideal choice for less complex applications, or applications that are not required to be highly scalable. Since the Web container has a minimal resources foot print, low use applications that have a few users are ideal to run on the integrated Web application server. This Web container is also a good choice when working on a proof of concept because no other products are necessary, the server is easy to create, and applications are easy to deploy. For applications that require a high degree of scalability, the IBM WebSphere Application Server product should be used.

The integrated Web application server is built on an infrastructure that leverages the Open Services Gateway initiative (OSGi) and Eclipse platforms. The integrated Web application server is available for IBM i 5.4, or later. The server supports dynamic Web applications running JSP and servlets. Support for database connectivity is also included for DB2 on either local or remote systems. Any Web application that was targeted to be run on a Web container such as AFS Tomcat would be a good candidate.



The IBM Web Administration for i interface has been updated to include full support for the integrated Web application server. The interface provides several simple, easy to use wizards to create new server instances, stop and start servers, and deploy and manage the applications running on each server.

One of the main differences between the integrated Web application server and WebSphere Application Server is the format of the applications. Applications that run in WebSphere Application Server are in the EAR or WAR format. Applications for the integrated Web application server need to be in the Web Application Bundle (WAB) format. The WAB architecture consist of the jar files for the application and the manifest file that details all the class files that are used by the application. A WAB file is typically created as a Java Archive (JAR) file with the .jar file extension. Using the Web Administration for i interface to deploy an application that is in a WAR file will deploy with out issue. The interface will convert the WAR file into a WAB file before it is deployed to the Web server.

## Prerequisites and assumptions

Ensure the HTTP Server prerequisites have been installed. In addition, load the latest HTTP Server group PTF.

## Creating an integrated Web application server

The Web Administration for i interface provides an easy to use wizard to create an integrated Web application server on your IBM i server.

1. Access the IBM Web Administration for i from your browser. For information about how to access the Web Administration for i interface, see “Starting Web Administration for i” on page 7.
2. From the Web Administration for i interface, select the **Setup** tab, and then click **Create Application Server** to launch the Create Application Server wizard.
3. Click **Next** after reading the wizards welcome page.
4. From the **integrated Web application server:** section select **V7.1**.
5. Complete the wizards to create an integrated Web application server. Click on the (?) icon to display the help information for a particular panel.

## Installing an application on the integrated Web application server

Now that you have created an integrated Web application server you will want to install and run applications on the server. The Web Administration for i interface provides an easy to use wizard to install your applications. Remember, you can only install applications that are contained in a Web Archive (WAR) file or Web Application Bundle (WAB).

Complete the following steps to install an application on an integrated Web application server:

1. Access the IBM Web Administration for i from your browser. For information about how to access the Web Administration for i interface, see “Starting Web Administration for i” on page 7.
2. From the IBM Web Administration for i interface, select the **Manage** tab.
3. Select the **Application Servers** subtab.
4. Select the integrated Web application server that you want to install the application from the **Server** list.
5. Click **Install New Application** to launch the Install New Application wizard.
6. Complete the wizards to install your application. Click on the (?) icon to display the help information for a particular panel.

## Creating a database connection for an integrated Web application server application

The Create Database Connection wizard helps the user create a new entity that allows them to connect the integrated Web application server to a specified database. The database connection allows installed

applications to retrieve and store information in a database. There are two different database connection types supported by the integrated Web application server. The IBM Developer Kit for Java database provider is available if you need to connect to a DB2® database. The Derby Database Provider is available if you need to connect to a Derby database.

For applications to use the database connection wizard they must meet the following requirements:

- The JNDI name used by the application must be set to jdbc/ProxyDS.


**Note:** This process is different for a WAS where each datasource has its own JNDI key.

- The connection ID must be passed on the getConnection() call. The connection ID used must match the name of the database connection created in the wizard. If no parameters are passed on the getConnection() call, the \*DEFAULT database connection properties are used.

Complete the following steps to create a database connection:

1. Access the IBM Web Administration for i from your browser. For information about how to access the Web Administration for i interface, see “Starting Web Administration for i” on page 7.
2. From the IBM Web Administration for i interface, select the **Manage** tab.
3. Select the **Application Servers** subtab.
4. Select the integrated Web application server that you want to create a database connection from the **Server** list.
5. Click **Create Database Connection** to start the Database Connection wizard
6. Complete the wizards to install your application. Click on the (?) icon to display the help information for a particular panel.

**Related information:**

 [Integrated Web Application Server home page](#)

## Integrated Web services for i

In support of Web services and Service Oriented Architecture (SOA), the IBM i operating system integrates software technologies that support externalizing integrated language environment (ILE) program objects as Web serviced and the consumption of a Web service by an ILE program object. These technologies are the integrated Web services server and the integrated Web services client for ILE.

### Overview of technology

Web service technology promises a new range of possibilities for how organizations and their partners interoperate to offer dynamic e-business solutions. Web services connect business applications to each other, inside and outside the enterprise, regardless of their platform, design, or runtime environment. IBM provides the tools, protocols, technologies, support, and commitment to open standards, to help businesses create and use innovative Web services technology.

A Web service is a self-contained software component with a well-defined interface that describes a set of operations that are accessible over the Internet. XML technology provides a platform and programming language-independent means by which a Web service's interface can be defined. Web services can be implemented using any programming language, and can be run on any platform, as long as two components are provided to indicate how the Web service can be accessed: a standardized XML interface description, called WSDL (Web Services Description Language), and a standardized XML-based protocol, called Simple Object Access Protocol (SOAP). Applications can access a Web service by issuing requests formatted according to the XML interface.

Web services do not provide a Graphical User Interface (GUI) for the user. Instead, Web services share business logic, data, and processes through a programming interface across a network. Therefore,

developers can access Web services from applications to gain specific functionality. In short, Web services are encapsulated functions which are offered using broadly adopted standard interface descriptions and protocols.

The Web services architecture is based on the interactions among three roles: service provider, service registry, and service requestor. The interactions involve the publish operations, find operations, and bind operations. Together, these roles and operations act upon the Web service artifacts: the Web service software module and its description. In a typical scenario, a service provider defines a service description for the Web service using Web Services Description Language (WSDL). The WSDL description of the service is then published to the service requestor or service registry. The service requestor uses a find operation to retrieve the service description locally or from the service registry. Once obtained, the service description is used to bind with the service provider and invoke or interact with the Web service implementation.

#### **Service Provider (integrated Web services server)**

From a business perspective, this is the owner of the service. From an architectural perspective, this is the platform that hosts access to the service.

#### **Service Requestor (integrated Web services client for ILE)**

From a business perspective, this is the business that demands that certain requirements be satisfied. From an architectural perspective, this is the application that is looking for and invoking, or initiating, an interaction with a Web service. The service requestor role can be played by a browser driven by a person, a program with a user interface, or a program without a user interface.

### **Prerequisites and assumptions**

Ensure the HTTP Server prerequisites have been installed. In addition, load the latest HTTP Server group PTF.

### **Integrated Web services server**

The integrated Web services server for IBM i greatly simplifies the process of externalizing ILE business logic as a service via the IBM Web Administration for i interface. The externalization of RPG and COBOL business logic as a service has been simplified to be an administrative task on IBM i. This simplification has been accomplished by abstracting the hidden complexities of Web services and extending the ILE programming model, to allow a System i administrators to directly externalize various ILE business tasks as services.

### **Creating a Web services server**

The Create New Web Services server wizard provides a convenient way to externalize programs running on IBM i, such as RPG or COBOL ILE programs, as Web Services.

1. Access the IBM Web Administration for i from your browser. For information about how to access the Web Administration for i interface, see “Starting Web Administration for i” on page 7.
2. From the IBM Web Administration for i interface, select the **Setup** tab, and then click **Create Web Services Server** to launch the Create Web Services Server wizard.
3. Complete the wizards to create a Web services server. Click on the (?) icon to display the help information for a particular panel.

### **Externalizing IBM i programs as Web services**

The Install New Service wizard provides a convenient way to externalize an IBM i program or service program as a Web Service. The wizards provides steps to specify the program object, select program export procedures to be made available through the Web service, and other parameters. When you finish the wizard, the Web service artifacts are created and a new Web service is deployed on the server.

1. Access the IBM Web Administration for i from your browser. For information about how to access the Web Administration for i interface, see "Starting Web Administration for i" on page 7.
2. From the IBM Web Administration for i interface, select the **Manage** tab.
3. Select the **Application Servers** subtab.
4. Select the Web services server that you want to install the new service on from the **Server** list.
5. Click **Install New service** to launch the Install New Service wizard.
6. Complete the wizards to install your program as Web services. Click on the (?) icon to display the help information for a particular panel.

## Web services client for ILE

The Web services client is integrated into IBM i, providing a mechanism to generate service artifacts and allow ILE (RPG, COBOL, C, C++) to act as a services consumer with enablement for calling a variety of Web service implementations, including RPG, COBOL, C, C++, Java, PHP, .NET, WebSphere Process Server (WPS), and WebSphere Enterprise Service Bus (ESB).

The following lists some of the benefits and features for the web services client:

- Natural extension for the ILE programmer to consume services from a program or service program.
- ILE enablement to bind and call a service directly from IBM i service program or program.
- Leverages WSDL to generates proxy client code to be integrated in program or service program.
- Enhances existing System i development skills to interact with Web services and SOA.

## Creating ILE Web service client stub (proxy) service program

Before you can create a Web service client application, you must first generate the client stubs using the `wSDL2ws.sh` tool.

1. Copy the WSDL file to a directory in which the client stubs will be generated.
2. Open Qshell and change the current working directory to where the WSDL file is located. For example, if the WSDL source file `GetQuote.wsdl` is in `/stockquoteWS`, then you would specify: `cd /stockquoteWS`
3. Run the `wSDL2ws.sh` tool with the following command to generate the client stubs:  
`/qibm/proddata/os/webservices/v1/client/bin/wSDL2ws.sh GetQuote.wsdl`

**Note:** The command above generates C++ stubs. To generate C stubs simply add the `-lc` option to the command. For example:

```
/qibm/proddata/os/webservices/v1/client/bin/wSDL2ws.sh GetQuote.wsdl -lc .
```

4. Examine the generated web service stub artifacts in the IBM i Integrated File System (IFS), to determine the interfaces for ILE service programs/programs to interact and invoke the service stub code.
5. Compile the C or C++ stubs you generated in the previous step. In the following example the generated stub file is `StockQuote.cpp`:  
`CRTCPPMOD MODULE(MYLIB/STOCKQUOTE) SRCSTMF('/stockquoteWS/StockQuote.cpp') INCDIR('/qibm/proddata/os/webservices/v1/client/include') ENUM(*INT).`
6. Create the Web service client proxy service program.

**For C++ stubs, your will need to bind to service program QSYS/QAXIS10C.**

```
CRTSRVPGM SRVPGM(MYLIB/GETQUOTEWS) MODULE(MYLIB/STOCKQUOTE) EXPORT(*ALL)
BNDSRVPGM(QSYS/QAXIS10C)
```

**For C stubs, you will need to bind to service program QSYS/QAXIS10CC.**

```
CRTSRVPGM SRVPGM(MYLIB/GETQUOTEWS) MODULE(MYLIB/STOCKQUOTE) EXPORT(*ALL)
BNDSRVPGM(QSYS/QAXIS10CC)
```

Once the client stubs have been generated and a service program containing the stubs created, you can now develop a Web service client application that can invoke the Web service via the stubs. More information on Web services client programming using Web services client for ILE can be found in the PDF files located in /qibm/proddata/os/webservices/v1/client/docs.

#### **Related information:**

- [Integrated Web Services for i home page](#)
- [Consuming Web services from RPG or COBOL programs on System i](#)
- [New to SOA and Web services](#)
- [WebSphere Application Server for IBM i](#)
- [WebSphere Enterprise Service Bus](#)
- [WebSphere Process Server](#)
- [WebSphere Integration Developer](#)

### **Web Performance Monitor**

The IBM Web Administration for i interface provides the Web Performance Monitor to monitor and evaluate the performance characteristics of your Web environment.

A Web environment is a grouping of related HTTP and application servers that form a Web solution. In a Web environment, you may have to monitor the performance of application servers, HTTP servers, databases, and applications jobs. The Web Performance Monitor provides a single interface to monitor your entire Web environment, to find how your system time and resources are being spent, and to identify potential bottlenecks. Only one Web environment on the system can be monitored at one time.

#### **Details**

When performance monitoring is enabled using the Web Performance monitor, your HTTP server instances are stopped their configurations are updated to enable ARM data collection, your application servers are stopped and their configurations are updated to enable ARM data collection, ARM data collection is started, and your HTTP server and application servers are restarted to activate the configuration changes. You should only enable Web performance monitoring when it is acceptable to not access the Web environment for a period of time, as your HTTP and application servers are stopped while their configurations are updated. Enabling performance monitoring may take several minutes to complete.

When the Web Performance Monitor is enabled, the **Transactions** tab displays the transaction statistics for jobs that are collecting ARM data in your environment. For each transaction, the average response time and the number of completed transactions information is shown. Very large results for average response time may indicate a bottleneck and provide a starting point to investigate a performance problem.

You can also choose to look at the transaction times for a single IP Address or hostname, allowing you several creative ways to figure out what is going on within an application. The transaction information shows what jobs the processing requests for each user or IP address has been processed by. This very specific user information can help identify which jobs are slowing things down.

When the Web Performance Monitor is enabled, the **Jobs** tab displays the active servers and jobs in your environment. The Web Performance Monitor tool displays only the jobs on the system that are related to the selected Web server. For example, if the HTTP server is on a remote system, then there will not be any jobs for the HTTP server displayed. For each job, the current user, CPU usage percentages, run priority, number of threads, average transaction response time, and number of complete transactions is provided. Very large results for average response time or CPU usage percentage may indicate a bottleneck and provide a starting point to investigate a performance problem.

When performance monitoring is disabled using the Web Performance monitor, your HTTP server instances are stopped and their configurations are updated to disable ARM data collection, your application servers are stopped and their configurations are updated to disable ARM data collection, ARM data collection is stopped, and your HTTP server and application servers are restarted to activate the configuration changes. You should only disable Web performance monitoring when it is acceptable to not access the Web environment for a period of time, as your HTTP and application servers are stopped while the configurations are updated. Disabling performance monitoring may take several minutes to complete.

## Prerequisites and assumptions

Each of the following WebSphere Application Server products must be at the fix level specified before Web Performance Monitor can work. When WebSphere Application Server fixes are installed, the activation instructions must be followed completely, and the ADMIN server must be stopped and restarted. The following versions are supported:

- WebSphere Portal V6.1.5
- WebSphere Application Server V6.1 (Base and Express editions and Network Deployment in a stand-alone environment)
  - Base/Express/ND - PTF Group SF99323 Level 5 or higher
- WebSphere Application Server V7 (Base and Express editions and Network Deployment in a stand-alone environment)

## Start the Web Performance Monitor

The Web Performance Monitor can be started from the Web Administration for i interface:

1. Access the Web Administration for i from your browser. For information about how to access the Web Administration for i interface, see “Starting Web Administration for i” on page 7.
2. From the IBM Web Administration for i interface, select the server you want to monitor.
3. In the navigation pane, expand **Web Performance**, and select **Web Performance Monitor**.

**Note:** If **Web Performance Monitor** is not displayed in the navigation pane, either you need to install the latest HTTP Server group PTF, or the selected server is not supported by the Web Performance Monitor.

If Web Performance Monitor is not currently active, the **Enable** panel will be shown. Click **Start** to enable monitoring.

Once you are satisfied with the data you've collected, you should disable the Web Performance Monitor, as it requires extra system resource.

## Web Performance Advisor

The Web Performance Advisor provides a way to view, evaluate and modify the attributes that affect the performance of your Web environment. Clear definitions of the attributes are provided along with recommended values. The tool also provides rating for each attribute to help guide the user to acceptable settings.

A Web environment is a grouping of related Web and application servers that form a Web solution. A Web environment is typically made up of a single application server, its corresponding IBM HTTP Server for i Web server, and any system attributes that could have a direct effect on the performance of the Web environment. Supported application servers include WebSphere Application Server, Integrated Web Application Server for i, and the Integrated Web Services Server for i.

The Web Performance Advisor is made up of multiple components to help you tune the performance of your system and Web environment. These components include an advisor and an export function. These

can be launched from the Web Performance Advisor introduction page. On this introduction page, the user is provided a quick, easy-to-read, high-level view of their system and Web environment performance.

The Advisor function allows you to manage system attributes and to manage Web environment attributes. From the manage system and manage Web environment panels, you can view, evaluate, and change each performance attribute. While evaluating each performance attribute, click the attribute's Advise link to learn about the attribute and find the recommended setting.

The export function allows you to save existing performance settings in a performance profile. This profile can be evaluated, compared, or sent to a performance expert for analysis and modification.

When the Web Performance Advisor tool is used to examine a Web related server, a flight recorder performance profile is created to save what all performance attributes are set to prior to any changes being made. Whenever changes are made through the Web Performance Advisor, all the performance attributes are saved (including the new changes) to another flight recorder performance profile file. This is necessary so that you can keep track of all changes made to a Web environment. All flight recorder performance profile files are located in the '/QIBM/UserData/HTTPPA/admin/WPA' directory. The Web Performance Advisor tool does not clean up these files; they remain until someone deletes them manually.

Because the attributes affecting performance in a Web environment are located in many places, the Web Performance Advisor combines all of the performance attributes into a performance profile. The profile contains:

- System attribute information made up of the physical and logical resources that have been allocated to the system and partition and selected system values that can have a direct effect on Web performance, TCP/IP settings, and PTF information including the PTF Groups and the individual product PTFs for the products that are used in a Web environment.
- Web attribute information for an application server.
- Web performance attributes for an application server, including the JVM settings, system and server resource settings, server JDBC providers and data source resources, and other additional server settings.
- Web attribute information related to your external HTTP server that is associated with the application server.

## Details

The Web Performance Advisor gathers ratings and recommendations for each of the performance attributes being tuned. From these ratings, icons are displayed to convey whether the attribute is tuned well (green), may need some additional tuning (yellow), or needs immediate attention (red). The ratings that are displayed may vary based on the risk level (conservative or aggressive) you have configured in the General Settings. Conservative means that you do not want to be alerted to those performance attributes that are on the fringe. By using the conservative approach, fewer attributes are changed and drastic performance updates are not made. Of course, performance may not be tuned as well, but there is much less risk of degrading your machine as a whole. Using the aggressive approach, any attribute that is on the fringe is flagged as needing to be changed. In addition, attributes that would be flagged as good in a conservative mode, might actually be flagged as needing improvement. By doing this, more drastic performance updates are made which may dramatically improve performance. On the downside, the possibility exists that unexpected, unwanted consequences may result from these drastic performance changes.

## Prerequisites and assumptions

The Web Performance Advisor feature supports a wide variety of WebSphere and non-WebSphere products. These include WebSphere Application Server, WebSphere Portal Server, Integrated Application

Server, and the Integrated Web Services Server. The other product that is supported is the IBM HTTP Server for i Web server when it is configured to be used by one of the previously listed products.

Each of the following WebSphere Application Server products must be at the fix level specified before Web Performance Advisor can work. When WebSphere Application Server fixes are installed, the activation instructions must be followed completely, and the ADMIN server must be stopped and restarted. The following versions are supported:

- | • WebSphere Application Server V7 (Base and Express® editions and Network Deployment in a stand-alone environment)
- | • WebSphere Application Server V6.1 (Base and Express editions and Network Deployment in a stand-alone environment)
- | • WebSphere Portal V6.1.5

**Note:** The Web Performance Advisor supports an HTTP server when it is configured to be used by one of the other products supported. Standalone HTTP servers are not supported.

### Start the Web Performance Advisor

The Web Performance Advisor can be started from the Web Administration for i interface:

1. Access the Web Administration for i from your browser. For information about how to access the Web Administration for i interface, see “Starting Web Administration for i” on page 7.
2. From the IBM Web Administration for i interface, select the server you want to examine.
3. In the navigation pane, expand **Web Performance**, and select **Web Performance Advisor**.

**Note:** If **Web Performance Advisor** is not displayed in the navigation pane, either you need to install the latest HTTP Server group PTF, or the selected server is not supported by the Web Performance Advisor.

The Web Performance Advisor introduction page displays. From this page, you can select to manage your system or your Web attributes, change your general settings, or export your current performance settings.

### More performance tuning tools

The Web Performance Advisor is only one tool available for you to tune your performance settings. The Web Performance Advisor, the Workload Estimator, and the documented minimums are all tools available to help you achieve improved performance. These resources can be used together to find the settings that are best for you.

**Documented minimums:** This is the smallest possible system you should run on. These systems may be appropriate for development or internal systems with a small number of users where longer response times are acceptable. Good performance is not expected on these systems.


**Workload Estimator:** This tool accounts for the specific characteristics of your workload to recommend an appropriate system. It should be used to determine the size and type of system that should be used based on the type of workloads you plan on running. It cannot recommend a system that is smaller than the documented minimum recommendations, but it may recommend a larger system.

**Web Performance Advisor:** This tool is recommended if you are trying to get good performance out of your applications and environment. It may recommend configurations that are somewhat larger than the documented minimums. The recommendations could, however, be smaller than the Workload Estimator recommendations, because the Web Performance Advisor does not account for the specific workload your system faces during the runtime of your applications or other things that may be running on your system.




## WebSphere Portal

- | The IBM Web Administration for i interface provides an easy to use wizard to configure WebSphere Portal V6.1.5 and WebSphere Portal Express V6.1.5 on your IBM i server.

It is recommended that you use the **Create WebSphere Portal wizard** in the Web Administration for i interface when configuring your WebSphere Portal server. This reduces the time required to configure and start the Portal server and reduces the complexity of the many different features of the Portal server. Information documenting the installation and configuration of the Portal server on an IBM i server is located in the WebSphere Portal Product Documentation  Web site.

WebSphere Portal is a J2EE application that runs on WebSphere Application Server. WebSphere Portal creates an environment that provides connectivity, administration, and presentation services that allow portal users a means for doing business efficiently and with high satisfaction. The wizard will help you create and configure the elements of a portal server. The following elements are configured through the wizard:

- **WebSphere Application Server:** The WebSphere Application Server is the engine that drives the WebSphere Portal. This is the Web server that provides J2EE services for the portal environment. For more information on WebSphere Application server, see WebSphere Application Server for IBM i .
- **HTTP Server:** The HTTP server routes the incoming URL requests to the proper locations. All dynamic requests are routed by the HTTP server to the application server, which then displays the required portlets.
- **WebSphere Portal:** WebSphere Portal provides a composite application or business mashup framework and the advanced tooling needed to build flexible, Service Oriented Architecture (SOA)-based solutions, as well as the unmatched scalability required by any size organization.
- **LDAP:** The LDAP server controls who has access to the portal server. In the portal environment, it stores, updates, and retrieves user-specific data related to authentication such as users and passwords. For more information on LDAP, see IBM Tivoli Directory Server for i (LDAP).
- **Database:** DB2<sup>®</sup> database is used by the WebSphere Portal as a repository for user customized portal Web pages and portal configuration information. For more information on DB2, see Database.

## Prerequisites and assumptions

### Minimum hardware requirements

You can use the IBM Systems Workload Estimator for help with sizing all system configurations. It might be possible to use systems that do not meet the recommended minimums in environments that support a limited number of users and where longer server initialization times can be tolerated, but performance may suffer.

- **Processor:** Recent IBM i servers with at least 3800 CPW. Production environments should consider servers with a higher CPW rating.
- **Physical memory:** 5 GB at a minimum. Production environments and environments featuring multiple profiles require more memory.
- **Disk space:** The following values are required if you use the installation program to install WebSphere Application Server, extensions and fixes, and WebSphere Portal Express. These values assume a minimum of 6 disk arms and do not account for any other products or components that could be installed on the WebSphere Portal Express machine. Refer to the documentation for the other products for additional hardware requirement information. You can perform a custom installation of the components. The following list shows the space requirements by component:

Component	Install directory	Temp directory
WebSphere Portal Express and WebSphere Application Server V6.1.0.27 and each WebSphere Application Server profile configured for use with WebSphere Portal Express V6.1.5	5 GB	1 GB

### Minimum software requirements

- IBM HTTP Server for i (5770-DG1)
- IBM Developer Kit for Java (5761-JV1)
- IBM Developer Kit for Java (5761-JV1 Option 7)
- IBM TCP/IP Connectivity Utilities for i (5770-TC1)
- Host Server (5770-SS1 Option 12)
- Qshell (5770-SS1 Option 30)
- Portable App Solutions Environment (5770-SS1 Option 33)
- Additional Fonts (5770-SS1 Option 43) - required for Document Manager on IBM i
- IBM WebSphere Application Server V6.1 (5733-W61)
- IBM WebSphere Application Server V6.1 (5733-W61 Option 1)
- WebSphere Portal 6.1.5

### Start the WebSphere Portal wizard

The WebSphere Portal wizard can be started from the Web Administration for i interface:

1. Access the Web Administration for i from your browser. For information about how to access the Web Administration for i interface, see “Starting Web Administration for i” on page 7.
2. From the IBM Web Administration for i interface, select **Common Tasks and Wizards -> Create WebSphere Portal** .

The WebSphere Portal wizard walks you through the creation of a new WebSphere Portal server.

### Virtual host tasks

This topic provides step-by-step tasks for configuring virtual hosts in the IBM HTTP Server for i Web server.

#### Related information:

“Virtual hosts on HTTP Server” on page 23

This topic provides information about virtual host types on the IBM HTTP Server for i Web server.

### Setting up virtual hosts on HTTP Server

Set up virtual hosts on your IBM HTTP Server for i instance using the IBM Web Administration for i interface.

Virtual hosts allow more than one Web site on one system or Web server. The servers are differentiated by their host name. Visitors to the Web site are routed by host name or IP address to the correct virtual host. Virtual hosting allows companies sharing one server to each have their own domain names. For example, *www.company1.com* and *www.company2.com* can both be hosted on the same server. See “Virtual hosts on HTTP Server” on page 23 for more information.

You can configure virtual hosts by doing the following:

1. Click the **Manage** tab.

2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Select **Global configuration** from the **Server area** list.
5. Expand **Server Properties**.
6. Click **Virtual Hosts**.
7. Click either the **Name-based** virtual host tab or the **IP-based** virtual host tab in the form.

### **Name-based virtual hosts**

The name-based virtual host allows one IP address to host more than one Web site (hostname). This approach allows a single HTTP Server to service requests directed at many different hostnames. This simplifies configuration and use, and requires no additional hardware or software. The main disadvantage to this approach is that the client must support HTTP 1.1 (or HTTP 1.0 with 1.1 extensions) that include the server hostname information inside the HTTP document requests. The latest versions of most browsers support HTTP 1.1 (or HTTP 1.0 with 1.1 extensions), but there are still old browsers that only support HTTP 1.0. For more information on virtual hosts refer to the <VirtualHost> directive.

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Select **Global configuration** from the **Server area** list.
5. Expand **Server Properties**.
6. Click **Virtual Hosts**.
7. Click the **Name-based** tab in the form.
8. Click **Add** under the **Named virtual hosts** table.
9. Select or enter an IP address in the **IP address** column.

**Note:** The Web Administration for i provides the IP addresses used by your IBM i system in the IP Address list; however, you will need to provide the hostname associated with the address you choose and register the hostname with your Domain Name Server (DNS).

10. Enter a port number in the **Port** column.
11. Click **Add** under the **Virtual host containers** table in the **Named host** column.
12. Enter the fully qualified server hostname for the virtual host in the **Server name** column.

**Note:** Make sure the server hostname you enter is fully qualified and associated with the IP address you selected.

13. Enter a document root for the virtual host index file or welcome file in the **Document root** column.
14. Click **Continue**.
15. Click **OK**.

### **IP-based virtual hosts**

The IP-based virtual host requires one IP address per Web site (host name). This approach works very well, but requires a dedicated IP address for every virtual host. For more information on virtual hosts refer to the <VirtualHost> directive.

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Select **Global configuration** from the **Server area** list.
5. Expand **Server Properties**.

6. Click **Virtual Hosts**.
7. Click the **IP-based** tab in the form.
8. Click **Add** under the **Virtual host containers** table.
9. Enter a valid IP address in the **IP address or hostname** column.
10. Enter a valid port number in the **Port** column.
11. **Optional:** Enter a server name in the **Server name** column.
12. **Optional:** Enter the document root from where the files will be served in the **Document root** column.
13. Click **Continue**.
14. Click **OK**.

## Mass-dynamic virtual hosting

Use the Mass-dynamic tab to create a dynamic virtual host with a Name-based or IP-based virtual host, or work with canonical names. A canonical name is the actual name of an HTTP Server resource. For example, a canonical name of the HTTP Server is its true name rather than an alias. See directive `<UseCanonicalName>` for more information.

The dynamic virtual host allows you to dynamically add Web sites (hostnames) by adding directories of content. This approach is based on automatically inserting the IP address and the contents of the Host: header into the pathname of the file that is used to satisfy the request.

The Mass-dynamic tab provides a subset of options that are more complex than those provided by the other tabs. The options include specifying the root directory for serving files, and selecting the root directory for CGI scripts. The availability of these settings are dependent on what server area you are working with.

At the global configuration server area, all mass-dynamic settings are available. These include:

- Options on how to build self-referencing URL's.
- Options for the root directory for serving files.
- Options for the root directory for CGI scripts.

The mass-dynamic settings use strings and substrings to create a dynamic virtual hosts. For example, to create a simple dynamic virtual host, the Root directory for serving files option is defined as `/usr/local/apache/vhosts/%0` and **Use server name** is selected. A request for `http://www.ibm.com/directory/file.html` returns `/usr/local/apache/vhosts/www.ibm.com/directory/file.html`.

The string `%0` is an interpolate (insert) string of the server name or IP address. The following defines the interpolate string:

Interpolate (insert) strings	
%%	inserts a %
%p	inserts the port number of the virtual host
%N.M	inserts (part of) the name

**N** and **M** are used to specify substrings of the name. **N** selects from the period-separated components of the name, and **M** selects characters within whatever **N** has selected. **M** is optional and defaults to zero if it is not present; the period must be present if and only if **M** is present. The interpretation is as follows:

Substring interpretation	
0	the whole name

Substring interpretation	
1	the first part
2	the second part
-1	the last part
-2	the next to last part
2+	the second and all subsequent parts
-2+	the next to last part and all preceding parts
1+ and -1+	the same as 0

For more information on mass-dynamic virtual hosts refer to `mod_vhost_alias`.

## CGI tasks

This topic provides step-by-step tasks for configuring various HTTP Server attributes that affect how CGI programs are run within your Web server.

### Related concepts:

“CGI” on page 40

The Common Gateway Interface (CGI) specification was introduced to enable and standardize the interface between Web servers and external programs. The CGI is a relatively simple, platform and language independent, industry-standard interface for Web application development. Programs that implement the CGI standard are commonly called *CGI programs*.

## Setting up CGI jobs

Use this topic to set up CGI jobs that can run on your IBM HTTP Server for i Web server.

To set CGI settings, do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Select the context you want to work with from the **Server area** list.
5. Expand **Server Properties**.
6. Click **Dynamic Content and CGI**.
7. Click the **General Settings** tab in the form.
8. Enter the values associated with your CGI jobs.
9. Click **OK**.

See CGI Programming examples  for sample CGI programs.

## Setting up persistent CGI jobs

Use this topic to set up persistent CGI jobs that can run on your IBM HTTP Server for i Web server.

Persistent CGI is an extension to the CGI interface. It allows a CGI program to remain active across multiple browser requests and maintain a client session. To set persistent CGI settings, do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Select the context you want to work with from the **Server area** list.
5. Expand **Server Properties**.

6. Click **Dynamic Content and CGI**.
7. Click the **Persistent CGI** tab in the form.
8. Enter the values associated with the persistent CGI jobs.
9. Click **OK**.

See CGI Programming examples  for sample CGI programs.

## Apache module tasks

This topic provides step-by-step tasks for configuring Apache modules to extend the functionality of your IBM HTTP Server for i Web server.

### Related concepts:

“Apache modules” on page 42

Modules are service programs that can be dynamically linked and loaded to extend the nature of the HTTP Server.

## Setting up Apache modules

Use this topic to configure an Apache module in order to extend the functionality of your IBM HTTP Server for i Web server.

To add an Apache model to your Web server, you will need to add the `LoadModule` directive to your HTTP Server configuration. Do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the **Server** list.
4. Expand **Tools**.
5. Click **Edit Configuration File**.
6. Add the following, where *MODULELIB* is the library in which the module service program resides and *MODULE* is the name of your module:

```
LoadModule Module /QSYS.LIB/MODULELIB.LIB/MODULE.SRVPGM
```

7. Add any additional directives needed to the configuration file. Note that the compiled service program, `MOD_FOOTER`, is located in the `QSYS` directory. The `.` represent existing lines in the configuration file.

Example (replace *MYLIB.LIB* with your library name):

```
LoadModule footer_module /QSYS.LIB/MYLIB.LIB/MOD_FOOTER.SRVPGM
```

```
·  
·  
·
```

```
<Directory "/www/mydocs/htdocs">  
  SetOutputFilter FOOTERFILTER  
  FooterFile footer.hf  
</Directory>
```

The **FOOTERFILTER** output filter and the **FooterFile** directive are defined in `MOD_FOOTER`, the module that was compiled and configured.

8. Click **OK**.

---

## Programming

This topic provides information on CGI programming, Apache module programming, APIs, and other programming topics for the IBM HTTP Server for i Web server.

## Application Programming Interface

This topic lists the application programming interfaces (APIs) that are supported by IBM HTTP Server for i.

### Apache module APIs

This topic provides information about the Apache portable runtime (APR) and application programming interfaces (APIs) for the IBM HTTP Server for i. These APIs are generally used to write cross-platform Apache modules.

Links to the HTTP Server and APR APIs are listed below. To write, compile, and configure an HTTP Server module, you will need to use both APR and HTTP Server APIs.

**Note:** The APR APIs are actually independent of the HTTP Server. Users of APR can create their own applications using APR and not touch any Web servers.

- APR Core APIs - The APR APIs are not application specific and may be used with different server types and applications.
- HTTP Server APIs - The HTTP Server APIs are HTTP Server specific and are not part of the APR APIs.


“Module mod\_example” on page 355 provides a simple example of the use of the HTTP Server and APR APIs.

#### Related information:

“Apache module programming” on page 198

The IBM HTTP Server for i supports the extension of the functionality of the HTTP Server through the use of third-party Apache modules.

 [Developer Documentation for Apache 2.0](#)

 [Apache Portable Runtime Project](#)

### CGI APIs

This topic provides information about IBM HTTP Server for i APIs for CGI applications.

HTTP Server supports the APIs listed below in C++, REXX, ILE C, ILE COBOL, and ILE RPG programming languages. Although all APIs are supported in all of these languages, most ILE C CGI applications will only need to use QtmhCvtDB(), QzhhCgiParse(), or QzhhCgiUtils(). This is because ANSI C can work with stdin, stdout, and environment variables directly. ILE C CGI applications use ANSI C function calls to work with stdin, stdout, environment variables, and string functions for parsing stdin and environment variable data.

To use these APIs in a CGI application, you must bind the CGI program to \*SRVPGM QZHBCGI in library QHTTSPVR. ILE C programs must include header file QSYSINC/H(QZHBCGI). CGI application programs must be written and compiled in Integrated Language Environment® ILE C, ILE RPG, and ILE COBOL.

- “Get Environment Variable (QtmhGetEnv) API” on page 140
- “Put Environment Variable (QtmhPutEnv) API” on page 141
- “Read from Stdin (QtmhRdStin) API” on page 142
- “Write to Stdout (QtmhWrStout) API” on page 143
- “Convert to DB (QtmhCvtDB) API” on page 138
- “Parse QUERY\_STRING Environment Variable or Post stdin data (QzhhCgiParse) API” on page 144
- “Produce Full HTTP Response (QzhhCgiUtils) API” on page 152
- “Send or Save CGI Stateful Data (QzhhCgiSendState\_r) API” on page 152
- “Receive CGI Stateful Data (QzhhCgiRecvState\_r) API” on page 148
- “Put environment variable with CCSID (QzsrPutEnvCCSID) API” on page 149

- “Get environment variable with CCSID (QzsrGetEnvCCSID) API” on page 150

### Convert to DB (QtmhCvtDB) API:

The QtmhCvtDB() API provides an interface for CGI programs to parse CGI input, defined as a series of keywords and their values, into a buffer which is formatted according to a DDS file specification.

Required Parameter Group:

1 Qualified database file name	Input	Char(20)
2 Input string	Input	Char(*)
3 Length of input string	Input	Binary(4)
4 Response variable	Output	Char(*)
5 Length of response variable	Input	Binary(4)
6 Length of response available	Output	Binary(4)
7 Response code	Output	Binary(4)
8 Error Code	I/O	Char(*)

CGI input data, which comes to the CGI program as character data, will be converted by the QtmhCvtDB() API to the data type defined for the keyword by the corresponding field name in the input DDS file. Language statements, such as the ILE C #pragma mapinc statement, provide the ability to map the returned structure with field names defined in the DDS file. See the appropriate language user's guide for details.

**Note:** QtmhCvtDB() API is not allowed in CGI mode %%BINARY%%.

The following DDS field types are handled:

- **A** - Alphanumeric (see note 1 below)
- **P** - Packed Decimal (see note 2 below)
- **S** - Zoned Decimal
- **F** - Floating Point
- **T** - Time
- **L** - Date
- **Z** - Timestamp
- **B** - Binary (see note 3 below)
- **O** - DBCS

The following DDS field types are not handled:

- **H** - Hexadecimal (see note 4 below)
- **G** - Graphic
- **J** - DBCS
- **E** - DBCS

#### Notes:

1. The VARLEN keyword is not supported.
2. When using a packed decimal field, the #pragma mapinc() must use **\_P** the option, to create a packed structure.
3. Input to Binary fields is converted to integer. The DDS file specification must declare zero decimal positions (for example, “xB 0”, where x is 1-9).
4. ILE C converts hex DDS field data to character fields. Since the input stream to QtmhCvtDB() is a text string, the “hex” data would be converted from text to character fields. Therefore, using the **A** (Alphanumeric) field type to obtain the same conversion.



*Required parameter group:*

**Qualified database file name**

Input:CHAR(20)

The input variable containing the name of the database file defining field names and data types for the keywords anticipated in the input to the CGI program. Typically, the database file is generated using DDS to define the fields corresponding to the keywords anticipated in the CGI inputs. The first 10 characters contain the database file name, and the second 10 characters contain the library name.

**Input string**

INPUT:CHAR(\*)

The input variable containing the string of CGI input parameters to be parsed. When the environment variable REQUEST\_METHOD indicates that the method is GET, characters up to the first ? are ignored. The string must meet the format requirements for CGI input keyword strings.

**Length of input string**

INPUT:BINARY(4)

The input variable containing the length of the character string that contains the CGI input parameters to be parsed. The length of the string must be greater than 0.

**Response variable**

OUTPUT:CHAR(\*)

The output variable which is to contain the structure mapped according to the database file describing the input parameters anticipated by the CGI program.

**Length of response available**

INPUT:BINARY(4)

The input variable containing the total length of the buffer into which the CGI input parameters will be parsed.

**Length of response**

OUTPUT:BINARY(4)

The output variable that contains the length of the response. If the response variable is too small to contain the entire response, this parameter will be set to the size that is required to contain the entire response.

**Response code**

OUTPUT:BINARY(4)

A code that indicates the status of the request.

- **0** - All keywords have been translated according the database file.
- **-1** - The database file contains definitions for structure fields for which the CGI input has no corresponding keyword.
- **-2** - The CGI input contains one or more keywords for which the database file contains no corresponding field.
- **-3** - A combination of the condition for response codes -1 and -2 has been detected.
- **-4** - An error occurred while converting the CGI input string to the DDS defined data types. The data may or may not be usable.
- **-5** - This API is not valid when a program is not called by HTTP Server. No data parsing is done.
- **-6** - This API is not valid when operating in %%BINARY%% mode. No data parsing is done.

**Error Code**

I/O CHAR(\*)

The structure in which to return error information. For the format of the structure and for details on how to process API errors, see the API error reporting topic in the IBM i Information Center.

*Error messages:*

**CPF24B4 E**

Severe Error while addressing parameter list.

**CPF3C17 E**

Error occurred with input data parameter.

**CPF3C19 E**

Error occurred with receiver variable specified.

**CPF3CF1 E**

Error code parameter not valid.

**CPF9810 E**

Library &1 not found.

**CPF9812 E**

File &1 in library &2 not found.

**CPF9822 E**

Not authorized to file &1 in library &2

**Get Environment Variable (QtmhGetEnv) API:**

The QtmhGetEnv() API allows you to get the value set by the IBM HTTP Server for i server for a particular HTTP environment variable.

Required Parameter Group:

1 Receiver variable	Output	Char(*)
2 Length of receiver variable	Input	Binary(4)
3 Length of response	Output	Binary(4)
4 Request variable	Input	Char(*)
5 Length of request variable	Input	Binary(4)
6 Error Code	I/O	Char(*)

*Required parameter group:*

**Receiver variable**

OUTPUT:CHAR(\*)

The output variable that contains the value set by the server for the requested environment variable. In CGI input mode `%%MIXED%%`, this value will be in CCSID 37; otherwise, it will be in the CCSID of the current job. Note that the QUERY\_STRING in `%%BINARY%%` mode is not converted by the server.

**Length of receiver variable**

INPUT:BINARY(4)

The input variable containing the length of the space provided to receive the environment variable value.

**Length of response**

OUTPUT:BINARY(4)

The output variable that contains the length of the environment variable value. When the API is unable to determine the value for the requested environment variable, the length of the

environment variable value is set to zero. When the size required for the environment variable value is larger than the length of the receiver variable, the size required to receive the value is returned.

**Request variable**

INPUT:CHAR(\*)

The input variable containing the desired environment variable name.

**Length of request variable**

INPUT:BINARY(4)

The input variable containing the length (without trailing blanks) of the desired environment variable name.

**Error Code**

I/O:CHAR(\*)

The structure in which to return error information. For the format of the structure and for details on how to process API errors, see the API error reporting topic in the IBM i Information Center.

*Error messages:*

**CPF24B4 E**

Severe Error while addressing parameter list.

**CPF3C17 E**

Error occurred with input data parameter.

**CPF3C19 E**

Error occurred with receiver variable specified.

**CPF3CF1 E**

Error code parameter not valid.

**Note:** The Environment Variable APIs provide the `getenv()` (Get Value of Environment Variable) function necessary to retrieve environment variables in ILE C. Therefore, programs written in ILE C do not need to use the `QtmhGetEnv()` API. This API, for ILE C, is more difficult to use (and is slower) than the `getenv()` API on which it is based.

Programs that need CCSID conversion support for environment variables should use the Get environment variable with CCSID (`QzsrGetEnvCCSID`) API.

**Put Environment Variable (`QtmhPutEnv`) API:**

The `QtmhPutEnv()` API allows you to set or create a job-level environment variable.

Required Parameter Group:

1 Environment string	Input	Char(*)
2 Length of environment string	Input	Binary(4)
3 Error Code	I/O	Char(*)

*Required parameter group:*

**Environment string**

INPUT:CHAR(\*)

The input string of the form: `△envVar=value△`. Where `△envVar△` is the name of the new or existing environment variable, and `△value△` is the value you want to set the environment variable. Note that they are both case sensitive. The server expects this value to be in the CCSID of the job.

**Length of environment string**

INPUT: BINARY(4)

The input variable that contains the length of the environment string parameter (without trailing blanks). For example, the length of the environment string `envVar=value` is twelve.

**Error Code**

I/O: CHAR(\*)

The structure in which to return error information. For the format of the structure and for details on how to process API errors, see the API error reporting topic in the IBM i Information Center.

*Error messages:***CPF24B4 E**

Severe Error while addressing parameter list.

**CPF3021 E**

The value specified for the argument is not correct.

**CPF3C17 E**

Error occurred with input data parameter.

**CPF3CF1 E**

Error code parameter not valid.

**CPF3408 E**

The address used for an argument is not correct.

**CPF3460 E**

Storage allocation request failed.

**CPF3474 E**

Unknown system state.

**CPF3484 E**

A damaged object was encountered.

**Note:** The Environment Variable APIs provide the `putenv()` (Put Value in Environment Variable) function necessary to set (or create and set) an environment variable. Therefore, programs written in ILE C do not need to use the `QtmhPutEnv()` API. This API, for ILE C, is more difficult to use (and is slower) than the `putenv()` API on which it is based.

Programs that need CCSID conversion support for environment variables should use the `Put environment variable with CCSID (QzsrPutEnvCCSID)` API.

**Read from Stdin (QtmhRdStin) API:**

The `QtmhRdStin()` API allows CGI programs that are written in languages other than ILE C to read from stdin. CGI programs read from stdin when the request from the browser indicates the method that is POST. This API reads what the server has generated as input for the CGI program.

Required Parameter Group:

1 Receiver variable	Output	Char(*)
2 Length of receiver variable	Input	Binary(4)
3 Length of response available	Output	Binary(4)
4 Error Code	I/O	Char(*)

**Important:** CGI input data is only available from standard input when the client request is submitted with method POST. There are no standard input data when the method is GET or HEAD. In addition, the Content\_Length environment variable is set only when the Request\_Method is POST.

The program reads all of the data in a single request. This is because the API treats each request as a request for data starting at its beginning. The API handles each request as if it was the only request.

The length of the data returned by QtmhRdStin includes all the data from stdin. This includes line-formatting characters that are normally a part of the POST data as defined by the CGI specification.

Note that the format of this data is different depending on the CGI input mode being used. For %%MIXED%% mode, the data will have American National Standard Code for Information Interchange (ASCII) hexadecimal encoded characters. For %%EBCDIC%% mode, all data including hexadecimal will be in the CCSID of the job. The server performs no conversion for %%BINARY%% mode.

*Required parameter group:*

**Receiver variable**

OUTPUT:CHAR(\*)

The output variable that contains the data read from stdin. In CGI input mode %%MIXED%%, this data is in the CCSID of the job except that the encoded characters “%xx” are still represented by the ASCII 819 octet. In %%EBCDIC%% mode, this data is in the CCSID of the job, including the escape sequences. In %%BINARY%% mode, the data is in the code page sent by the browser.

**Length of receiver variable**

INPUT:BINARy(4)

The input variable containing the number of bytes that are to be read from stdin.

**Length or response available**

OUTPUT:BINARy(4)

The output variable containing the length of the data read from stdin. If there is no data available from stdin, this variable will be set to zero.

**Error Code**

I/O:Char(\*)

The structure in which to return error information. For the format of the structure and for details on how to process API errors, see the API error reporting topic in the IBM i Information Center.

*Error messages:*

**CPF24B4 E**

Severe Error while addressing parameter list.

**CPF3C17 E**

Error occurred with input data parameter.

**CPF3C19 E**

Error occurred with receiver variable specified.

**CPF3CF1 E**

Error code parameter not valid.

**Write to Stdout (QtmhWrStout) API:**

The QtmhWrStout() API provides the ability for CGI programs that are written in languages other than ILE C to write to stdout.

Required Parameter Group:

1 Data variable	Input	Char(*)
2 Length of data variable	Input	Binary(4)
3 Error Code	I/O	Char(*)

*Required parameter group:*

**Data variable**

Input:CHAR(\*)

The input variable containing the data to write to stdout.

**Length of data variable**

INPUT:BINARY(4)

The input variable contains the length of the data written to stdout. The length of the data must be larger than 0.

**Error Code**

I/O:CHAR(\*)

The structure in which to return error information. For the format of the structure and for details on how to process API errors, see the API error reporting topic in the IBM i Information Center.

*Error messages:*

**CPF24B4 E**

Severe Error while addressing parameter list.

**CPF3C17 E**

Error occurred with input data parameter.

**CPF3CF1 E**

Error code parameter not valid.

**Note:** CGI programs written in the ILE C language do not require a special API to write data to stdout. The following example shows how a CGI program might write to stdout:

```
fwrite(buffer,1,sizeof(buffer),stdout);
```

CGI programs are expected to produce data in the stdout that is formatted according to the CGI interface specification. The QtmhWrStout() API provides no line formatting; the user of the API must perform prescribed formatting which includes the requirement for text line characters (such as new line). Errors are not indicated for data that is not formatted per CGI requirements.

**Parse QUERY\_STRING Environment Variable or Post stdin data (QzhbCgiParse) API:**

The QzhbCgiParse() API allows you to parse the QUERY\_STRING environment variable, in the case of the GET method, or standard input, in the case of POST method, for CGI scripts. If the QUERY\_STRING environment variable is not set, the QzhbCgiParse() API reads the CONTENT\_LENGTH characters from its input. All return output is written to its standard output.

Required Parameter Group:

1 Command string	Input	Char(*)
2 Output format	Input	Char(*)
3 Target Buffer	Output	Char(*)
4 Length of Target Buffer	Input	Binary(4)
5 Length of response	Output	Binary(4)
6 Error Code	I/O	Char(*)

You can only call `QzhbCgiParse()` once for the POST method. To use this API with the POST method, you would first want to read all of `stdin` and assign it to the `QUERY_STRING` environment variable. You would then change the environment variable `REQUEST_METHOD` to `GET`.

This API does not work with the `%%MIXED%%` CGI input mode.

*Required parameter group:*

### Command string

Input:CHAR(\*)

The command string is a null ended string for flags and modifiers. At least one space must separate each flag. There is a one-character equivalent for each flag. The following flags are supported:

#### **-a[ain]** *continuation-handle*

The continuation-handle is the value returned to the caller in the target buffer when only partial information is returned. This flag is not valid on the first call to this API. It is used to retrieve the next set of information that would have been returned on a previous call if there had been enough space in the target buffer. All other flags must be the same as the previous call. Incomplete or inaccurate information may result if all other flags are not the same.

**Note:** This flag can only be used for the `CGI0200` format.

#### **-k[keywords]**

Parses `QUERY_STRING` for keywords. Keywords are decoded and written to the target buffer, one per line.

#### **-f[orm]**

Parses `QUERY_STRING` as form request. The field names will be set as environment variables with the prefix `FORM_`. Field values are the contents of the variables.

#### **-v[alue]** *field-name*

Parses `QUERY_STRING` as form request. Returns only the value of *field-name* in the target buffer.

**-r[ead]** Reads `CONTENT_LENGTH` characters from standard input and writes them to the target buffer.

**-i[nit]** If `QUERY_STRING` is not set, reads the value of standard input and returns a string that can be used to set `QUERY_STRING`.

#### **-s[ep]** *separator*

Specifies the string that is used to separate multiple values. If you are using the **-value** flag, the default separation is newline. If you are using the **-form** flag, the default separator is a comma (,).

#### **-p[refix]** *prefix*

Used with **-POST** and **-form** to specify the prefix to use when creating environment variable names. The default is `△FORM_△`.

#### **-c[ount]**

Used with **-keywords**, **-form**, and **-value**, returns a count of items in the target buffer that is related to these flags:

- **-keywords:** Returns the number of keywords.
- **-form:** Returns the number of unique fields (multiple values are counted as one).
- **-value *field-name*:** Returns the number of values for *field-name*. If there is no field that is named *field-name*, the output is 0.

**-number**

Used with **-keywords**, **-form**, and **-value**. Returns the specified occurrence in the target buffer related to the following flags:

- **-keywords**: Returns the n'th keyword. For example, **-2 -keywords** writes the second keyword.
- **-form**: Returns all the values of the n'th field.
- **-value *field-name***: Returns the n'th of the multiple values of field *field-name*.

**-Post** Information from standard input is directly decoded and parsed into values that can be used to set environment variables. This flag is the equivalent to consecutive use of the **-init** and **-form** options.

**-F[*sccsid*] *FileCCSID***

The *FileCCSID* is the name of the file system CCSID used in CCSID conversion when processing the CGI input data. The CGI program wants the data to be returned in this CCSID. It only applies when the server is using %%BINARY%% CGI conversion mode. When an unknown CCSID is set, the current value of the CGI\_EBCDIC\_CCSID environment variable is used.

**-N[*etccsid*] *NetCCSID***

The *NetCCSID* is the network CCSID used in CCSID conversion when processing the CGI input data. This is the CCSID that the data is presumed to be in at this time (as assumed or as set in a charset tag). It only applies when the server is using %%BINARY%% CGI Input mode. When an unknown CCSID is set, the current value of the CGI\_ASCII\_CCSID environment variable is used.

**Output format**

INPUT:CHAR(\*)

The format of the data to be returned in the target buffer. You must use one of the following format names:

- **CGII0100** - This format is the free-form format returned to standard output on other platforms.
- **CGII0200** - CGI form variable format. This format only applies to the **-form** and **-POST** option.

**Target Buffer**

OUTPUT:CHAR(\*)

This is output buffer that contains the information requested by the command string (if any).

**Length of Target Buffer**

INPUT:BINARy(4)

The length of the target buffer provided to receive the API output.

**Length of Response**

OUTPUT:BINARy(4)

The actual length of the information returned in the target buffer.

**Error Code**

I/O:CHAR(\*)

The structure in which to return error information. For the format of the structure and for details on how to process API errors, see the API error reporting topic in the IBM i Information Center.

*CGII0200 Format:*

Offset Decimal	Offset Hexadecimal	Type	Field
0	0	Binary(4)	Bytes returned
4	4	Binary(4)	Bytes available



Offset Decimal	Offset Hexadecimal	Type	Field
8	8	Char(20)	Continuation handle
28	1C	Binary(4)	Offset to first variable entry
32	20	Binary(4)	Number of variable entries returned
36	24	Char(*)	Reserved
		Binary(4)	Length of variable entry (see note below)
		Binary(4)	Length of variable name (see note below)
		Char(*)	Variable name (see note below)
		Binary(4)	Length of variable value (see note below)
		Char(*)	Variable value (see note below)
		Char(*)	Reserved (see note below)

**Note:** These fields contain variable entry information and are repeated for each variable entry returned.

*Field descriptions:*

**Bytes returned**

The number of bytes of data returned.

**Bytes available**

The number of bytes of data available to be returned. All available data is returned if enough space is available.

**Continuation handle**

The handle that is returned when more data is available to return, but the target buffer is not large enough. The handle indicates the point in the repository that the retrieval stopped. If the handle is used on the next call to the API (using the **-again** flag), the API returns more data starting at the point that the handle indicates. This field is set to blanks when all information is returned.

**Offset to first variable entry**

The offset to the first variable entry returned. The offset is from the beginning of the structure. If no entries are returned, the offset is set to zero.

**Number of variable entries returned**

The number of variable entries returned. If the target buffer is not large enough to hold the information, this number contains only the number of variables actually returned.

**Reserved**

This field is ignored.

**Length of variable entry**

The length of this variable entry. This value is used in determining the offset to the next variable entry. Note that this value is always set to a multiple of four.

**Length of variable name**

The length of the variable name for this entry.

**Variable name**

A field name as found in the form data. If the server is using `%%EBCDIC%%` or `%%MIXED%%` CGI mode, this value is in the CCSID of the job. If the server is using `%%BINARY%%` CGI mode,

this value is in the codepage as sent from the browser unless **-fscsid** is specified on the API invocation. If **-fscsid** is specified, the value is in that CCSID.

**Length of variable value**

The length of the variable value for this entry.

**Variable value**

A field name as found in the form data. If the server is using `%%EBCDIC%%` or `%%MIXED%%` CGI mode, this value is in the CCSID of the job. If the server is using `%%BINARY%%` CGI mode, this value is in the codepage as sent from the browser unless **-fscsid** is specified on the API invocation. If **-fscsid** is specified, the value is in that CCSID.

*Error messages:*

**CPF24B4 E**

Severe Error while addressing parameter list.

**CPF3C17 E**

Error occurred with input data parameter.

**CPF3C19 E**

Error occurred with receiver variable specified.

**CPF3CF1 E**

Error code parameter not valid.

**Note:** Error messages are added to the error log or script log except for those listed.

**Receive CGI Stateful Data (QzhbCgiRecvState\_r) API:**

The QzhbCgiRecvState\_r() API allows a high availability CGI programs to receive the CGI stateful data.

Required Parameter Group:

1 Target buffer	Output	Char(*)
2 Length of target buffer	Input	Binary(4)
3 Length of response	Output	Binary(4)
4 Continuation handle	I/O	Char(*)
5 Error Code	I/O	Char(*)

The HTTP Server receives the data for the next request to the stateful CGI so that if a failover occurs the data is available on the backup system (new primary system). This API is used with API QzhbCgiSendState\_r().

*Required parameter group:*

**Target buffer**

OUTPUT:CHAR(\*)

The target buffer containing the state of a high availability CGI program.

**Length of target buffer**

INPUT:BINARY(4)

The length of the target buffer that receives the API output. The minimum length is 1 byte and the maximum length is 61,000 bytes.

**Length of response**

OUTPUT:BINARY(4)

The length of response is the actual length of the information that is returned from the target buffer. If this value is greater than the length of the target buffer, then there is more state to read. The difference between these two values represents the amount of bytes the caller should read in subsequent calls to this API.

**Continuation handle**

I/O:CHAR(\*)

The continuation handle is the handle that is returned when more data is available to return, but the target buffer is not large enough. The caller must pass this handle to the QzhhbCgiRecvState\_r() API on subsequent calls as it was received from the previous call. On the first call to this API, the continuation handle must be set to 0 (equivalent to NULL in C). The caller must not allocate, deallocate, or modify the continuation handle. This field is set to 0 when all information is returned.

**Error code**

I/O:CHAR(\*)

The structure in which to return error information. For the format of the structure, see the API error reporting topic in the IBM i Information Center.

*Error messages:*

**CPF24B4 E**

Severe Error while addressing parameter list.

**CPF3C17 E**

Error occurred with input data parameter.

**CPF3CF1 E**

Error code parameter not valid.

**HTP4005 E**

Highly Available CGI invoked QzhhbCgiRecvState\_r() after it had already received the entire state.

**HTP4006 E**

QzhhbCgiRecvState\_r() was called when there was no state.

**Put environment variable with CCSID (QzsrPutEnvCCSID) API:**

The QzsrPutEnvCCSID() API allows a CGI program to set or create a job-level environment variable. CCSID support allows you to specify the encoding of the environment string.

Required parameter group:

Environment string	Input	Char(*)
Length of environment string	Input	Binary(4)
CCSID of environment string	Input	Binary(4)
Error code	I/O	Char(*)

*Required parameter group:*

**Environment string**

INPUT:CHAR(\*)

The input string of the form: *envVar=value*. Where *envVar* is the name of the new or existing environment variable, and *value* is the value you want to set the environment variable. They are both case sensitive. The QzsrPutEnvCCSID() API expects this value to be in the CCSID of the environment string.

**Length of environment string**

INPUT:BINARY(4)

The input variable that contains the length of the environment string parameter (without trailing blanks). For example, the length of the environment string envVar=value is twelve.

**CCSID environment string**

INPUT:BINARY(4)

The CCSID to be used for the encoding of the environment string. The valid values for this parameter are:

- 0 : The CCSID of the environment string.
- 1-65533 : A valid CCSID in this range must be specified or an error is returned.

**Error code**

I/O:CHAR(\*)

The structure in which to return error information. Error messages are added to the error log or script log except for those listed below. For the format of the structure and for details on how to process API errors, see the API error reporting topic in the IBM i Information Center.

*Error messages:*

**CPF24B4 E**

Severe error while addressing parameter list.

**CPF3CF1 E**

Error code parameter not valid.

**Note:** The Environment Variable APIs provide the putenv() (Put Value in Environment Variable) function necessary to set (or create and set) an environment variable. Programs that need CCSID conversion support for environment variables should use the QzsrPutEnvCCSID() API. See also "Get environment variable with CCSID (QzsrGetEnvCCSID) API."

**Get environment variable with CCSID (QzsrGetEnvCCSID) API:**

The QzsrvGetEnvCCSID() API allows a CGI program to get the value set by the server for a particular HTTP environment variable using CCSID support for input and output values.

Required parameter group:

Receiver variable	Output	Char(*)
Length of receiver variable	Input	Binary(4)
Length of response	Output	Binary(4)
Request variable	Input	Char(*)
Length of request variable	Input	Binary(4)
CCSID of request variable	Input	Binary(4)
CCSID of returned environment variable	Input	Binary(4)
Error Code	I/O	Char(*)

*Required parameter group:*

**Receiver variable**

OUTPUT:CHAR(\*)

The output variable that contains the value set by the server for the requested environment variable. This value will be returned in the CCSID specified for the returned environment variable.

The output variable that contains the value set by the server for the requested environment variable. This value will be returned in the CCSID specified for the returned environment variable.

**Length of receiver variable**

INPUT:BINARY(4)

The input variable containing the length of the space provided to receive the environment variable value.

**Length of response**

OUTPUT:BINARY(4)

The output variable that contains the length of the environment variable value. When the QzsrvGetEnvCCSID() API is unable to determine the value for the requested environment variable, the length of the environment variable value is set to zero. When the size required for the environment variable value is larger than the length of the receiver variable, the size required to receive the value is returned.

**Request variable**

INPUT:CHAR(\*)

The input variable containing the desired environment variable name.

**Length of request variable**

INPUT:BINARY(4)

The input variable containing the length (without trailing blanks) of the desired environment variable name.

**CCSID of request variable**

INPUT:BINARY(4)

The CCSID to be used for the encoding of the request variable. The valid values for this parameter are:

- 0 : The CCSID of the job.
- 1-65533 : A valid CCSID in this range must be specified or an error is returned.

**CCSID of returned environment variable**

INPUT:BINARY(4)

The CCSID to be used for the encoding of the returned environment variable. The valid values for this parameter are:

- 0 : The CCSID of the job.
- 1-65533 : A valid CCSID in this range must be specified or an error is returned.

**Error code**

I/O:CHAR(\*)

The structure in which to return error information. Error messages are added to the error log or script log except for those listed below. For the format of the structure and for details on how to process API errors, see the API error reporting topic in the IBM i Information Center.

*Error messages:*

**CPF24B4 E**

Severe error while addressing parameter list.

**CPF3CF1 E**

Error code parameter not valid.

**Note:** The Environment Variable APIs provide the getenv() (Get Value of Environment Variable) function necessary to retrieve environment variables in ILE C. Programs that need CCSID conversion support for environment variables should use the "Get environment variable with CCSID (QzsrvGetEnvCCSID) API" on page 150.

## Send or Save CGI Stateful Data (QzhhCgiSendState\_r) API:

The QzhhCgiSendState\_r() API allows a high availability CGI program to send or save CGI stateful data.

Required Parameter Group:

1	CGI's state string	Input	Char(*)
2	Length of the string	Input	Binary(4)
3	Error Code	I/O	Char(*)

The HTTP Server saves the data for the next request to the stateful CGI so that if a failover occurs the data is available on the backup system (new primary system).

*Required parameter group:*

### CGI's state string

INPUT:CHAR(\*)

The CGI's state string is the state of a high availability CGI that the Web server stores and passes to the CGI with the subsequent request. This string can consist of any information necessary for the CGI state (for example, a structure of several variables or fields). The Web server treats the contents of the state as binary data.

### Length of the string

INPUT:BINAR(4)

The length of the CGI's state. The minimum length is 1 byte and the maximum length is 61,000 bytes.

### Error code

I/O:CHAR(\*)

The structure in which to return error information. For the format of the structure, see the API error reporting topic in the IBM i Information Center.

*Error messages:*

### CPF24B4 E

Severe Error while addressing parameter list.

### CPF3C17 E

Error occurred with input data parameter.

### CPF3CF1 E

Error code parameter not valid.

## Produce Full HTTP Response (QzhhCgiUtils) API:

The QzhhCgiUtils() API allows a CGI program to produce a full HTTP 1.0/1.1 response for non-parsed header CGI programs. This API provides functionality similar to the **cgutils** command used by other HTTP server platforms.

Required Parameter Group:

1	Command string	Input	Char(*)
2	Error code	I/O	Char(*)

*Required parameter group:*

### Command string

INPUT:CHAR(\*)

The command string is a null ended string of flags and modifiers. Each flag must be separated by at least one space. The following flags are supported:

**-nodate**

Does not return the Date: header to the browser.

**-noel** Does not return a blank line after headers. This is useful if you want other MIME headers after the initial header lines.

**-status** *nnn*

Returns full HTTP response with status code *nnn*, instead of only a set of HTTP headers. Do not use this flag if you only want the Expires: header.

**-reason** *explanation*

Specifies the reason line for the HTTP response. You can only use this flag with the **-status** flag. If the explanation text contains more than one word, you must enclose it in parentheses.

**-ct** [*type/subtype*]

Specifies MIME Content-Type header to return to the browser. If you omit the *type/subtype*, the MIME content type is set to the default text/plan.

**-charset** *character-set*

Used with the **-ct** flag to specify the charset tag associated with the text Content-Types.

**-ce** *encoding*

Specifies MIME Content-Encoding header to return to the browser.

**-cl** *language-code*

Specifies MIME Content-Language header to return to the browser.

**-length** *nnn*

Specifies MIME Content-Length header to return to the browser.

**-expires** *Time-Spec*

Specifies MIME Expires header to return to the browser. This flag specifies the time to live in any combination of years, months, days, hours, minutes, and seconds. The time must be enclosed in parentheses. For example:

`-expires (2 days 12 hours)`

**-expires now**

Produces an Expires: header that matches the Date: header to return to the browser.

**-uri** *URI*

Specifies the Universal Resource Identifier (URI) for the returned document. URI can be considered the same as URL.

**-extra** *xxx: yyy*

Specifies an extra header that cannot otherwise be specified.

**Error Code**

I/O:CHAR(\*)

The structure in which to return error information. For the format of the structure and for details on how to process API errors, see the API error reporting topic in the IBM i Information Center.

*Error messages:*

**CPF24B4 E**

Severe Error while addressing parameter list.

**CPF3C17 E**

Error occurred with input data parameter.

## CPF3CF1 E

Error code parameter not valid.

## HTTP Server configuration APIs

This topic provides information about IBM HTTP Server for i configuration APIs, server instance APIs, and group APIs.

HTTP Server supports the APIs listed below in C++, REXX, ILE C, ILE COBOL, and ILE RPG programming languages.

### Configuration APIs

The configuration APIs are in \*SRVPGM QZHBCONF in library QHTTSPVR. ILE C programs must include header file QHTTSPVR/H(QZHBCONF). While each individual API lists its own authorities, the following authorities are needed to run all configuration APIs:

- \*OBJOPR, \*READ, \*ADD, and \*EXECUTE to the QUSRSYS library
- \*READ, \*ADD, \*DELETE, \*EXECUTE, \*OBJOPR, \*OBJEXIST, and either \*OBJMGT or \*OBJALTER to the QUSRSYS/QATMHTTTPC file
- \*READ, \*ADD, \*DELETE, \*EXECUTE, \*OBJOPR, \*OBJEXIST, and either \*OBJMGT or \*OBJALTER to the QUSRSYS/QATMHTTTPA file

**Note:** The QUSRSYS/QATMHTTTPA file is the administration (ADMIN) server configuration file.

- “Add Config Object (QzuiAddConfigObject) API” on page 168
- “Change Config Object Value (QzuiChangeConfigObject) API” on page 169
- “Close Apache Config File (QzuiCloseConfig) API” on page 172
- “Find Config Object (QzuiFindConfigObject) API” on page 175
- “Open Apache Config File (QzuiOpenConfig) API” on page 180
- “Remove Config Object (QzuiRemoveConfigObject) API” on page 181

### Server instance APIs

The server instance APIs are in \*SRVPGM QZHBCONF in library QHTTSPVR. ILE C programs must include header file QHTTSPVR/H(QZHBCONF). While each individual API lists its own authorities, the following authorities are needed to run all server instance APIs:

- \*OBJOPR, \*READ, \*ADD, and \*EXECUTE to the QUSRSYS library
- \*READ, \*ADD, \*DELETE, \*EXECUTE, \*OBJOPR, \*OBJEXIST, and either \*OBJMGT or \*OBJALTER to the QUSRSYS/QATMHINSTC file
- \*READ, \*ADD, \*DELETE, \*EXECUTE, \*OBJOPR, \*OBJEXIST, and either \*OBJMGT or \*OBJALTER to the QUSRSYS/QATMHINSTA file

**Note:** The QUSRSYS/QATMINSTA file is the administration (ADMIN) server instance file.

- “Change Apache Server Instance Data (QzuiChangeInstanceData) API” on page 170
- “Create Apache Server Instance (QzuiCreateInstance) API” on page 173
- “Get Apache Server Instance Data (QzuiGetInstanceData) API” on page 177
- “Get Server Instance Names (QzuiGetInstanceNames) API” on page 178
- “Get Instance Type (QzuiGetInstanceType) API” on page 179

### Group file APIs

The group file APIs are in \*SRVPGM QZHBCONF in library QHTTSPVR. ILE C programs must include header file QHTTSPVR/H(QZHBCONF).

- “Create a new Group File (QzhbCreateGroupList) API” on page 158



- “Read a Group File into Memory (QzhbOpenGroupList) API” on page 165
- “Free Group File from Memory (QzhbCloseGroupList) API” on page 157
- “Retrieve the next Group in the Group List (QzhbGetNextGroup) API” on page 162
- “Locate a named group in a Group List (QzhbFindGroupInList) API” on page 159
- “Retrieve the Name of a Group (QzhbGetGroupName) API” on page 161
- “Add a new Group to the end of a Group List (QzhbAddGroupToList) API”
- “Remove a Group from a Group List (QzhbRemoveGroupFromList) API” on page 166
- “Retrieve the next User in the Group (QzhbGetNextUser) API” on page 163
- “Locate a User in a Group (QzhbFindUserInGroup) API” on page 160
- “Retrieve the Name of a User (QzhbGetUserString) API” on page 164
- “Add a new user to the end of a Group (QzhbAddUserToGroup) API” on page 156
- “Remove a User or Element from a Group (QzhbRemoveUserFromGroup) API” on page 167

### Add a new Group to the end of a Group List (QzhbAddGroupToList) API:

In the IBM HTTP Server for i, use the QzhbAddGroupToList() API to add a new group to an in-memory group list.

Required Parameter Group:

1	grplist	Input	Binary(4)
2	group	Input	Char(*)
3	group_len	Input	Binary(4)
4	grp	Output	Binary(4)
5	errcode	I/O	Char(*)

Threadsafe: Yes

*Authorities and locks:* None.

*Required parameter group:*

**grplist** INPUT: BINARY(4)

The group list handle returned from a call to the QzhbCreateGroupList() or QzhbOpenGroupList() API.

**group** INPUT: CHAR(\*)

The group name to add to the list.

**group\_len**  
INPUT: BINARY(4)

The length of the group name. The length must be greater than or equal to 1.

**grp** OUTPUT: BINARY(4)

The handle of the newly created group, or the handle of an existing group if the named group already exists. Attempting to add a group that already exists is not considered an error by the system.

**errcode**  
I/O: CHAR(\*)

The structure in which to return error information.

*Error messages:*

**CPF3CF1 E**

Error code parameter not valid.

**CPF3C1D E**

Input variable length in parameter &1 not valid.

**HTPA001 E**

Input parameter &1 not valid.

**HTPA203 E**

Input group list handle in parameter &1 not valid.

**Add a new user to the end of a Group (QzhbAddUserToGroup) API:**

In the IBM HTTP Server for i, use the QzhbAddUserToGroup() API to add a new user to an in-memory group.

Required Parameter Group:

1	grplist	Input	Binary(4)
2	grp	Input	Binary(4)
3	user	Input	Char(*)
4	user_len	Input	Binary(4)
5	usr	Output	Binary(4)
6	errcode	I/O	Char(*)

Threadsafe: Yes

*Authorities and locks:* None.

*Required parameter group:*

**grplist** INPUT: BINARY(4)

The group list handle returned from a call to the QzhbCreateGroupList() or QzhbOpenGroupList() API.

**grp** INPUT: BINARY(4)

The group handle returned from a call to the QzhbGetNextGroup(), QzhbFindGroupInList(), or QzhbAddGroupToList() API.

**user** INPUT: CHAR(\*)

The user name to be added to the group.

**user\_len**

INPUT: BINARY(4)

The length of the user string. The length must be greater than or equal to 1.

**usr** OUTPUT: BINARY(4)

The handle of the newly created user, or the handle of an existing user if the named user already exists in the group. Attempting to add a user that already exists is not considered an error by the system.

**errcode**

I/O: CHAR(\*)

The structure in which to return error information.

*Error messages:*

**CPF3CF1 E**

Error code parameter not valid.

**CPF3C1D E**

Input variable length in parameter &1 not valid.

**HTPA001 E**

Input parameter &1 not valid.

**HTPA203 E**

Input group list handle in parameter &1 not valid.

**HTPA204 E**

Input group handle in parameter &1 not valid.

**Free Group File from Memory (QzhbCloseGroupList) API:**

In the IBM HTTP Server for i, use the QzhbCloseGroupList() API to free the memory of an in-memory copy of a group file. You can optionally write the in-memory version of the group list back to the group file before the memory is freed.

Required Parameter Group:

1	grplist	Input	Binary(4)
2	write	Input	Binary(4)
3	errcode	I/O	Char(*)

Threadsafe: Yes

*Authorities and locks:* None.

*Required parameter group:*

**grplist** INPUT: BINARY(4)

The group list handle returned from a call to the QzhbCreateGroupList() API or QzhbOpenGroupList() API.

**write** INPUT: BINARY(4)

The value of 0 (false) or a value of 1 (true), indicating whether or not to write the contents of the in-memory group list back to the group file before freeing it from memory. If you specify 1 for this parameter, and the write fails, the memory is not freed and the grplist handle is still valid.

**Note:** In order to specify a write value of 1, you must have previously used either the QzhbCreateConfigList() API or specified a writelock of 1 on the QzhbOpenGroupList() API. If these conditions are not met, the contents of the file are not written.

**errcode**

I/O: CHAR(\*)

The structure in which to return error information.

*Error messages:*

**CPF3CF1 E**

Error code parameter not valid.

**HTPA001 E**

Input parameter &1 not valid.

**HTPA202 E**

Unable to update group file &1.

## HTPA203 E

Input group list handle in parameter &1 not valid.

### Create a new Group File (QzhbCreateGroupList) API:

In the IBM HTTP Server for i, use the QzhbCreateGroupList() API to create a new empty group file, and return a handle to that empty in-memory version of the file.

Required Parameter Group:

1	path	Input	Binary(4)
2	path_len	Input	Binary(4)
3	grplist	Output	Binary(4)
4	errcode	I/O	Char(*)

Threadsafe: Yes

Normally this API would be followed by calls to the QzhbAddGroupToList() and QzhbAddUserToGroup() APIs, followed by the QzhbCloseGroupList() API to write group information out.

Upon successful completion of this API, a new group list handle is returned. This is a handle much like the one returned by the QzhbOpenGroupList() API against an already existing file, with a writelock argument of 1 (TRUE). After a call to the QzhbCreateGroupList() API the new file is left open for write access and the QzhbCloseGroupList() API can be invoked with a write argument of 1. For more details about the writelock argument, see "Read a Group File into Memory (QzhbOpenGroupList) API" on page 165.

*Authorities and locks:*

- \*X authority to each directory in the path of the specified group file
- \*WX authority to the last directory in the path that will contain the group file path

*Required parameter group:*

**path** INPUT: BINARY(4)

The path to the group file to be created in the Integrated File System. You can specify an absolute or relative path to the working directory. This path should be in the job CCSID.

**path\_len**  
INPUT: BINARY(4)

The length of the path string.

**grplist** OUTPUT: BINARY(4)

The variable that receives the integer handle of the newly created empty group list. Subsequent API calls use this handle.

**errcode**  
I/O: CHAR(\*)

The structure in which to return error information.

*Error messages:*

**CPF3CF1 E**  
Error code parameter not valid.

**CPF3C1D E**  
Input variable length in parameter &1 not valid.

**HTPA001 E**

Input parameter &1 not valid.

**HTPA202 E**

Unable to update group file &1.

**HTPA208 E**

Group file &1 already exists.

**Locate a named group in a Group List (QzhbFindGroupInList) API:**

In the IBM HTTP Server for i, use the QzhbFindGroupInList() API to search an in-memory group list for a named group.

Required Parameter Group:

1	grplist	Input	Binary(4)
2	group	Input	Binary(4)
3	group_len	Input	Binary(4)
4	grp	Output	Binary(4)
5	errcode	I/O	Char(*)

Threadsafe: Yes

*Authorities and locks:* None.

*Required parameter group:*

**grplist** INPUT: BINARY(4)

The group list handle returned from a call to the QzhbCreateGroupList() or QzhbOpenGroupList() API.

**group** INPUT: CHAR(\*)

The group name for which the system will search the list. The group name is case-sensitive. Leading and trailing blanks are included with the name.

**group\_len**

INPUT: BINARY(4)

The length of the group name string. The length must be greater than or equal to 1.

**grp** OUTPUT: BINARY(4)

The group name handle returned if the named group was found in the list.

**errcode**

I/O: CHAR(\*)

The structure in which to return error information.

*Error messages:*

**CPF3CF1 E**

Error code parameter not valid.

**CPF3C1D E**

Input variable length in parameter &1 not valid.

**HTPA001 E**

Input parameter &1 not valid.

**HTPA203 E**

Input group list handle in parameter &1 not valid.

## HTPA206 E

Group file &1 not found in group list.

### Locate a User in a Group (QzhbFindUserInGroup) API:

In the IBM HTTP Server for i, use the QzhbFindUserInGroup() API to search an in-memory group for a specific user.

Required Parameter Group:

1	grplist	Input	Binary(4)
2	grp	Input	Binary(4)
3	user	Input	Char(*)
4	user_len	Input	Binary(4)
5	usr	Output	Binary(4)
6	errcode	I/O	Char(*)

Threadsafe: Yes

*Authorities and locks:* None.

*Required parameter group:*

**grplist** INPUT: BINARY(4)

The group list handle returned from a call to the QzhbCreateGroupList() or QzhbOpenGroupList() API.

**grp** INPUT: BINARY(4)

The group handle returned from a call to the QzhbGetNextGroup(), QzhbFindGroupInList(), or QzhbAddGroupToList() API.

**user** INPUT: CHAR(\*)

The user name for which the system will search the group. The user name is case-sensitive. Leading and trailing blanks are included with the name.

**user\_len**

INPUT: BINARY(4)

The length of the user string. The length must be greater than or equal to 1.

**usr** OUTPUT: BINARY(4)

The handle of the user if it was found in the group.

**errcode**

I/O: CHAR(\*)

The structure in which to return error information.

*Error messages:*

## CPF3CF1 E

Error code parameter not valid.

## CPF3C1D E

Input variable length in parameter &1 not valid.

## HTPA001 E

Input parameter &1 not valid.

## HTPA203 E

Input group list handle in parameter &1 not valid.

**HTPA204 E**

Input group handle in parameter &1 not valid.

**HTPA207 E**

User &1 not found in group.

**Retrieve the Name of a Group (QzhbGetGroupName) API:**

In the IBM HTTP Server for i, use the QzhbGetGroupName() API to retrieve the name of a group using the group handle.

Required Parameter Group:

1	grplist	Input	Binary(4)
2	grp	Input	Binary(4)
3	buf	Output	Char(*)
4	buf_len	Input	Binary(4)
5	buf_actlen	Output	Binary(4)
6	errcode	I/O	Char(*)

Threadsafe: Yes

*Authorities and locks:* None.

*Required parameter group:*

**grplist** INPUT: BINARY(4)

The group list handle returned from a call to the QzhbCreateGroupList() or QzhbOpenGroupList() API.

**grp** INPUT: BINARY(4)

The group handle returned from a call to the QzhbGetNextGroup(), QzhbFindGroupInList(), or QzhbAddGroupToList() API.

**buf** OUTPUT: BINARY(4)

The buffer to receive the group name.

**buf\_len**

OUTPUT: BINARY(4)

The size of the buffer.

**buf\_actlen**

OUTPUT: BINARY(4)

The actual length of the group name. If the buf\_actlen value is greater than the buf\_len value, the data is truncated.

**errcode**

I/O: CHAR(\*)

The structure in which to return error information.

*Error messages:*

**CPF3CF1 E**

Error code parameter not valid.

**CPF3C1D E**

Input variable length in parameter &1 not valid.

**HTPA001 E**

Input parameter &1 not valid.

**HTPA203 E**

Input group list handle in parameter &1 not valid.

**HTPA204 E**

Input group handle in parameter &1 not valid.

**Retrieve the next Group in the Group List (QzhbGetNextGroup) API:**

In the IBM HTTP Server for i, use the QzhbGetNextGroup API to retrieve the next group from an in-memory group list.

Required Parameter Group:

1	grplist	Input	Binary(4)
2	prev_grp	Input	Binary(4)
3	grp	Output	Binary(4)
4	errcode	I/O	Char(*)

Threadsafe: Yes

*Authorities and locks:* None.

*Required parameter group:*

**grplist** INPUT: BINARY(4)

The group list handle returned from a call to the QzhbCreateGroupList() or QzhbOpenGroupList() API.

**prev\_grp**

INPUT: BINARY(4)

The group handle returned from a call to the QzhbGetNextGroup(), QzhbGetNextGroup(), QzhbFindGroupInList(), or QzhbAddGroupToList() API, that returns the group immediately following this group. A handle of 0 returns the first group in the group list.

**grp**

OUTPUT: BINARY(4)

The group name handle returned if the next group is found in the list. If no next group exists, then error HTPA206 is returned.

**errcode**

I/O: CHAR(\*)

The structure in which to return error information.

*Error messages:*

**CPF3CF1 E**

Error code parameter not valid.

**HTPA001 E**

Input parameter &1 not valid.

**HTPA203 E**

Input group list handle in parameter &1 not valid.

**HTPA204 E**

Input group handle in parameter &1 not valid.

**HTPA206 E**

Group file &1 not found in group list.



## Retrieve the next User in the Group (QzhbGetNextUser) API:

In the IBM HTTP Server for i, use the QzhbGetNextUser() API to retrieve the next user from a group.

Required Parameter Group:

1	grplist	Input	Binary(4)
2	grp	Input	Binary(4)
3	prev_usr	Input	Binary(4)
4	usr	Output	Binary(4)
5	errcode	I/O	Char(*)

Threadsafe: Yes

*Authorities and locks:* None.

*Required parameter group:*

**grplist** INPUT: BINARY(4)

The group list handle returned from a call to the QzhbCreateGroupList() or QzhbOpenGroupList() API.

**grp** INPUT: BINARY(4)

The group handle returned from a call to the QzhbGetNextGroup(), QzhbFindGroupInList(), or QzhbAddGroupToList() API.

**prev\_usr**

INPUT: BINARY(4)

The user handle for an existing user that returns the user immediately following this user. A handle of 0 returns the first user in the group list.

**usr** OUTPUT: BINARY(4)

The handle of the user if a next user is found in the group. If no next user is found, error HTPA207 is returned.

**errcode**

I/O: CHAR(\*)

The structure in which to return error information.

*Error messages:*

**CPF3CF1 E**

Error code parameter not valid.

**HTPA001 E**

Input parameter &1 not valid.

**HTPA203 E**

Input group list handle in parameter &1 not valid.

**HTPA204 E**

Input group handle in parameter &1 not valid.

**HTPA205 E**

Input user handle in parameter &1 not valid.

**HTPA207 E**

User &1 not found in group.

## Retrieve the Name of a User (QzhbGetUserString) API:

In the IBM HTTP Server for i, use the QzhbGetUserString() API to retrieve the name string of a group member given the user handle, as returned by the QzhbGetNextUser(), QzhbFindUserInGroup(), or QzhbAddUserToGroup() API.

Required Parameter Group:

1	grplist	Input	Binary(4)
2	grp	Input	Binary(4)
3	usr	Input	Binary(4)
4	buf	Output	Char(*)
5	buf_len	Input	Binary(4)
6	buf_actlen	Output	Binary(4)
7	errcode	I/O	Char(*)

Threadsafe: Yes

*Authorities and locks:* None.

*Required parameter group:*

**grplist** INPUT: BINARY(4)

The group list handle returned from a call to the QzhbCreateGroupList() or QzhbOpenGroupList() API.

**grp** INPUT: BINARY(4)

The group handle returned from a call to the QzhbGetNextGroup(), QzhbFindGroupInList(), or QzhbAddGroupToList() API.

**usr** INPUT: BINARY(4)

The user handle returned from a call to the QzhbGetNextUser(), QzhbFindUserInGroup(), or QzhbAddUserToGroup() API.

**buf** OUTPUT: CHAR(\*)

The buffer to receive the user string.

**buf\_len**

INPUT: BINARY(4)

The size of the buffer.

**buf\_actlen**

OUTPUT: BINARY(4)

The actual length of the user string. If the buf\_actlen value is greater than the buf\_len value, the data is truncated by the system.

**errcode**

I/O: CHAR(\*)

The structure in which to return error information.

*Error messages:*

**CPF3CF1 E**

Error code parameter not valid.

**CPF3C1D E**

Input variable length in parameter &1 not valid.

**HTPA001 E**

Input parameter &1 not valid.

**HTPA203 E**

Input group list handle in parameter &1 not valid.

**HTPA204 E**

Input group handle in parameter &1 not valid.

**HTPA205 E**

Input group handle in parameter &1 not valid.

**Read a Group File into Memory (QzhbOpenGroupList) API:**

In the IBM HTTP Server for i, use the QzhbOpenGroupList() API to read in an existing group file, and return a handle to an in-memory version of the file.

Required Parameter Group:

1 path	Input	Binary(4)
2 path_len	Input	Binary(4)
3 writelock	Input	Binary(4)
4 grplist	Output	Binary(4)
5 errcode	I/O	Char(*)

Threadsafe: Yes

See "Free Group File from Memory (QzhbCloseGroupList) API" on page 157 for information about freeing memory and optionally writing the group list information out.

*Authorities and locks:*

- \*X authority to each directory in the path of the specified group file
- \*R authority to the group file for a writelock value of 0
- \*RW authority to the group file for a writelock value of 1

*Required parameter group:*

**path** INPUT: BINARY(4)

The path to the group file to be created in the integrated file system. You can specify an absolute or relative path to the working directory.

**path\_len**

INPUT: BINARY(4)

The length of the path string.

**writelock**

INPUT: BINARY(4)

If the value is 1, the group file is opened for write access with a lock and kept open. No other user is allowed to update the group file while the lock is in place. The group file is closed and the lock released by invoking the QZHBcloseGroupList() API. If the value is 0, then the following are true:

- The group file is opened for read access
- A lock is not placed on the group file
- The group file does not remain open

**Note:** You must specify a writelock of 1 in order to later specify a write argument of 1 on the QzhbCloseGroupList() API. If you do not hold the group file open for write, the QzhbCloseGroupList() API will not write the contents of the file.

**grplist** OUTPUT: BINARY(4)

The handle of the group list. Subsequent API calls use this handle.

**errcode**

I/O: CHAR(\*)

The structure in which to return error information.

*Error messages:*

**CPF3CF1 E**

Error code parameter not valid.

**CPF3C1D E**

Input variable length in parameter &1 not valid.

**HTPA001 E**

Input parameter &1 not valid.

**HTPA201 E**

Group file &1 not found or is unreadable.

**HTPA202 E**

Unable to update group file &1.

### **Remove a Group from a Group List (QzhbRemoveGroupFromList) API:**

In the IBM HTTP Server for i, use the QzhbRemoveGroupFromList() API to remove a named group, and all the users in that group, from an in-memory group list.

Required Parameter Group:

1	grplist	Input	Binary(4)
2	grp	Input	Binary(4)
3	errcode	I/O	Char(*)

Threadsafe: Yes

*Authorities and locks:* None.

*Required parameter group:*

**grplist** INPUT: BINARY(4)

The group handle returned from a call to the QzhbCreateGroupList() or QzhbOpenGroupList() API.

**grp** INPUT: BINARY(4)

The group handle returned from a call to the QzhbGetNextGroup(), QzhbFindGroupInList(), or QzhbAddGroupToList() API.

**errcode**

I/O: CHAR(\*)

The structure in which to return error information.

*Error messages:*

**CPF3CF1 E**

Error code parameter not valid.

**HTPA001 E**

Input parameter &1 not valid.

**HTPA203 E**

Input group list handle in parameter &1 not valid.

**HTPA204 E**

Input group handle in parameter &1 not valid.

**Remove a User or Element from a Group (QzhbRemoveUserFromGroup) API:**

In the IBM HTTP Server for i, use the QzhbRemoveUserFromGroup() API to remove a user from an in-memory group.

Required Parameter Group:

1	grplist	Input	Binary(4)
2	grp	Input	Binary(4)
3	usr	Input	Binary(4)
4	errcode	I/O	Char(*)

Threadsafe: Yes

*Authorities and locks:* None.

*Required parameter group:*

**grplist** INPUT: BINARY(4)

The group list handle returned from a call to the QzhbCreateGroupList() or QzhbOpenGroupList() API.

**grp** INPUT: BINARY(4)

The group handle returned from a call to the QzhbGetNextGroup(), QzhbFindGroupInList(), or QzhbAddGroupToList() API.

**usr** INPUT: BINARY(4)

The user handle returned from a call to the QzhbGetNextUser(), QzhbFindUserInGroup(), or QzhbAddUserToGroup() API.

**errcode**

I/O: CHAR(\*)

The structure in which to return error information.

*Error messages:*

**CPF3CF1 E**

Error code parameter not valid.

**HTPA001 E**

Input parameter &1 not valid.

**HTPA203 E**

Input group list handle in parameter &1 not valid.

**HTPA204 E**

Input group handle in parameter &1 not valid.

**HTPA205 E**

Input user handle in parameter &1 not valid.

## Add Config Object (QzuiAddConfigObject) API:

In the IBM HTTP Server for i, use the QzuiAddConfigObject() API to add scope or directive to the configuration. It may be placed relative to a directive or scope, at the end or beginning of the file, or at the beginning or end of a scope. A handle to the object is returned allowing directives to be added to it.

Required Parameter Group:

1	cfg	Input	Binary(4)
2	obj_type	Input	Binary(4)
3	key	Input	Char(*)
4	key_size	Input	Binary(4)
5	val	Input	Char(*)
6	val_size	Input	Binary(4)
7	place	Input	Binary(4)
8	where	Input	Binary(4)
9	object	Output	Binary(4)
10	errcode	I/O	Char(*)

Threadsafe: Yes

*Authorities and locks:* None.

*Required parameter group:*

**cfg** INPUT: BINARY(4)

Handle to the config.

**obj\_type**

INPUT: BINARY(4)

Type of object to add (0 = directive, 1 = scope).

**key** INPUT: CHAR(\*)

Keyword of scope or directive to add.

**key\_size**

INPUT: BINARY(4)

Size of key passed.

**val** INPUT: CHAR(\*)

Value for scope.

**val\_size**

INPUT: BINARY(4)

Size of value.

**place** INPUT: BINARY(4)

Placement directive (0 = at the end of the file, 1 = at start of file, 2 = after  $\Delta$ where $\Delta$ , 3 = before  $\Delta$ where $\Delta$ , 4 = at start of scope specified by  $\Delta$ where $\Delta$ , 5 = at end of scope specified by  $\Delta$ where $\Delta$ ).

**where** INPUT: BINARY(4)

Optional handle to scope or directive for scope placement.

**object** OUTPUT: BINARY(4)

Handle of the object added.

**errcode**

I/O: CHAR(\*)

Error information structure.

*Error messages:*

**CPF3C17 E**

Error occurred with input data parameter.

**CPF3C19 E**

Error occurred with receiver variable specified.

**CPF3C1D E**

Input variable length in parameter &1 not valid.

**CPF3CF1 E**

Error code parameter not valid.

**HTPA001 E**

Input parameter &1 not valid.

**HTPA106 E**

Input configuration handle not valid.

**HTPA121 E**

Object handle in parameter &1 not valid.

**HTPA122 E**

Object handle in parameter &1 not a scope.

**HTPA124 E**

Combination of insertion position and relative object not valid.

**HTPA126 E**

Keyword &1 not valid for object type.

**Change Config Object Value (QzuiChangeConfigObject) API:**

In the IBM HTTP Server for i, use the QzuiChangeConfigObject() API to change the value portion of a scope or directive.

Required Parameter Group:

1	cfg	Input	Binary(4)
2	object	Input	Binary(4)
3	value	Input	Char(*)
4	value_size	Input	Binary(4)
5	errcode	I/O	Char(*)

Threadsafe: Yes

The value is considered anything after the keyword. For example, in the directive  $\Delta$ BrowserMatch Mozilla/2 nokeepalive $\Delta$ , the keyword is  $\Delta$ BrowserMatch $\Delta$  and the value is  $\Delta$ Mozilla/2 nokeepalive $\Delta$ .

*Authorities and locks:* None.

*Required parameter group:*

**cfg** INPUT: BINARY(4)

Handle to the config.

**object** INPUT: BINARY(4)

Handle to the scope or directive to be changed.

**value** INPUT: CHAR(\*)

New value for the object.

**value\_size**

INPUT:BINARY(4)

Size of value.

**errcode**

I/O:CHAR(\*)

Error information structure.

*Error messages:*

**CPF3C17 E**

Error occurred with input data parameter.

**CPF3CF1 E**

Error code parameter not valid.

**HTPA001 E**

Input parameter &1 not valid.

**HTPA106 E**

Input configuration handle not valid.

**HTPA121 E**

Object handle in parameter &1 not valid.

**HTPA125 E**

Value &1 not valid for keyword &2.

**Change Apache Server Instance Data (QzuiChangeInstanceData) API:**

In the IBM HTTP Server for i, use the QzuiChangeInstanceData() API to change the information contained in the instance file. The information is retrieved in the format specified by INSD0110.

Required Parameter Group:

1 name	Input	Char(10)
2 idata	Input	Char(*)
3 idata_size	Input	Binary(4)
4 format	Input	Char(8)
5 errcode	I/O	Char(*)

Threadsafe: Yes

*Authorities and locks:*

- \*EXECUTE authority to the QUSRSYS library
- \*OBJOPR, \*OBJMGT, \*ADD, and \*DLT authority to the QUSRSYS/QATMHINSTC file

*Required parameter group:*

**name** INPUT:CHAR(10)

Name of the server instance from which data is retrieved.

**idata** INPUT:CHAR(\*)

Buffer in format INSD0110 containing instance file data.

**idata\_size**

INPUT:BINARY(4)

Length of instance data passed.



**format**

INPUT:CHAR(8)

Format of the instance data (INSD0110).

**errcode**

I/O:CHAR(\*)

Error information structure.

*INSD0110 format:* This data format is used by the QzuiCreateInstance(), QzuiGetInstanceData(), and QzuiChangeInstanceData() APIs.

Offset	Type	Field
0	Char(10)	Autostart
12	Binary(4)	Threads
16	Binary(4)	CCSID
20	Char(10)	Outgoing table name
30	Char(10)	Outgoing table library
40	Char(10)	Incoming table name
50	Char(10)	Incoming table library
60	Char(512)	Config file (full path)
572	Char(512)	Server root path

*Field description:*

**Note:** In the descriptions below, \*GLOBAL indicates that the global server parameter value for this field is used by the instance, and \*CFG indicates that the value from the named configuration file is used. All character strings are padded with blanks as necessary, and are NOT null terminated.

**Autostart**

Indicates if the instance starts automatically. It is a 10 character string that contains \*NO, \*YES, or \*GLOBAL.

**Threads**

The number of threads to use for this instance. It is an integer from 0 to 999, where 0 means the \*CFG value.

**CCSID**

The character set to be used by the instance. It is an integer from 0 to 65533, where 0 means \*GLOBAL.

**Outgoing table name**

The name of the table object to use as the EBCDIC to ASCII conversion table for outgoing data. It is a 10 character name or \*GLOBAL.

**Outgoing table library**

The library containing the EBCDIC to ASCII table. This field is blank if the outgoing table name is \*GLOBAL.

**Incoming table name**

The name of the table object to use as the ASCII to EBCDIC conversion table for incoming data. It is a 10 character name or \*GLOBAL.

**Incoming table library**

The library containing the ASCII to EBCDIC table. This field is blank if the incoming table name is \*GLOBAL.

**Config file (full path)**

The path to the server instance configuration file.

**Server root path**

The path to the server root.

*Error messages:***CPF3C17 E**

Error occurred with input data parameter.

**CPF3C1D E**

Input variable length in parameter &1 not valid. CPF3C21 E

**CPF3C21 E**

Format name &1 not valid.

**CPF3CF1 E**

Error code parameter not valid.

**CPF9822 E**

Not authorized to file &1 in library &2.

**CPFB602 E**

Cannot open file.

**HTPA001 E**

Input parameter &1 not valid.

**HTPA101 E**

Server instance &1 not found or is unreadable.

**HTPA102 E**

Unable to update server instance &1.

**HTPA103 E**

Value in field &1 of the instance data structure not valid.

**HTPA127 E**

Server instance &1 is not a HTTP Server type instance.

**Close Apache Config File (QzuiCloseConfig) API:**

In the IBM HTTP Server for i, use the QzuiCloseConfig() API to optionally write the memory copy of the configuration out to the file and then free the memory copy. If the file name is specified, the configuration is written to that file, otherwise it is written to the original file.

**Required Parameter Group:**

1	cfg	Input	Binary(4)
2	write	Input	Binary(4)
3	fname	Input	Char(*)
4	fname_size	Input	Binary(4)
5	errcode	I/O	Char(*)

Threadsafe: Yes

*Authorities and locks:*

- If the file is closed without write, no authority is needed
- \*X authority to each directory in the path of the specified group file
- \*RW authority to the group file for a writelock value of 1

*Required parameter group:*

- cfg** INPUT: BINARY(4)  
Handle to the config to be closed.
- write** INPUT: BINARY(4)  
Has the following values: 0 = no write, 1 = write.
- fname** INPUT: CHAR(\*)  
Path and name of config file to be written (optional).
- fname\_size**  
INPUT: BINARY(4)  
Length of file name ( 0 for no file name).
- errcode**  
I/O: CHAR(\*)  
Error information structure.

*Error messages:*

- CPF3C17 E**  
Error occurred with input data parameter.
- CPF3C1D E**  
Input variable length in a parameter &1 not valid.
- CPF3CF1 E**  
Error code parameter not valid.
- HTPA001 E**  
Input parameter &1 not valid.
- HTPA120 E**  
Unable to update server configuration &1.

### **Create Apache Server Instance (QzuiCreateInstance) API:**

The QzuiCreateInstance() API allows users to create a new IBM HTTP Server for i server instance.

Required Parameter Group:

1 name	Input	Char(10)
2 idata	Input	Char(*)
3 idata_size	Input	Binary(4)
4 format	Input	Char(8)
5 errcode	I/O	Char(*)

Threadsafe: Yes

*Authorities and locks:*

- \*EXECUTE and \*ADD authority to the QUSRSYS library
- \*OBJOPR, \*ADD, \*DLT, and either \*OBJMGT or \*OBJALTER authority to the QUSRSYS/QATMHINSTC file

*Required parameter group:*

- name** INPUT: CHAR(10)  
The name of the instance to be created.

**idata** INPUT:CHAR(\*)

The instance data.

**idata\_size**

INPUT:BINARY(4)

The length of the instance data.

**format**

INPUT:CHAR(8)

The format of the instance data (INSD0110).

See "INSD0110 format" on page 171 for more information.

**errcode**

I/O:CHAR(\*)

The error information structure.

*Error messages:*

**CPF3C17 E**

Error occurred with input data parameter.

**CPF3C19 E**

Error occurred with receiver variable specified.

**CPF3CF1 E**

Error code parameter not valid.

**CPF9822 E**

Not authorized to file &1 in library &2.

**HTPA001 E**

Input parameter &1 not valid.

**HTPA102 E**

Unable to update server instance &1.

**HTPA103 E**

Value in field &1 of the instance data structure not valid.

### Delete a Server Instance (QzuiDeleteInstance) API:

The QzuiDeleteInstance() API allows you to delete an IBM HTTP Server for i server instance.

Required Parameter Group:

1 name	Input	Char(10)
2 errcode	I/O	Char(*)

Threadsafe: Yes

*Authorities and locks:*

- \*EXECUTE authority to the QUSRSYS library
- \*OBJOPR, \*OBJEXIST, \*DLT and either \*OBJMGT or \*OBJALTER authority to the QUSRSYS/QATMHINSTC file

*Required parameter group:*

**name** INPUT:CHAR(10)

The server instance name you want to delete. The name can be up to 10 characters long (padded with blanks).

**errcode**

I/O:CHAR(\*)

The structure in which to return error information.

*Error messages:***CPF3CF1 E**

Error code parameter not valid.

**HTPA001 E**

Input parameter &amp;1 not valid.

**HTPA101 E**

Server instance &amp;1 not found or is unreadable.

**HTPA102 E**

Unable to update server instance &amp;1.

**CPF9802 E**

Not authorized to object &amp;2 &amp;3.

**Find Config Object (QzuiFindConfigObject) API:**

The QzuiFindConfigObject() API allows you to search an IBM HTTP Server for i configuration file for the object (and possibly value) specified and returns a handle to it.

Required Parameter Group:

1	cfg	Input	Binary(4)
2	fdata	Input	Char(*)
3	fdata_size	Input	Binary(4)
4	format	Input	Char(8)
5	object	Output	Binary(4)
6	val	Output	Char(*)
7	val_size	Input	Binary(4)
8	val_actlen	Output	Binary(4)
9	errcode	I/O	Char(*)

Threadsafe: Yes

If  $\Delta$ start $\Delta$  is not specified, the configuration file scope is used. If a value is specified, the value is tokenized and compared with the tokens of the matching keywords. For example, if the keyword is  $\Delta$ BrowserMatch $\Delta$  and the value is  $\Delta$ Mozilla/2 $\Delta$  the search would find  $\Delta$ BrowserMatch Mozilla/2 nokeepalive $\Delta$ . Also, the  $\Delta$ val $\Delta$  field would contain  $\Delta$ Mozilla/2 nokeepalive $\Delta$ . You need only pass the object type and keyword. For example, to find a  $\Delta$ Port $\Delta$  directive, set the object type to 1, the keyword to  $\Delta$ Port $\Delta$  and fdata\_size to 8. If the value field is not needed, set  $\Delta$ val\_size $\Delta$  to 0.

*Authorities and locks:* None.*Required parameter group:***cfg** INPUT: BINARY(4)

Handle to the configuration file.

**fdata** INPUT: CHAR(\*)

Find data in format CFGF0110.

**fdata\_size**

INPUT: BINARY(4)

Size of Find data format buffer.

**format**

INPUT:CHAR(8)

Name of format (CFGF0110).

**object** OUTPUT:BINARY(4)

Handle to the object found (-1 indicates not found).

**val** OUTPUT:CHAR(\*)

Contains the whole value of the configuration object found.

**val\_size**

INPUT:BINARY(4)

Size of value buffer.

**val\_actlen**

OUTPUT:BINARY(4)

Actual size of value.

**errcode**

I/O:CHAR(\*)

Error information structure.

*CFGF0110 format:* This data format is used by QzuiFindConfigObject() API.

Offset	Type	Field
0	Binary(4)	Object type (0=directive, 1=scope)
4	Char(40)	Keyword object to search for (required)
44	Binary(4)	Case sensitive (0=insensitive, 1=sensitive)
48	Binary(4)	Where to search (0=entire configuration, 1=with scope specified in "Start", 2=start search at object specified in "Start", 3=start search at scope specified in object start)
52	Binary(4)	Start - the search start handle (0 if no start is to be used)
56	Binary(4)	Value when searching (0=no, 1=yes)
60	Char(100)	Value of object to search for

*Error messages:***CPF3C17 E**

Error occurred with input data parameter.

**CPF3C19 E**

Error occurred with receiver variable specified.

**CPF3C1D E**

Input variable length in parameter &amp;1 not valid.

**CPF3C21 E**

Format name &amp;1 not valid.

**CPF3CF1 E**

Error code parameter not valid.

**HTPA001 E**

Input parameter &1 not valid.

**HTPA106 E**

Input configuration handle not valid.

**HTPA121 E**

Object handle in parameter &1 not valid.

**HTPA122 E**

Object handle in parameter &1 not a scope.

**HTPA123 E**

No matching object found.

**Get Apache Server Instance Data (QzuiGetInstanceData) API:**

The QzuiGetInstanceData() API allows you to retrieve configuration data from a specified IBM HTTP Server for i server instance file.

Required Parameter Group:

1	name	Input	Char(10)
2	buf	Output	Char(*)
3	buf_size	Input	Binary(4)
4	format	Input	Char(8)
5	buf_actlen	Output	Binary(4)
6	running	Output	Binary(4)
7	errcode	I/O	Char(*)

Threadsafe: Yes

The data information is returned in the format specified by INSD0110. See "INSD0110 format" on page 171 for more information.

*Authorities and locks:*

- \*EXECUTE authority to the QUSRSYS library
- \*OBJOPR and \*READ authority to the QUSRSYS/QATMHINSTC file

*Required parameter group:*

**name** INPUT:CHAR(10)

Name of the server instance from which data is retrieved.

**buf** OUTPUT:CHAR(\*)

Buffer in format INSD0110 containing instance file data.

**buf\_size**

INPUT:BINARY(4)

Length of instance data buffer.

**format**

INPUT:CHAR(8)

Format of the instance data (INSD0110).

See "INSD0110 format" on page 171 for more information.

**buf\_actlen**

OUTPUT:BINARY(4)

Actual length of instance data returned.

**running**

OUTPUT:BINARY(4)

Indicates if the instance is currently running (1 = running).

**errcode**

I/O:CHAR(\*)

Error information structure.

*Error messages:***CPF3C17 E**

Error occurred with input data parameter.

**CPF3C19 E**

Error occurred with receiver variable specified.

**CPF3C1D E**

Input variable length in parameter &amp;1 not valid.

**CPF3C21 E**

Format name &amp;1 not valid.

**CPF3CF1 E**

Error code parameter not valid.

**HTPA001 E**

Input parameter &amp;1 not valid.

**HTPA101 E**

Server instance &amp;1 not found or is unreadable.

**HTPA127 E**

Server instance &amp;1 is not a HTTP Server type instance.

**Get Server Instance Names (QzuiGetInstanceNames) API:**

The QzuiGetInstanceNames() API allows you to obtain a list of IBM HTTP Server for i instance names.

## Required Parameter Group:

1	buf	Output	Char(*)
2	buf_size	Input	Binary(4)
3	format	Input	Char(8)
4	buf_actlen	Output	Binary(4)
5	count	Output	Binary(4)
6	errcode	I/O	Char(*)

Threadsafe: Yes

*Authorities and locks:*

- \*EXECUTE authority to the QUSRSYS library
- \*OBJOPR and \*READ authority to the QUSRSYS/QATMHINSTC file

*Required parameter group:***buf** OUTPUT:CHAR(\*)

Buffer to hold instance names and running data.

**buf\_size**

INPUT:BINARY(4)

Size of buffer passed.



**format**

INPUT:CHAR(8)

Format of instance name data (INSN0110).

**buf\_actlen**

OUTPUT:BINARY(4)

Number of bytes of data placed in buf.

**count** OUTPUT:BINARY(4)

Total number of instance names.

**errcode**

I/O:CHAR(\*)

Error information structure.

*INSN0110 format:* This data format is used by the QzuiGetInstanceNames() API.

Offset	Type	Field
0	Char(10)	Instance name
10	Char(2)	Reserved
12	Binary(4)	Running status
16	Binary(4)	Instance type 1 = HTTP Server

*Error messages:***CPF3C17 E**

Error occurred with input data parameter.

**CPF3C19 E**

Error occurred with receiver variable specified.

**CPF3C1D E**

Input variable length in parameter &amp;1 not valid.

**CPF3C21 E**

Format name &amp;1 not valid.

**HTPA001 E**

Input parameter &amp;1 not valid.

**Get Instance Type (QzuiGetInstanceType) API:**

The QzuiGetInstanceType() API allows you to obtain the type of an IBM HTTP Server for i instance. If the specified instance is not a valid instance, a -1 is returned.

## Required Parameter Group:

1 name	Input	Char(10)
2 itype	Output	Binary(4)
3 errcode	I/O	Char(*)

Threadsafe: Yes

*Authorities and locks:*

- \*EXECUTE authority to the QUSRSYS library
- \*OBJOPR and \*READ authority to the QUSRSYS/QATMHINSTC file

*Required parameter group:*

**name** INPUT:CHAR(10)  
The name of the instance.

**itype** OUTPUT:BINARY(4)  
The type of instance (-1 = Invalid, 1 = Apache)

**errcode**  
I/O:CHAR(\*)  
The error information structure.

*Error messages:*

**CPF3C17 E**  
Error occurred with input data parameter.

**CPF3C19 E**  
Error occurred with receiver variable specified.

**CPF3CF1 E**  
Error code parameter not valid.

**CPF9822 E**  
Not authorized to file &1 in library &2.

**HTPA101 E**  
Server instance &1 not found or is unreadable.

### **Open Apache Config File (QzuiOpenConfig) API:**

The QzuiOpenConfig() API allows you to read into memory an IBM HTTP Server for i server configuration file. The handle that is returned by the QzuiOpenConfig() API is used in subsequent API calls to manipulate directives and scopes within the server configuration data.

Required Parameter Group:

1 name	Input	Char(*)
2 namelength	Input	Binary(4)
3 writelock	Input	Binary(4)
4 cfg	Output	Binary(4)
5 errcode	I/O	Char(*)

Threadsafe: Yes

*Authorities and locks:*

- \*X authority to each directory in the path of the specified group file
- \*WX authority to the last directory in the path that will contain the group file path

*Required parameter group:*

**name** INPUT:CHAR(\*)  
File name (including path) to the configuration file to be opened.

**namelength**  
INPUT:BINARY(4)  
Length of the file name.

**writelock**  
INPUT:BINARY(4)

Has the following values: 0 = no lock, 1 = exclusive write lock will be put on config file.

**cfg** OUTPUT:BINARY(4)

Handle to the memory copy of the config file.

**errcode**

I/O:CHAR(\*)

Error information structure.

*Error messages:*

**CPF3C17 E**

Error occurred with input data parameter.

**CPF3C19 E**

Error occurred with receiver variable specified.

**CPF3C1D E**

Input variable length in parameter &1 not valid.

**CPF3CF1 E**

Error code parameter not valid.

**CPFB602 E**

Cannot open file.

**HTPA001 E**

Input parameter &1 not valid.

**HTPA104 E**

Server configuration not found or is unreadable.

### **Remove Config Object (QzuiRemoveConfigObject) API:**

The QzuiRemoveConfigObject() API allows you to remove a directive or scope from the IBM HTTP Server for i server configuration data. If a scope is removed, all the directives within it are also removed.

Required Parameter Group:

1	cfg	Input	Binary(4)
2	object	Input	Binary(4)
3	errcode	I/O	Char(*)

Threadsafe: Yes

*Authorities and locks:* None.

*Required parameter group:*

**cfg** INPUT:BINARY(4)

Handle to the config.

**object** INPUT:BINARY(4)

Handle to the object to be removed.

**errcode**

I/O:CHAR(\*)

Error information structure.

*Error messages:*

**CPF3C17 E**

Error occurred with input data parameter.

**CPF3CF1 E**

Error code parameter not valid.

**HTPA001 E**

Input parameter &1 not valid.

**HTPA106 E**

Input configuration handle not valid.

**HTPA121 E**

Object handle in parameter &1 not valid.

## **CGI programming**

The IBM HTTP Server for i supports the extension of the functionality of the HTTP Server through the use of Common Gateway Interface (CGI) programs.

### **The CGI Process**

The basic principle of Common Gateway Interface (CGI) is that a Web server passes client request information to CGI programs in system environment variables (and in some cases through standard input or command line arguments) and all standard output of CGI programs is returned to Web clients.

Most CGI programs include the following three stages:

- Parsing CGI input
- Processing the request
- Generating the response

Throughout the topic there will be references to conversion modes, which has to deal with how data is presented to a CGI programs and how data that is returned by the CGI program is processed by the HTTP Server. To learn more about conversion modes, see “CGI data conversions” on page 184.

**Note:** Any CGI program with a name that begins with `nph_` is considered a no parse header CGI program. This means that the server does no conversions on the data and adds no headers back in the response from the CGI program. The CGI programmer is in total control and is responsible for parsing the request and then sending all of the necessary headers back with the response.

### **Parsing CGI input**

When the environment variables have been set by the HTTP server, it starts the CGI program. (For complete list of environment variables set by the HTTP Server, see “Environment variables set by HTTP Server” on page 493.) It is then up to this CGI program to find out where to get the information needed to fulfill the request.

The two most common ways a CGI program may be called from the HTML document:

- By using an HTML form and the request method (environment variable `REQUEST_METHOD`) `POST`.
- By using an HTML anchor tag to specify the URL for the CGI program and adding the variables to this URL. This would be interpreted as `REQUEST_METHOD=GET`.

The CGI script has to perform the following tasks in order to retrieve the necessary information:

1. Find out the `REQUEST_METHOD` used by the client.
2. If the `REQUEST_METHOD` used was the `GET` method, the CGI program knows that all additional values may be retrieved from the `QUERY_STRING` environment variable.

3. If the REQUEST\_METHOD used was POST, the CGI knows that additional information was passed using STDIN. It will then have to query the CONTENT\_LENGTH environment variable to know how much information it will have to read from STDIN.

An example of data read in the QUERY\_STRING variable (%%MIXED%% mode):

```
NAME=Eugene+T%2E+Fox&ADDR=etfox%40ibm.net&INTEREST=RCO
```

Where

- A plus sign (+) represents spaces.
- A percent sign (%) that is followed by the American National Standard Code for Information Interchange (ASCII) hexadecimal equivalent of the symbol represents special characters, such as a period (.) or slash (/).
- An ampersand (&) separates fields and sends multiple values for a field such as check boxes.

Parsing breaks the fields at the ampersands and decodes the ASCII hexadecimal characters. The results look like this:

```
NAME=Eugene T. Fox  
ADDR=etfox@ibm.net  
INTEREST=RCO
```

You can use the QtmhCvtDb() API to parse the information into a structure. The CGI program can refer to the structure fields. If using %%MIXED%% input mode, the “%xx” encoding values are in ASCII and must be converted into the “%xx” EBCDIC encoding values before calling QtmhCvtDb(). If using %%EBCDIC%% mode, the server will do this conversion for you. The system converts ASCII “%xx” first to the ASCII character and then to the EBCDIC character. Ultimately, the system sets the EBCDIC character to the “%xx” in the EBCDIC CCSID.

The main advantage of using the GET method is that you can access the CGI program with a query without using a form.

The main advantage to the POST method is that the query length can be unlimited so you do not have to worry about the client or server truncating data. The query string of the GET method cannot exceed 8 KB.

## Processing the request

Processing the request is the second stage of a CGI program. In this stage, the program takes the parsed data and performs the appropriate action. For example, a CGI program designed to process an application form might perform one of the following functions:

1. Take the input from the parsing stage
2. Convert abbreviations into more meaningful information
3. Plug the information into an e-mail template
4. Use SNDDST to send the e-mail.

## Generating the response

When the CGI program has finished processing it has to send its result back to the HTTP server that invoked the program. By doing so the output indirectly is sent to the client that initially requested the information.

Because the CGI program issues its result through STDOUT, the HTTP server has to read the information from there and interpret what to do.

A CGI program writes a CGI header that is followed by an entity body to standard output. The CGI header is the information that describes the data in the entity body. The entity body is the data that the server sends to the client. A single newline character always ends the CGI header. The newline character for ILE C is \n. For ILE RPG or ILE COBOL, it is hexadecimal '15'. The following are some examples of Content-Type headers:

```
Content-Type: text/html\n\nContent-Type: text/html; charset=iso-8859-2\n\n
```

If the response is a static document, the CGI program returns either the URL of the document using the CGI Location header or returns a Status header. The CGI program does not have an entity body when using the Location header. If the host name is the local host, HTTP Server will retrieve the specified document that the CGI program sent. It will then send a copy to the Web client. If the host name is not the local host, the HTTP processes it as a redirect to the Web client. For example:

```
Location: http://www.acme.com/products.html\n\n
```

The Status header should have a Content\_Type: and a Status in the CGI header. When Status is in the CGI header, an entity body should be sent with the data to be returned by the server. The entity body data contains information that the CGI program provides to a client for error processing. The Status line is the Status with an HTTP 3 digit status code and a string of alphanumeric characters (A-Z, a-z, 0-9 and space). The HTTP status code must be a valid 3 digit number from the HTTP/1.1 specification.

**Note:** The newline character \n ends the CGI header.

```
CONTENT-TYPE: text/html\nStatus: 600 Invalid data\n\n<html><head><title>Invalid data</title>\n</head><body>\n<h1>Invalid data typed</h1>\n<br><pre>\nThe data entered must be valid numeric digits for id number\n<br></pre>\n</body></html>
```

#### **Related information:**

“CGI data conversions”

The server can perform ASCII to EBCDIC conversions before sending data to CGI programs. This is needed because the Internet is primarily ASCII-based and the IBM i server is an extended binary-coded decimal interchange code (EBCDIC) server. The server can also perform EBCDIC to ASCII conversions before sending data back to the browser. HTTP and HTML specifications allow you to tag text data with a character set (charset parameter on the Content-Type header). However, this practice is not widely in use today (although technically required for HTTP1.0/1.1 compliance). According to this specification, text data that is not tagged can be assumed to be in the default character set ISO-8859-1 (US-ASCII). The server correlates this character set with ASCII coded character set identifier (CCSID) 819.

“CGI APIs” on page 137

This topic provides information about IBM HTTP Server for i APIs for CGI applications.

“Environment variables set by HTTP Server” on page 493

The IBM HTTP Server for i supports the standard environment variables in addition to environment variables that are unique to the IBM i server.

## **CGI data conversions**

The server can perform ASCII to EBCDIC conversions before sending data to CGI programs. This is needed because the Internet is primarily ASCII-based and the IBM i server is an extended binary-coded decimal interchange code (EBCDIC) server. The server can also perform EBCDIC to ASCII conversions before sending data back to the browser. HTTP and HTML specifications allow you to tag text data with a character set (charset parameter on the Content-Type header). However, this practice is not widely in use today (although technically required for HTTP1.0/1.1 compliance). According to this specification,

text data that is not tagged can be assumed to be in the default character set ISO-8859-1 (US-ASCII). The server correlates this character set with ASCII coded character set identifier (CCSID) 819.

## National language support HTTP Server CGI directives

You can configure HTTP Server to control which mode is used by specifying the CGIConvMode directive in different contexts, such as server config or directory:

CGIConvMode Mode

Where *Mode* is one of the following:

- BINARY
- EBCDIC
- EBCDIC\_JCD

You can configure HTTP Server to set the ASCII and EBCDIC CCSIDs that are used for conversions by specifying the directives DefaultNetCCSID and CGIJobCCSID in different contexts, such as server config or directory. For example:

- DefaultNetCCSID 819
- CGIJobCCSID 37

You can configure HTTP Server to set the locale environment variable by specifying the CGIJobLocale in different contexts, such as server config or directory: CGIJobLocale /QSYS.LIB/EN\_US.LOCALE.

## CGI input conversion modes

The following table summarizes the type of conversion that is performed by the server for each CGI mode.

Table 13. Conversion action for text in CGI Stdin

CGI_MODE	Conversion	Stdin encoding	Environment variable	Query_String encoding	argv encoding
BINARY or %%BINARY%%	None	No conversion	CGI job CCSID	No conversion	No conversion
EBCDIC or %%EBCDIC%%	CGI NetCCSID to CGI job CCSID	CGI job CCSID	CGI job CCSID	CGI job CCSID	CGI job CCSID
%%EBCDIC%% or %%EBCDIC_JCD%% with charset tag received	Calculate target EBCDIC CCSID based on received ASCII charset tag	EBCDIC equivalent of received charset	CGI job CCSID	CGI job CCSID	CGI job CCSID
EBCDIC_JCD or %%EBCDIC_JCD%%	Detect input based on received data. Convert data to CGI job CCSID	Detect ASCII input based on received data. Convert data to CGI job CCSID.	CGI job CCSID	Detect ASCII input based on received data. Convert data to CGI job CCSID.	Detect ASCII input based on received data. Convert data to CGI job CCSID
%%MIXED%% (Compatibility mode)	CGI NetCCSID to CGI job CCSID (receive charset tag is ignored)	CGI job CCSIDwith ASCII escape sequence	CCSID 37	CCSID 37 with ASCII escape sequence	CCSID 37 with ASCII escape sequence

**Note:** If the directive CGIJobCCSID is present, the CGI job runs under its specified CCSID value. Otherwise, the DefaultFsCCSID value is used (the default job CCSID).

## BINARY

The BINARY mode, delivers QueryString and stdin to the CGI program in ASCII, exactly as it was received from the client. The environment variables are in the CGI job CCSID. If CGIJobCCSID is present the job CCSID has its value; otherwise, the value associated with DefaultFsCCSID (the default job CCSID) is used.

## EBCDIC

The EBCDIC mode, delivers all of the information to the CGI program in the job CCSID. The ASCII CCSID of the QueryString or stdin data is determined from a charset tag on the content type header if present. If CGIJobCCSID is present the job CCSID has its value; otherwise, the value associated with DefaultFsCCSID (the default job CCSID) is used.

## EBCDIC\_JCD

The EBCDIC\_JCD mode is the same as the EBCDIC mode except that a well-known Japanese codepage detection algorithm is used to determine the ASCII CCSID when the charset tag is not present. Japanese browsers can potentially send data in one of three code pages, JIS (ISO-2022-JP), S-JIS (PC-Windows), or EUC (UNIX).

## CGI output conversion modes

This following table summarizes the type of conversion that is performed and the charset tag that is returned to the browser by the server.

*Table 14. Conversion action and charset tag generation for text in CGI Stdout*

CGI Stdout CCSID/Charset in HTTP header	Conversion action	Server reply charset tag
EBCDIC CCSID/Charset	Calculate EBCDIC to ASCII conversion based on supplied EBCDIC CCSID/Charset	Calculated ASCII charset
ASCII CCSID/Charset	No conversion	Stdout CCSID/Charset as Charset
65535	No conversion	None
None (CGIConvMode= %%BINARY%%, %%BINARY/MIXED%%, or %%BINARY/EBCDIC%%)	Default Conversion - job CCSID to NetCCSID	NetCCSID as charset
None (CGIConvMode= BINARY or %%BINARY/BINARY%%)	No conversion	None
None (CGIConvMode= EBCDIC, %%EBCDIC%%, %%EBCDIC/MIXED%%, or %%EBCDIC/EBCDIC%%)	Default Conversion - job CCSID to NetCCSID	NetCCSID as charset
None (CGIConvMode= EBCDIC, EBCDIC_JCD, %%EBCDIC%%, %%EBCDIC/MIXED%%, or %%EBCDIC/EBCDIC%% with charset tag received on HTTP request)	Use inverse of conversion calculated for stdin	Charset as received on HTTP request
None (CGIConvMode= %%EBCDIC_JCD%%, %%EBCDIC_JCD/MIXED%%, or %%EBCDIC_JCD/EBCDIC%%)	Use inverse of conversion calculated by the Japanese codepage detection	ASCII CCSID as charset
None (CGIConvMode= %%MIXED%% or %%MIXED/MIXED%%)	Default Conversion - job CCSID to NetCCSID	None (compatibility mode)
Invalid	CGI error 500 generated by server	

## BINARY

In this mode HTTP header output is in CCSID 819 with the escape sequences also being the



ASCII representative of the ASCII code point. An example of a HTTP header that may contain escape sequences is the Location header. The body is always treated as binary data and the server performs no conversion.

## EBCDIC

In this mode HTTP header output is assumed to be in the CGI job CCSID, unless otherwise specified in a charset or CCSID tag by the CGI program. However, the escape sequence must be the EBCDIC representative of the EBCDIC code point for the 2 characters following the  $\Delta\Delta$  in the escape sequence. An example of a HTTP header that may contain escape sequences is the Location header. The body (if the mime type is text/\*) is assumed to be in the job CCSID, unless otherwise specified in a charset or CCSID tag by the CGI program. If CGIJobCCSID is present the CGI job CCSID has its value; otherwise, the value associated with DefaultFsCCSID (the default job CCSID) is used.

## EBCDIC\_JCD

In this mode HTTP header output is assumed to be in the job CCSID, unless otherwise specified in a charset or CCSID tag by the CGI program. However, the escape sequence must be the EBCDIC representation of the EBCDIC code point for the 2 characters following the  $\Delta\%$  in the escape sequence. An example of a HTTP header that may contain escape sequences is the Location header. The body (if the mime type is text/\*) is assumed to be in the job CCSID, unless otherwise specified in a charset or CCSID tag by the CGI program. If CGIJobCCSID is present the job CCSID has its value; otherwise, the value associated with DefaultFsCCSID (the default job CCSID) is used.

## CGI environment variables

The following CGI environment variables that are related to national language support are set by the HTTP server prior to calling a CGI program:

- CGI\_MODE - which input conversion mode the server is using (%%MIXED%%, %%EBCDIC%%, %%BINARY%%, %%EBCDIC\_JCD%%, EBCDIC, BINARY, or EBCDIC\_JCD)
- CGI\_ASCII\_CCSID - from which ASCII CCSID was used to convert the data
- CGI\_EBCDIC\_CCSID - which EBCDIC CCSID the data was converted into
- CGI\_OUTPUT\_MODE - which output conversion mode the server is using (%%MIXED%%, %%EBCDIC%%, %%BINARY%%, EBCDIC, BINARY, or EBCDIC\_JCD)
- CGI\_JOB\_LOCALE - which locale to use in the CGI program. This environment variable is set only if the CGIJobLocale directive is set.

For complete list of environment variables set by the HTTP Server, see “Environment variables set by HTTP Server” on page 493.

## DBCS considerations

URL-encoded forms containing DBCS data could contain ASCII octets that represent parts of DBCS characters. The server can only convert non-encoded character data. This means that it must un-encode the double-byte character set (DBCS) stdin and QUERY\_STRING data before performing the conversion. In addition, it has to reassemble and re-encode the resulting EBCDIC representation before passing it to the CGI program. Because of this extra processing, CGI programs that you write to handle DBCS data may choose to receive the data as BINARY and perform all conversions to streamline the entire process.

**Using the EBCDIC\_JCD mode:** The EBCDIC\_JCD mode determines what character set is being used by the browser for a given request. This mode is also used to automatically adjust the ASCII/EBCDIC code conversions used by the web server as the request is processed.

After auto detection, the %%EBCDIC\_JCD%% or EBCDIC\_JCD mode converts the stdin and QUERY\_STRING data from the detected network CCSID into the correct EBCDIC CCSID for Japanese. The default conversions configured for the CGI job are overridden. The DefaultFsCCSID directive or the

-fscsid startup parameter specifies the default conversions. The startup FsCCSID must be a Japanese CCSID. Alternately, the CGIJobCCSID can be set to a Japanese CCSID.

The possible detected network code page is Shift JIS, eucJP, and ISO-2022-JP. The following are the associated CCSIDs for each code page:

Shift JIS

=====

CCSID 932: IBM PC (old JIS sequence, OS/2 J3.X/4.0, IBM Windows J3.1)

CCSID 942: IBM PC (old JIS sequence, OS/2 J3.X/4.0)

CCSID 943: MS Shift JIS (new JIS sequence, OS/2 J4.0

MS Windows J3.1/95/NT)

eucJP

=====

CCSID 5050: Extended UNIX Code (Japanese)

ISO-2022-JP

=====

CCSID 5052: Subset of RFC 1468 ISO-2022-JP (JIS X 0201 Roman and JIS X 0208-1983) plus JIS X 0201 Katakana.

CCSID 5054: Subset of RFC 1468 ISO-20220JP (ASCII and JIS X 0208-1983)

plus JIS X 0201 Katakana.

The detected network CCSID is available to the CGI program. The CCSID is stored in the CGI\_ASCII\_CCSID environment variable. When JCD can not detect, the default code conversion is done as configured (between NetCCSID and FsCCSID or CGIJobCCSID).

Since the code page of Stdin and QUERY\_STRING are encoded according to the web client's outbound code page, we recommend using the following configuration value combinations when you use the EBCDIC\_JCD or %%EBCDIC\_JCD%% mode.

Table 15. Recommended CCSID configuration combinations

Startup (FsCCSID)/CGI job CCSID (CGIJobCCSID)	Startup (DefaultNetCCSID)/CGI Net CCSID (DefaultNetCCSID)	Description
5026/5035 (See note 4)	943 Default:	MS Shift JIS
5026/5035 (See note 4)	942 Default	IBM PC
5026/5035 (See note 4)	5052/5054 Default	ISO-2022-JP

Using CCSID 5050(eucJP) for the startup NetCCSID, is not recommended. When 5050 is specified for the startup NetCCSID, the default code conversion is done between FsCCSID and 5050. This means that if JCD cannot detect a code page, JCD returns 5050 as the default network CCSID. Most browser's use a default outbound code page of Shift JIS or ISO-2022-JP, not eucJP.

If the web client sends a charset tag, JCD gives priority to the charset tag. Stdout function is the same. If the charset/ccsid tag is specified in the Content-Type field, stdout gives priority to charset/ccsid tag. Stdout also ignores the JCD detected network CCSID.

**Notes:**

1. If startup NetCCSID is 932 or 942, detected network, Shift JIS's CCSID is the same as startup NetCCSID. Otherwise, Shift JIS's CCSID is 943.

Startup NetCCSID Shift JIS (JCD detected CCSID)

-----

932 932

942 942

943 943

5052 943

5054 943

5050 943

2. Netscape Navigator 3.x sends the alphanumeric characters by using JIS X 0201 Roman escape sequence (CCSID 5052) for ISO-2022-JP. Netscape Communicator 4.x sends the alphanumeric characters by using ASCII escape sequence (CCSID 5054) for ISO-2022-JP.
3. JCD function has the capability to detect EUC and SBCS Katakana, but it is difficult to detect them. IBM recommends that you do not use SBCS Katakana and EUC in CGI.
4. CCSID 5026 assigns lowercase alphabet characters on a special code point. This often causes a problem with lowercase alphabet characters. To avoid this problem, do one of the following:
  - Do not use lowercase alphabet literals in CGI programs if the FsCCSID is 5026.
  - Use CCSID 5035 for FsCCSID.
  - Use the Charset/CCSID tag as illustrated in the following excerpt of a CGI program:
 

```
main(){
printf("Content-Type: text/html; Charset=ISO-2022-JP\n\n");
...
}
```
  - Do the code conversions in the CGI program. The following sample ILE C program converts the literals into CCSID 930 (the equivalent to CCSID 5026):
 

```
main(){
printf("Content-Type: text/html\n\n");
#pragma convert(930)
printf("<html>");
printf("This is katakana code page\n");
#pragma convert(0)
...
}
```
  - When the web client sends a charset tag, the network CCSID becomes the ASCII CCSID associated with Multipurpose Internet Mail Extensions (MIME) charset header. The charset tag ignores the JCD detected CCSID. When the Charset/CCSID tag is in the Content-Type header generated by the CGI program, the JCD-detected CCSID is ignored by this Charset/CCSID. Stdout will not perform a conversion if the charset is the same as the MIME's charset. Stdout will not perform a conversion if the CCSID is ASCII. Stdout will perform code conversion if the CCSID is EBCDIC. Because the environment variables and stdin are already stored in job CCSID, ensure that you are consistent between the job CCSID and the Content-Type header's CCSID.

## Writing high availability CGI programs

High availability CGI programs use APIs to preserve state information. The state information can be accessed by different IBM i servers that are participating as cluster nodes in a clustered environment, even after a failure or switchover of the HTTP Server or IBM i server.

During the configuration of a Web server, the server administrator indicates whether CGI programs are allowed to be cluster-enabled high availability CGI programs. If the server receives a request for a CGI program that is allowed to be Highly Available (HA), the Web server passes to the CGI an environment variable that indicates the CGI may be cluster-enabled. The server also creates and passes a unique session handle to the CGI program. The CGI program must then acknowledge that it is a cluster-enabled HA CGI program to the server, otherwise the server will regard the CGI as not being cluster-enabled.

The following environment variables are passed by the Web server to High Availability CGI programs:

- QZHBIS\_FIRST\_REQUEST
- QZHBIS\_CLUSTER\_ENABLED
- QZHBNEXT\_SESSION\_HANDLE
- QZHBRECOVERY
- QZHBHA\_MODEL

The `Cluster-Enabled` and `Accept-HTSession` headers should be returned in each response from a High Availability CGI program. For example,

```
Cluster-Enabled:1
```

An error will result if the `Cluster-Enabled` header is returned by a CGI program with a value of `1`, but the Web Server is not configured to allow that CGI program to be Highly available.

When the Web server receives the `Cluster-Enabled` header with a value of `1`, the server will create a new session entry and indicate that the session is cluster-enabled.

Cluster-enabled CGI programs will return the `Accept-HTSession` header to the Web server with a value equal to the value passed to the CGI in the `QZHBNEXT_SESSION_HANDLE` environment variable. An error will result if the value specified with `Accept-HTSession` does not match the value passed to the CGI in `QZHBNEXT_SESSION_HANDLE`. For CGI programs that are not cluster-enabled, the `Accept-HTSession` CGI header remains unmodified.

The Web server associates a high availability CGI program's state with the unique session handle that was passed as an environment variable to the CGI. If a request to run the CGI is sent to the Web server, and the requested URL includes the specific session handle, the Web server will be able to correctly restore the previous state of the CGI. For this reason it is important that the session handle appear in all URLs that were generated by the high availability CGI program to be returned to the client.

A high availability CGI program uses two APIs to maintain its state. To store state information, the CGI calls the API `QzhbCgiSendState_r()`. To retrieve state information, the CGI program calls the API `QzhbCgiRecvState_r()`.

## Guidelines for writing high availability CGIs

A CGI program developer should follow the following rules when writing high availability CGI programs:

- Write the CGI in such a way that running them with the same state more than once does not cause any problem.
- Store the CGI program's state between client's requests only in the Web server.
- Avoid using data sharing mechanisms that do not fit in the high availability Web server programming model provided by the HTTP Server. An example of such a model would be a CGI program that is using shared memory.
- The Web server limits the total number of persistent CGIs, which includes high availability CGI, using the `MaxPersistentCGI` directive.

*Table 16. CGI problems and solutions.* This table identifies potential problem areas and suggests a solution:

Potential problems	Solutions
When the stateful CGI is run with the same state more than once, its correctness is not ensured.	Rewrite the CGI so that it can run with the same state more than once.
The stateful CGI accesses shared memory.	Eliminate the use of shared memory.
The stateful CGI generates session handles ignoring session handles passed by the Web server.	Rewrite the CGI to use session handles passed by the Web server.

There are two categories of high availability Web server programming models to consider when writing high availability CGI programs or enabling an existing CGI program for use as a high availability CGI program. The two categories are:

- Primary/backup
- Peer model

For the primary/backup, follow these additional guidelines:

- The stateful data is saved by the high availability CGI program by calling the `QzhbCgiSendState_r()` API. To retrieve any stateful data that has been stored use the `QzhbCgiRecvState_r()` API. The `QzhbCgiRecvState_r()` API returns stateful information when the environment variable `QZHBRECOVERY` is set and `QZHBHA_MODEL` is equal to `PRIMARYBACKUP`. If the `QZHBRECOVERY` is not set, then the CGI program should not use the `QzhbCgiRecvState_r()` API. You must write a persistent CGI that maintains the data in static variables. If the environment variable `QZHBRECOVERY` is set, retrieve the data using the `QzhbCgiRecvState_r()` API and restore the static variables.

For the Peer model, follow these additional guidelines:

- The stateful data is saved by the high availability CGI program by calling the `QzhbCgiSendState_r()` API. To retrieve any stateful data that has been stored use the `QzhbCgiRecvState_r()` API. The `QzhbCgiRecvState_r()` API must be used with each new request to retrieve any stateful data that has been stored for a previous high availability CGI program invocation. In this model your CGI program must not save stateful data in static variables.
- If `QZHBHA_MODEL` is `PUREPEER` the CGI is expected to restore its state, to serve the request, and to return its new state to the Web server. When the Web server receives the new CGI's state, it stores the state (which will be passed to the CGI with the subsequent request), returns the response to the client, and terminates the CGI job.

#### **Related information:**

“Highly available HTTP Server” on page 43

The IBM HTTP Server for i supports Web server clusters, which ensures high availability of your Web site.

“CGI data conversions” on page 184

The server can perform ASCII to EBCDIC conversions before sending data to CGI programs. This is needed because the Internet is primarily ASCII-based and the IBM i server is an extended binary-coded decimal interchange code (EBCDIC) server. The server can also perform EBCDIC to ASCII conversions before sending data back to the browser. HTTP and HTML specifications allow you to tag text data with a character set (charset parameter on the Content-Type header). However, this practice is not widely in use today (although technically required for HTTP1.0/1.1 compliance). According to this specification, text data that is not tagged can be assumed to be in the default character set ISO-8859-1 (US-ASCII). The server correlates this character set with ASCII coded character set identifier (CCSID) 819.

“CGI APIs” on page 137

This topic provides information about IBM HTTP Server for i APIs for CGI applications.

“Environment variables set by HTTP Server” on page 493

The IBM HTTP Server for i supports the standard environment variables in addition to environment variables that are unique to the IBM i server.

## **Writing persistent CGI programs**

*Persistent CGI* is an extension to the CGI interface that allows a CGI program to remain active across multiple browser requests and maintain a session with that browser client. This allows files to be left open, the state to be maintained, and long running database transactions to be committed or rolled-back based on end-user input.

The CGI program must be written using named activation groups which allows the program to remain active after returning to the server. The CGI program notifies the server it wants to remain persistent using the `␣Accept-HTTPSession␣` CGI header as the first header it returns. This header defines the session ID associated with this instance of the CGI program and is not returned to the browser. Subsequent URL requests to this program must contain the session ID as the first parameter after the program name. The server uses this ID to route the request to that specific instance of the CGI program. The CGI program should regenerate this session ID for each request. It is strongly recommended that you use Secure Sockets Layer (SSL) for persistent and secure business transaction processing.

## Accept-HTSession CGI Header

This header specifies the session handle associated with this instance of the Persistent CGI program. This session handle is used to route back subsequent requests to that program and must be unique, or the server will not honor the persistence request. A message is logged in the error log of the server.

```
Accept-HTSession = "Accept-HTSession" ":" handle
```

When the server receives this header, the CGI job servicing the request will be reserved in a persistent state. Only requests coming in with that session handle in the URL are routed back to that instance of the CGI program. The URL must be in the following format:

```
/path/cgi-name/handle/rest/of/path
```

Where *handle* is an exact match of the handle provided in the `Accept-HTSession` CGI header for the program `cgi-name`.

**Note:** The `cgi-name` that is being resolved is the name as it appears in the URL. It is not necessarily the actual name of the program being started on the system. This is to remain consistent with the name resolution performed by the server.

## HTTimeout CGI Header

The `HTTimeout` header is for the CGI program to define the amount of time, in minutes, that this CGI program wants to wait for a subsequent request. If not specified, the value specified on the `PersistentCGITimeout` directive is used. If specified, it takes precedence over the `PersistentCGITimeout` directive, but the server will not wait longer than the time specified on the `MaxPersistentCGITimeout` directive. This allows individual CGI programs to give users more time to respond to lengthy forms or explanations. However, it still gives the server ultimate control over the maximum time to wait.

```
HTTimeout = "HTTimeout" ":" minutes
```

The *time-out* value is a non-negative decimal integer, representing the time in minutes. This header must be preceded by an `Accept-HTSession` header, if not, it is ignored. If you omit the header, the default *time-out* value for the server is used. When a CGI program is ended because of a timeout, a message is logged in the error log of the server.


## Considerations for using Persistent CGI Programs

You should be aware of the following considerations when using persistent CGI programs:

- The web administrator can limit the number of persistent CGI programs that the server supports by using the `MaxPersistentCGI` configuration directive.
- There are some job or thread-level resources that the server code running in the CGI job usually manipulates (directly or indirectly) on behalf of CGI programs. The following attributes will (potentially) change across calls:
  - Environment variables the server sets
  - Stdin/Stdout/Stderr file descriptors
  - User profile
  - Library list
- The server will not set the rest of the job attributes set by the server, and therefore, will maintain state across calls if changed by the CGI program. Note, however, that the CGI program must restore the initial state of these values before ending its persistence in order to guarantee compatibility across subsequent server requests:
  - Job Language, Region, CCSID
  - Job Priority
  - Printer/Output Queue

- Message Logging
- Environment variables set by the CGI program
- For added security, web server administrators can protect their persistent CGI programs using registered Internet users, thereby forcing authentication by the user before processing each request.

## Persistent CGI Program Example

The Persistent CGI programming example located at CGI Programming examples  displays a counter that is increased each time the Persistent CGI program is called.

### Related information:

“CGI data conversions” on page 184

The server can perform ASCII to EBCDIC conversions before sending data to CGI programs. This is needed because the Internet is primarily ASCII-based and the IBM i server is an extended binary-coded decimal interchange code (EBCDIC) server. The server can also perform EBCDIC to ASCII conversions before sending data back to the browser. HTTP and HTML specifications allow you to tag text data with a character set (charset parameter on the Content-Type header). However, this practice is not widely in use today (although technically required for HTTP1.0/1.1 compliance). According to this specification, text data that is not tagged can be assumed to be in the default character set ISO-8859-1 (US-ASCII). The server correlates this character set with ASCII coded character set identifier (CCSID) 819.

“CGI APIs” on page 137

This topic provides information about IBM HTTP Server for i APIs for CGI applications.

“Environment variables set by HTTP Server” on page 493

The IBM HTTP Server for i supports the standard environment variables in addition to environment variables that are unique to the IBM i server.

## CGI programs and activation groups

The following section is intended to give a brief overview of activation groups.

**Note:** It is very important to become familiar with the details of activation groups prior to developing or porting a CGI application that will use this support.

### Activation groups

*Program activation* is the process that is used to prepare a program to run. The system must activate ILE programs before they can be run. Program activation includes the allocation and initialization of static storage for the program in addition to completing the binding of programs to service programs. Named activation groups must be used when running persistent CGI.

Program activation is not a unique concept. All modern computer operating systems must perform program initialization and load. What is unique to CGI programs on the IBM i server is the concept of Activation Groups. All ILE programs and service programs are activated within an activation group. This substructure contains the resources necessary to run the program. The resources that are contained and are managed with an activation group include:

- Static and automatic program variables
- Dynamic storage
- Temporary data management resources (For example, open files and SQL cursors)
- Certain types of exception handlers and ending procedures

Runtime creation of ILE activation groups is controlled by specifying an activation group attribute when your program or service program is created. The attribute is specified by using the ACTGRP parameter on the CRTPGM or CRTSRVPGM command. The valid options for this attribute include user-named, \*NEW, and \*CALLER. The following is a brief description of these options:

## user-named

A named activation group allows you to manage a collection of ILE programs and ILE service programs as one application. The activation group is created when it is first needed. All programs and service programs that specify the same activation group name use it then. A user-named activation group is left active after the program has exited normally. All storage associated with that program is still allocated and in  $\Delta$ last-used $\Delta$  state. The program is not initialized when it is called again. In addition, for the ILE C runtime, all settings are in  $\Delta$ last-used $\Delta$  state, such as `signal()`, and `strtok()`. The `RCLACTGRP` command is used to end a named activation group. Use the `DSPJOB OPTION(*ACTGRP)` command to display all the activation groups for the job.

**\*NEW** The name for this activation group is selected by ILE and will always be unique. System-named activation groups are always deleted when the high level language returns. **\*NEW** is the standard behavior that can be expected on other systems such as UNIX.

## \*CALLER

Specifying **\*CALLER** causes the ILE program or service program to be activated within the activation group of the calling program. A new activation group is never created with this attribute.

## Notes:

1. When you create a persistent CGI program, you must specify a named activation group.
2. CGI programs that are not persistent should not refer to job-level scoped resources.

For additional information about activation groups see the ILE Concepts manual.

## CGI considerations

There are advantages to running CGI programs in either a user-named or **\*CALLER** activation group. The performance overhead associated with activating a CGI every time that is requested can be drastically reduced. It is important to understand that because the system does not delete user-named activation groups, normal high level language end verbs cannot provide complete end processing. For example, the system will not close open files, and the system will not return the static and heap storage that are allocated by a program. The program must manage these resources explicitly. This will be especially important when changing the activation group of CGI programs that rely on their end processing functions to run properly.

**Note:** When you activate multi-threaded CGI on your web server, you get multiple thread support for your CGI application. Your CGI application must end all of its threads before returning to the server. When using multi-thread capable CGI, you need to put the CGI program in a new or named activation group.

The following section shows examples which will work fine running in a **\*NEW** activation group, however will cause problems if run in a user-named or **\*CALLER** activation group.

## Activation group examples

**Note:** CGI programming examples are also available on the IBM HTTP Server for i website  .

In the following example a CGI program when run in a **\*NEW** activation group, would write Hello World to the browser. What is important to understand is that this application is taking advantage of job end processing to delete the stdio buffers that are used to buffer the stdout data.

You could build the following CGI program to run in either a user-named or **\*CALLER** activation group. In such an instance, the server will not process the information that was written to stdout. This will cause



the web browser to display a  $\Delta$ Document Contains No Data $\Delta$  error message. Another application could run again in the same activation group that properly erased stdout. In this instance, the data that has been buffered from previous calls would be sent.

```
#include <stdio.h>
void main(void) {

    /* Write header information. */
    printf("Content-type: text/html\n\n");

    /* Write header information. */
    printf("Hello World\n");
}
```

End processing may not erase stdout buffers so the application must erase the stdout with a `fflush(stdout)` call. The following example will work regardless of the activation group specification:

```
#include <stdio.h>
void main(void) {

    /* Write header information. */
    printf("Content-type: text/html\n\n");

    /* Write header information. */
    printf("Hello World\n");

    /* Flush stdout. */
    fflush(stdout);
}
```

When run in a `*NEW` activation group, this example CGI would read `CONTENT_LENGTH` bytes of data from stdin and write this back out to stdout. The system has allocated the buffer that is used to hold the data by invoking `malloc()`. Like the example that is previously shown, this application is relying on several aspects of job end processing to function properly.

If this CGI program were built to run in either a user-named or `*CALLER` activation group, the following problems would occur:

- As with the simple example that is previously shown, the application is not erasing stdout. This would cause the web browser to display a  $\Delta$ Document Contains No Data $\Delta$  error message. You could run another application again in the same activation group that properly erased stdout. This would send the data that has been buffered from previous calls.
- Stdin is buffered similar to stdout. If the contents of stdin are not erased, the stdin data on the second and all following calls of the CGI program will be unpredictable and the contents may at times contain information from subsequent requests.
- The heap storage allocated using `malloc()` is not being freed. Over time, a memory leak error like this could use significant amounts of memory. This is a common application error that only surfaces when the application is not running in a `*NEW` activation group.

```
/******
/* CGI Example program.
/******
#include
void main(void)
{
char* stdinBuffer;
char* contentLength;
int numBytes;
int bytesRead;
FILE* pStdin;

/* Write the header. */
printf("Content-type: text/html\n\n");
```

```

/* Get the length of data on stdin. */
contentLength = getenv("CONTENT_LENGTH");
if (contentLength != NULL) {

    /* Allocate storage and clear the storage to hold the data. */
    numBytes = atoi(contentLength);
    stdinBuffer = (char*)malloc(numBytes+1);
    if ( stdinBuffer )
        memset(stdinBuffer, 0x00, numBytes+1);

    /* Read the data from stdin and write back to stdout. */
    bytesRead = fread(stdinBuffer, 1, numBytes, pStdin);
    stdinBuffer[bytesRead+1] = '\0';
    printf("%s", stdinBuffer);
}
else
    printf("Error getting content length\n");
return;
}

```

The following example shows the changes that would be required to this application to allow it to run in a user-named or \*CALLER activation group:

```

/*****
/* CGI Example program with changes to support user-named */
/* and *CALLER ACTGRP. */
*****/
#include
void main(void)
{
char* stdinBuffer;
char* contentLength;
int numBytes;
int bytesRead;
FILE* pStdin;

/* Write the header. */
printf("Content-type: text/html\n\n");

/* Get the length of data on stdin. */
contentLength = getenv("CONTENT_LENGTH");
if (contentLength != NULL) {

    /* Allocate storage and clear the storage to hold the data. */
    numBytes = atoi(contentLength);
    stdinBuffer = (char*)malloc(numBytes+1);
    if ( stdinBuffer )
        memset(stdinBuffer, 0x00, numBytes+1);

    /* Reset stdin buffers. */
    pStdin = freopen("", "r", stdin);

    /* Read the data from stdin and write back to stdout. */
    bytesRead = fread(stdinBuffer, 1, numBytes, pStdin);
    stdinBuffer[bytesRead+1] = '\0';
    printf("%s", stdinBuffer);

    /* Free allocated memory. */
    free(stdinBuffer);
}
else
    printf("Error getting content length\n");

/* Flush stdout. */
fflush(stdout);
return;
}

```

## Running CGI programs in IBM PASE for i

The IBM HTTP Server for i Web server can run CGI programs created to run in the IBM Portable Application Solutions Environment for i. In addition, the HTTP Server can also run programs that follow the FastCGI protocol.

CGI programs that currently run on the AIX platform may be able to run on an IBM HTTP Server for i Web server in PASE for i. To do this, store your CGI programs in the QOpenSys file system. You should then verify that your program will run in PASE for i. For more information on how to prepare your code and ensure that it will run effectively in PASE for i, see the Prepare programs to run in IBM PASE for i topic. And finally, use the ScriptAlias directive in the configuration file, httpd.conf, to map the URL to the program, as you would with any CGI program.

For CGI programs that run in PASE for i, environment variables are converted from the CGI job CCSID to the CCSID specified by the ILE environment variable QIBM\_PASE\_CCSID. The ILE environment variable PASE\_LANG specifies the PASE Locale. The default values are functions of the current LANGID and CNTRYID attributes of the CGI job, but the system uses PASE\_LANG=POSIX and QIBM\_PASE\_CCSID=819 if it does not recognize the LANGID and CNTRYID pair. The LANG environment variable controls the default locale for the CGI program that will be running in PASE for i. These default values may be overridden by setting both ILE environment variables 'PASE\_LANG' and 'QIBM\_PASE\_CCSID' using the HTTP directive 'setenv'. If either of these are not set, the default values will be used. For example, setenv PASE\_LANG JA\_JP setenv QIBM\_PASE\_CCSID 1208

See PASE for i Locales to determine what locales are supported.

**Note:** CGI programs that run in PASE for i must have file names that do not include the following extensions which are reserved for CGI programs that do not run in PASE for i:

- .rexx
- .pl
- .pgm
- .class

### Sample CGI Program Configuration

This sample code shows one way to use the ScriptAlias directive to map your CGI program to a URL.

```
ScriptAlias /cgi-pase/ /QOpenSys/myserver/cgi-bin/
```

## Running FastCGI applications in IBM PASE for i

The IBM HTTP Server for i is able to run AIX programs that implement the FastCGI protocol. FastCGI is an interface between Web servers and applications which combines some of the performance characteristics of native Web server modules with the Web server independence of the CGI programming interface. Like AIX CGI programs, AIX FastCGI applications are run in the PASE for i environment. For more information about how the HTTP Server supports FastCGI, see the IBM HTTP Server for i

Documentation  Web page.

### Related information:

 [FastCGI Web site](#)

## Setting up CGI programs for HTTP Server

This topic provides information about how to set up CGI programs for your IBM HTTP Server for i Web server.

You can extend the capability of the HTTP Server by adding CGI programs. The HTTP Server supports Integrated Language Environment (ILE) CGI programs and AIX CGI programs.

Here is a summary of the steps you need to take to enable your server to run CGI programs:

### 1. Create the CGI program.

ILE CGI programs can be written in ILE C/C++, ILE RPG, or ILE COBOL programming languages. The HTTP Server provides CGI application programming interfaces in support of ILE CGI programming.

In addition to support for ILE CGI programs, the HTTP Server has the ability to run REXX programs and AIX programs as CGI programs. For more information about running AIX CGI programs, see "Running CGI programs in IBM PASE for i" on page 197.

### 2. Move the CGI program to the CGI directory.

ILE CGI programs must reside in the QSYS.LIB file system. REXX CGI programs must reside in database files named REXX or QREXSRC. AIX CGI programs must reside in the QOpenSys file system.

### 3. Ensure that your program has the correct authority using \*PUBLIC, QTMHHTTP or QTMHHTTP1.

If the UserID directive is not active, the server profile QTMHHTTP1 needs access to the CGI program and all objects the program accesses. If the UserID directive is active, the UserID profile needs access to the CGI program and all objects the program accesses.

### 4. Make changes to the HTTP Server configuration file.

For example, you need to add the ScriptAlias directive at a minimum.

**Note:** For REXX programs, you only need to indicate the path and the file name in the ScriptAlias directive. For example:

```
ScriptAlias /REXX /QSYS.LIB/AS400CGI.LIB/QREXSRC.FILE/*
```

The URL is :

```
http://hostname/REXX/samplecgi.REXX
```

## Apache module programming

The IBM HTTP Server for i supports the extension of the functionality of the HTTP Server through the use of third-party Apache modules.

### Related information:

"Apache module APIs" on page 137

This topic provides information about the Apache portable runtime (APR) and application programming interfaces (APIs) for the IBM HTTP Server for i. These APIs are generally used to write cross-platform Apache modules.


[Developer Documentation for Apache 2.0](#)

[Apache Portable Runtime Project](#)

## Setting up third party modules for HTTP Server

This topic provides information about how to set up third party modules for your IBM HTTP Server for i Web server.

The HTTP Server can extend its functionality in specific areas of your server using modules. For example, a module could be configured to create a new type of authentication that is not available with the

shipped HTTP Server. The Apache Software Foundation (ASF)  provides basic information for writing your own modules. Before the module can be used by your HTTP Server, it must be compiled

and saved in the QSYS directory. In addition, the LoadModule directive must be entered in your server configuration file along with any specific context required information.

As of IBM i 5.4, modules must be recompiled with a UTF locale. This creates an environment where locale-dependent C runtime functions assume that string data is encoded in UTF-8. Any hardcoded constants can be encoded in UTF-8 by adding a `#pragma convert(1208)` statement in the module. Additionally, input data from the client will no longer be converted to EBCDIC but will be passed as-is. Output data sent from the module is not converted either so it must be encoded in ASCII or UTF8 as required. APR and HTTP APIs as of V5R4, expect data in UTF-8. Note that several APIs have additional functions that allow a CCSID to be set to indicate the encoding of the parameters being passed. Conversion functions between UTF-8 and EBCDIC have been added. Be sure to review APIs used by your module to be aware of current changes.

Follow the below directions to compile and use a new module.

### 1. Save the source code

Save the source code in your QSYS or IFS directory. All objects created from compiling and creating the service program must be placed in the QSYS directory.

### 2. Compile the source code

Compile the source code using the CRTCMOD command. Before you compile the program, make sure you have the correct programming language compiler installed on your IBM i server (the most common programming language used is C). Replace the text in the parenthesis ( ) with your own information.

`CRTCMOD MODULE(Destination module name and library for the compiled module object.)`

Any Apache modules will need to be changed in order to run as a UTF-8 based server module as opposed to an EBCDIC based server module.

- For ILE C use:

```
CRTCMOD MODULE(MYLIB/MOD_TEST) SRCSTMF('/mydir/mymodule/source/mod_test.c')
DEFINE(AS400 AS400 UTF8) LOCALETYPE(*LOCALEUTF) TERASPACE(*YES)
INCDIR('/qibm/proddata/http/include')
```

- For C++ use:

```
CRTCPMOD MODULE(MYLIB/MOD_TEST) SRCSTMF('/mydir/mymodule/source/mod_test.c')
DEFINE(AS400 AS400 UTF8) LOCALETYPE(*LOCALEUTF) TERASPACE(*YES)
INCDIR('/qibm/proddata/http/include')
```

Notice the change in the LOCALETYPE parameter. Using LOCALETYPE(\*LOCALEUTF) does the following: Program objects created with this option use the locale support provided by \*LOCALE objects. Wide-character types contain four-byte UTF-32 values. Narrow character types contain UTF-8 values. The effect of this change enables the locale dependent C runtime functions to work on UTF-8 strings. See WebSphere Development Studio: ILE C/C++ Programmer's Guide for more information.

Correct any errors found while compiling. Continue to compile the source code until there are no errors. Save the compiled module in the QSYS directory.

### 3. Create a service program

Create a service program using the CRTSRVPGM command. Replace the text in the parenthesis ( ) with your own information.

```
CRTSRVPGM SRVPGM(Destination service program name and library.)
MODULE(Module or modules to be built into the service program. Same as CRTCMOD above.)
EXPORT(Name of the data item to be exported.)
BNDSRVPGM(Specifies other service programs needed to bind to when creating the service program.)
```

**Note:** The **EXPORT** field can only have the value of either **\*ALL** or **\*SRCFILE**. If **\*SRCFILE** is used, you will need to have an export source file defining which data items or procedures need to be exported and contain the name of the module structure (for example, `cgi_module`).

The **BNDSRVPGM** field must have, at a minimum, the following: (**QHTTSPVR/QZSRAPR QHTTSPVR/QZSRCORE QHTTSPVR/QZSRXMLP QHTTSPVR/QZSRSDBM** ). These values will cover all the HTTP Sever APIs that may be used when building the service program.

#### 4. Add LoadModule to HTTP Server configuration file


See “Setting up Apache modules” on page 136 for the steps you need to perform to add the `LoadModule` directive.

##### Related information:

“Apache module APIs” on page 137

This topic provides information about the Apache portable runtime (APR) and application programming interfaces (APIs) for the IBM HTTP Server for i. These APIs are generally used to write cross-platform Apache modules.

 [Developer Documentation for Apache 2.0](#)

 [Apache Portable Runtime Project](#)

## Handler for HTTP Server

In the IBM HTTP Server for i, a handler is an internal representation of the action that is performed when a file or URL is requested.

Generally, files have implicit handlers, based on the file type. Normally, all files are simply served by the server, but certain file types are handled separately. For example, you may use a type of `application/x-httpd-cgi` to invoke CGI scripts.

Handlers are unrelated to file type. They are either based on filename extensions or on location. This allows both a type and a handler to be associated with a file (see [Files with Multiple Extensions](#)).

Handlers are either built into the server, built into a module, or are added with the `Action` directive. The built-in handlers are:

- **default-handler:** Send the file using the `default_handler()`, which is the handler used by default to handle static content. (`core`)
- **send-as-is:** Send file with HTTP headers as is (`mod_asis`).
- **cgi-script:** Treat the file as a CGI script (`mod_cgi`).
- **imap-file:** Imagemap rule file (`mod_imap`).
- **type-map:** Parse as a type map file for content negotiation (`mod_negotiation`).
- **proxy-server:** Determine if file is local (`mod_proxy`)

## Server-side scripting languages

The IBM HTTP Server for i supports the extension of the functionality of the HTTP Server through the use of scripting languages that run on the server.

##### Related concepts:

“Service-side includes” on page 43

Server-side includes (SSI) are the simplest way to add dynamic content to a Web site. A set of directives is embedded in the HTML code and is interpreted by the server before the document is sent to a client. SSI can be used to call a CGI program or return information about documents or the value of environment variables.

## Net.Data

Net.Data is a server-side scripting engine that allows you to easily create dynamic documents using live data from a variety of sources such as relational and non-relational database management systems (DBMSs), including DB2 databases that can be accessed through DRDA, files, and native applications written in programming languages such as RPG, Cobol, Java, C, C++, and REXX.


Net.Data operates on scripts called macros, which contains a series of statements that are defined by the Net.Data macro language. These statements can include standard HTML (or XML, etc.) and language environment-specific statements (for example, SQL statements) as well as macro directives. These statements act as instructions to the Net.Data macro processor, telling it how to construct the dynamic page. Net.Data interprets the statements to create dynamic Web pages with customized content based on input from the user, the current state of your databases, other data sources, existing business logic, and other factors that you design into your macro. The dynamic page that is generated can be rendered in a variety of formats. For example, HTML for browser clients, XML for browser and application clients, wireless markup language (WML) for wireless clients, and Excel for application clients.

The Net.Data macro processor communicates with the HTTP Server through its CGI-BIN interface. Like other CGI-BIN programs, Net.Data is typically stored in the server's CGI-BIN directory. Net.Data is accessed when a URL received by the server refers to the Net.Data macro processor executable, DB2WWW, in the CGI-BIN directory.

When a URL is received by the server that refers to the Net.Data macro processor program, the server starts an instance of the macro processor. It then passes essential information, including the name of the requested macro and the section of the macro to use. The macro processor then:

1. Reads and parses through the macro.
2. Interprets all the macro statements.
3. Dynamically builds the page.
4. Sends the data to the HTTP server by writing to stdout.

The macro writer has complete control over what format the generated data is in (for example: HTML or XML). The macro processor imposes no restrictions. After the text is passed back to the server, the macro processor ends. The resulting text is passed to the client (or browser) where the user interacts with it. Further requests from this user or any other user will result in the whole process just described taking place again.


For more detailed information about Net.Data, including how to configure Net.Data and how to write Net.Data macros and language environments, see the IBM Net.Data for i  Web site.

## PHP

Hypertext Preprocessor (PHP) is one of the world's most popular server-side scripting language for building dynamic, data-driven Web applications.

PHP is a powerful, open, and easy-to-use Web application environment that has the support of a large community with thousands of applications and components to share. It is an open source scripting language that is designed for Web application development. PHP is widely used for content management, customer relationship management, database access, forums, blogs, wikis, and other Web-based applications.

PHP applications are easily integrated with data in IBM DB2 for i and RPG, COBOL, and other business applications on IBM i.

If you want to run PHP scripts, you will need the PHP Zend Core PHP runtime. See the Zend and IBM i  product Web page for information about Zend Core for IBM i.

| **Related information:**

- |  [Configure FastCGI Support for PHP Processing](#)

## Running Java Web applications

Java servlets and Java server pages (JSPs) are Java programs that run on a Java application server and extend the capabilities of the Web server.





Java servlets are Java classes that are designed to respond to HTTP requests in the context of a Web application.

You can look at JSPs as an extension of HTML that gives you the ability to seamlessly embed snippets of Java code within your HTML pages. These bits of Java code generate dynamic content, which is embedded within the other HTML/XML content. A JSP is translated into a Java servlet and executed on the server. JSP statements embedded in the JSP become part of the servlet generated from the JSP. The resulting servlet is executed on the server.

The HTTP Server does not run Java Web applications directly. HTTP requests for Java applications are forwarded by the HTTP Server to Java application servers. IBM provides the following Java application servers to run Java applications:

- **WebSphere Application Server**  
IBM's strategic Web application server and provides enterprise level support for Java servlets, JSPs, and EJBs (Enterprise Java Beans).
- **Integrated Web Application Server**  
A lightweight application server for Java applications that is integrated into the IBM i operating system.


**Related information:**

-  [JavaServer Pages Technology](#)
-  [Java Servlet Technology](#)
-  [WebSphere Application Server for IBM i](#)
-  [Integrated Web Application Server](#)



---

## Troubleshooting

This topic lists common problems and solutions for the IBM HTTP Server for i, the IBM Web Administration for i, and other features associated with the product.

**Important:** Information for this topic supports the latest PTF levels for IBM HTTP Server for i. It is recommended that you install the latest PTFs to upgrade to the latest level of IBM HTTP Server for i. See the IBM HTTP Server for i Support  Web page for more information.


**Related information:**

-  [IBM HTTP Server for i FAQs](#)
-  [IBM HTTP Server for i Support](#)

## Troubleshooting Web Administration for i

This topic lists common problems and solutions for the IBM Web Administration for i, and other features associated with the product.



**Important:** Information for this topic supports the latest PTF levels for IBM HTTP Server for i. It is recommended that you install the latest PTFs to upgrade to the latest level of IBM HTTP Server for i. See the IBM HTTP Server for i Support  Web page for more information.

#### List of symptoms:

- “Symptom: Cannot read or write to QUSRSYS/QATMHINSTC”
- “Symptom: Web browser problems with HTTP Server”
- “Symptom: ADMIN server will not start” on page 204
- “Symptom: HTTP Server will not start or functions will not work” on page 204
- “Symptom: Unknown server type when working with HTTP Servers in ADMIN” on page 205
- “Symptom: All servers show status 'Stopped'” on page 205
- “Symptom: Cannot access ADMIN or some functions do not work” on page 205
- “Symptom: User Profile does not have \*IOSYSCFG” on page 205
- “Symptom: Cannot create new HTTP Server instance” on page 205
- “Symptom: Net.Data error” on page 205
- “Symptom: Error occurred opening file” on page 205
- “Symptom: Databases fail to deploy when configuring with the Web Administration for i interface” on page 205

### Symptom: Cannot read or write to QUSRSYS/QATMHINSTC

#### Cause

The Web Administration for i interface uses the IBM Toolbox for Java. When reading and writing files in QSYS, the Java Toolbox sometimes uses the DDM server. If the DDM server is not running, this may result in problems reading or writing the QUSRSYS/QATMHINSTC file containing HTTP Server definitions.

#### Solution

On an IBM i command line, enter STRTCPSVR \*DDM.

### Symptom: Web browser problems with HTTP Server

#### Cause

Your Web browser may not be configured correctly.

#### Solution

Below is a list of common problems and solutions for your Web browser.

#### Miscellaneous Microsoft Internet Explorer errors related to incorrect interpretation of HTTP/1.1 in response

Microsoft Internet Explorer sends requests in HTTP/1.1 format but seems to only accept responses in HTTP/1.0 format. The work around is to tell HTTP Server the request came in as HTTP/1.0 format.

Fore example: BrowserMatch "MSI 4\.0b2;" downgrade-1.0 force-response-1.0

#### URL not found when clicking on a file in a directory listing from Netscape

If AlwaysDirectoryIndex is set to OFF and a URL for a directory without a trailing slash is requested, then Netscape does not request the file relative to the director in which the file exists in the resulting directory listing.

#### Microsoft Internet Explorer does not display customized error messages

If Internet Explorer is not displaying the customized error messages, check to see if the preferences for the browser are set to show friendly HTTP error messages. Disable this preference and the customized error messages should display properly.

**When using HTTPS, Microsoft Internet Explorer shows pages that were cached when using HTTP** If the browser is showing cached pages instead of connecting to the server using SSL, clear the browser's cache.

**Prompted for password when using certificate for client authentication**

If you are using a Certificate Authority that offers the option to protect the private key of your certificate with a password (such as for the Microsoft Internet Explorer browser), and you use the certificate for client authentication, you are prompted for the password after about 2 minutes of idle time. This happens even if you have disabled SSLV2 in the browser being used and in the server because you are trying to use the longer SSLV3 cache time-out interval. This is a security feature that protects your private key if you are away from your client, even though it may look like an SSLV3 caching problem.

**Certificate not recognized by browser**

If you add a certificate to your browser, the browser may not recognize that there is a new certificate until you restart your computer.

**Symptom: ADMIN server will not start**

**Solution**

Check to make sure you have the proper authorities. See "User profiles and required authorities for HTTP Server" on page 30 for specific authority and profile information.

**Symptom: HTTP Server will not start or functions will not work**

**Solution**

General items to check:

1. Check /QIBM/UserData/HTTPA/admin/logs, HTTPAdmin.log, error\_log, and any other logs you may have. More information on the cause of the problem may be found there.
2. Use CHKPRDOPT to 57XXDG1, SS1, TC1 and JV1.
3. Check joblog for user QTMHHTTP.
4. Check QTMHHTTP and QTMHHTTP1 user profiles.
5. Verify that \*PUBLIC is not \*EXCLUDEd from '/' (Use WRKLNK '/' and take option 9).
6. Verify that QSERVER and QUSRWRK subsystems are running.

**Error messages:**

**Error ZSRV\_MSG0358**

Found in admin log. Verify that there is a host table entry in CFGTCP Option 10 that matches the host + domain name in CFGTCP Option 12, and set 'Host Name Search Priority' to \*LOCAL.

**Error ZUI\_50004 - 'no \*IOSYSCFG authority'**

Verify that user has \*IOSYSCFG Authority. If \*IOSYSCFG is granted by a GROUP profile, verify that PTF SF65588 (V4R5) is applied. Check that there are NO user .jar files in the /QIBM/ProdData directory path - this directory is for IBM use only.

**Error HTP8015**

Verify that the latest PTFs for DG1 product are applied.

**Error CEE0200**

Verify that 57XXJV1 Options \*Base, 5, and 6 are installed,

**Error ZSRV\_MSG0302 :User qsecofr:authentication failure for "":1**

Known problem with 128 character passwords on V5R1. HTTP servers cannot use 128 character passwords. You may be able to circumvent this problem by changing the password in the user profile to CAPITAL letters and using CAPITAL letters to log into the ADMIN screen.

## **Symptom: Unknown server type when working with HTTP Servers in ADMIN**

### **Solution**

Ensure that LOOPBACK and LOCALHOST are configured to resolve to 127.0.0.1 and can be PINGed from the IBM i command line. Verify that there are no exit programs for exit point QIBM\_QPWFS\_FILE\_SERV. Verify that QSERVER and QUSRWRK subsystems are running and that current group PTF for DG1 product is applied.

## **Symptom: All servers show status 'Stopped'**

**Cause** This problem was determined to be caused by an OEM security application that registers many exit point programs.

### **Solution**

Remove the application to eliminate the problem.

## **Symptom: Cannot access ADMIN or some functions do not work**

### **Solution**

Verify the following:

- Verify that user's browser is not using a proxy to access the ADMIN server.
- Verify latest DG1 PTF's.
- Verify that user profiles QTMHHTTP and QTMHHTTP1 are enabled.

## **Symptom: User Profile does not have \*IOSYSCFG**

### **Solution**

In the HTTPAdmin.log you will find error: 'NoRouteToHostException'. Do the following:

- Verify that 127.0.0.1, LOOPBACK and LOCALHOST are configured and work.

## **Symptom: Cannot create new HTTP Server instance**

### **Solution**

Verify LOCALHOST , LOOPBACK and 127.0.0.1 exist and work.

## **Symptom: Net.Data error**

Include object specified in /QIBM/ProdData/HTTPSVR/MRISXX/Macro/qzhamsg.nds at line 208

### **Solution**

Verify that directory /QIBM/ProdData/HTTPSVR/Macro/ contains only objects that are appropriate to the current OS version .

## **Symptom: Error occurred opening file**

**Cause** If your HTTP Server configuration uses the Rewrite directive and does not have the proper access for QTMHHTTP configured, your server will not start.

### **Solution**

Make sure QTMHHTTP has \*RWX access authority to the /tmp directory.

## **Symptom: Databases fail to deploy when configuring with the Web Administration for i interface**

**Cause** This error occurs when the user ID selected as the WebSphere Portal database owner does not have authority to the CHGJOB command. The configuration wizard requires this authority to autoreply when the system would otherwise require a response from the user. Without this authority, the create-all-db configuration task fails, and databases are not deployed.

If databases fail to deploy when configuring WebSphere Portal with the IBM Web Administration for i interface, check the /QIBM/UserData/Webas5/Base/<instance>/logs/<instance>/WPSWIZARD\_<timestamp>\_create-all-db.log log file for the following error:

```
[java] java.lang.RuntimeException: error when creating statement [CPF0006]
Errors occurred in command.
[java] java/lang/Throwable.(Ljava/lang/String;)V+4 (Throwable.java:85)
[java] java/lang/Exception.(Ljava/lang/String;)V+1 (Exception.java:33) [java]
java/lang/RuntimeException.(Ljava/lang/String;)V+1 (RuntimeException.java:38)
[java] com/ibm/wps/config/SqlProcessor.process([Ljava/lang/String;Ljava/lang/
String;Ljava/lang/String;Ljava/lang/String;Ljava/lang/String;)I+0
(SqlProcessor.java:78)
[java] com/ibm/wps/config/SqlProcessor.main([Ljava/lang/String;)V+0
(SqlProcessor.java:478)
```

### Solution

Ensure the user ID selected as the WebSphere Portal database owner has authority to the CHGJOB command.

### Related information:

“Troubleshooting HTTP Server”


This topic lists common problems and solutions for the IBM HTTP Server for i and other features associated with the product.

[IBM HTTP Server for i FAQs](#)

[IBM HTTP Server for i Support](#)

## Troubleshooting HTTP Server

This topic lists common problems and solutions for the IBM HTTP Server for i and other features associated with the product.

**Important:** Information for this topic supports the latest PTF levels for IBM HTTP Server for i. It is recommended that you install the latest PTFs to upgrade to the latest level of IBM HTTP Server for i. See the IBM HTTP Server for i Support  Web page for more information.

### List of symptoms:

- “Symptom: Error 404 on HTTP Server”
- “Symptom: HTTP Server has a slow response” on page 207
- “Symptom: Error 500 on HTTP Server” on page 207
- “Symptom: HTTP Server on port 80 does not start” on page 207
- “Symptom: Web browser problems with HTTP Server” on page 208
- “Symptom: Error occurred opening file” on page 209
- “Symptom: WebSphere Portal authentication performance problems” on page 210

## Symptom: Error 404 on HTTP Server

### Cause

HTTP Server is not able to find the resource that was requested or the user profile on HTTP Server does not have authority to the requested resource.

### Solution

Check the following:

- Make sure the file exists.
- Make sure that the user profile used to access the resource has object authority. The user profile QTMHHTTP is used by default. The user profile QTMHHTTP1 is used by default when the request is a CGI program.

## Symptom: HTTP Server has a slow response

### Solution

Refer to the following:

- “Managing HTTP Server performance” on page 105

## Symptom: Error 500 on HTTP Server

### Cause

A program on your HTTP Server has failed or there is an error in your CGI program.

### Solution

Check the following:

- Check the server Primary job log, QSYSOPR messages, error log and CGI job logs for more information.
- If you have not used the IBM Web Administration for i interface to create an HTTP Server configuration, a required directive may be missing from the configuration file. View the configuration file with the Web Administration for i interface for possible errors.

## Symptom: HTTP Server on port 80 does not start

### Cause

By default, APACHEDFT server autostart setting is \*GLOBAL. If, in addition, the global server setting for autostart is "Yes", then APACHEDFT server will start during STRTCP command processing. APACHEDFT server uses port 80 and may cause any other HTTP Server using port 80 to not start.

### Solution

Do the following:

If your HTTP Server does not start or appears to start, but then stops, check the following:

1. The cause of the problem may be in the job log. Use WRKACTJOB immediately after the server is started. If the job is active, then enter **WRKACTJOB** to work with job and display the job log. If the job is not active, then enter **WRKSPLF SELECT(QTMHHTTP)** to find the name of the server and display the spool file.
2. If you have configured the error logs, then the cause of the problem may be in the error log. For example, /www/myserver/logs/basic\_error\_log, where "myserver" is the name of your HTTP Server.

**Note:** If the error messages have been customized, the error will not be identified in the same manner as the above example.

If these steps do not help, then try starting the server with verbose tracing. See Manage server performance for HTTP Server for tracing.

By default, APACHEDFT server autostart setting is \*GLOBAL. If, in addition, the global server setting for autostart is "Yes", then APACHEDFT will start during STRTCP command processing. APACHEDFT server uses port 80 and may cause any other HTTP Server using port 80 to not start. To avoid this condition, you can :

- Change APACHEDFT server configuration autostart setting to "No".
- Change APACHEDFT server configuration to use a port other than 80.

To change the autostart value on APACHEDFT server, do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select **APACHEDFT** from the **Server** list.
4. Expand **Server Properties**.

5. Click **General Server Configuration**.
6. Click the **General Settings** tab in the form.
7. Select **No** (instead of \*GLOBAL or Yes) from the **Autostart** list.
8. Click **OK**.

To change the port number on APACHEDFT server, do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select **APACHEDFT** from the **Server** list.
4. Expand **Server Properties**.
5. Click **General Server Configuration**.
6. Click the **General Settings** tab in the form.
7. Select the IP address and port from the **Server IP addresses and ports to listen on** table.
8. Enter a new value for the port number in the **Port** column.
9. Click **Continue**.
10. Click **OK**.

As a final precaution, make sure APACHEDFT server is not started by doing the following:

1. Click the **Manage** tab.
2. Click the **All Servers** subtab.
3. Click the **All HTTP Servers** tab.
4. Select **APACHEDFT** from the table.
5. Click **Stop**.

## Symptom: Web browser problems with HTTP Server

### Cause

Your Web browser may not be configured correctly.

### Solution

Below is a list of common problems and solutions for your Web browser.

#### Miscellaneous Microsoft Internet Explorer errors related to incorrect interpretation of HTTP/1.1 in response

Microsoft Internet Explorer sends requests in HTTP/1.1 format but seems to only accept responses in HTTP/1.0 format. The work around is to tell HTTP Server the request came in as HTTP/1.0 format.

Fore example: BrowserMatch "MSI 4\.\0b2;" downgrade-1.0 force-response-1.0

#### URL not found when clicking on a file in a directory listing from Netscape

If AlwaysDirectoryIndex is set to OFF and a URL for a directory without a trailing slash is requested, then Netscape does not request the file relative to the director in which the file exists in the resulting directory listing.

#### Microsoft Internet Explorer does not display customized error messages

If Internet Explorer is not displaying the customized error messages, check to see if the preferences for the browser are set to show friendly HTTP error messages. Disable this preference and the customized error messages should display properly.

#### When using HTTPS, Microsoft Internet Explorer shows pages that were cached when using HTTP

If the browser is showing cached pages instead of connecting to the server using SSL, clear the browser's cache.

#### Prompted for password when using certificate for client authentication

If you are using a Certificate Authority that offers the option to protect the private key of

your certificate with a password (such as for the Microsoft Internet Explorer browser), and you use the certificate for client authentication, you are prompted for the password after about 2 minutes of idle time. This happens even if you have disabled SSLV2 in the browser being used and in the server because you are trying to use the longer SSLV3 cache time-out interval. This is a security feature that protects your private key if you are away from your client, even though it may look like an SSLV3 caching problem.

#### **Certificate not recognized by browser**

If you add a certificate to your browser, the browser may not recognize that there is a new certificate until you restart your computer.

### **Symptom: HTTP Server will not start or functions will not work**

#### **Solution**

General items to check:

1. Check /QIBM/UserData/HTTPA/admin/logs, HTTPAdmin.log, error\_log, and any other logs you may have. More information on the cause of the problem may be found there.
2. Use CHKPRDOPT to 57XXDG1, SS1, TC1 and JV1.
3. Check joblog for user QTMHHTTP.
4. Check QTMHHTTP and QTMHHTTP1 user profiles.
5. Verify that \*PUBLIC is not \*EXCLUDEd from '/' (Use WRKLNK '/' and take option 9).
6. Verify that QSERVER and QUSRWRK subsystems are running.

#### **Error messages:**

##### **Error ZSRV\_MSG0358**

Found in admin log. Verify that there is a host table entry in CFGTCP Option 10 that matches the host + domain name in CFGTCP Option 12, and set 'Host Name Search Priority' to \*LOCAL.

##### **Error ZUI\_50004 - 'no \*IOSYSCFG authority'**

Verify that user has \*IOSYSCFG Authority. If \*IOSYSCFG is granted by a GROUP profile, verify that PTF SF65588 (V4R5) is applied. Check that there are NO user .jar files in the /QIBM/ProdData directory path - this directory is for IBM use only.

##### **Error HTP8015**

Verify that the latest PTFs for DG1 product are applied.

##### **Error CEE0200**

Verify that 57XXJV1 Options \*Base, 5, and 6 are installed,

##### **Error ZSRV\_MSG0302 :User qsecofr:authentication failure for "":1**

Known problem with 128 character passwords on V5R1. HTTP servers cannot use 128 character passwords. You may be able to circumvent this problem by changing the password in the user profile to CAPITAL letters and using CAPITAL letters to log into the ADMIN screen.

### **Symptom: Error occurred opening file**

**Cause** If your HTTP Server configuration uses the Rewrite directive and does not have the proper access for QTMHHTTP configured, your server will not start.

#### **Solution**

Make sure QTMHHTTP has \*RWX access authority to the /tmp directory.

## Symptom: WebSphere Portal authentication performance problems



If you are experiencing performance problems when users are logging into Portal (the authentication phase), the following indicators may help determine that the filters are causing these performance problems:

- Your LDAP server is populated with a large number of entries.
- When you type `WRKACTJOB` in a console command line, `QSQRVR` jobs are using an excessive amount of CPU during the Portal authentication (sign on) phase.
- When two Portal users sign on concurrently, one sign on request takes two times as long as the other request.

**Cause** You may encounter a performance problem if you configure a secure WebSphere Portal server with LDAP. This problem only occurs if you use the **Create WebSphere Portal wizard** in the Web Administration for i interface. When configuring LDAP with the WebSphere Portal wizard, the two LDAP fields **LDAPUserFilter** and **LDAPGroupFilter** are configured with default values depending on the type of LDAP server being used. For example, if you are securing your WebSphere Portal server using the IBM Directory Server, the two LDAP fields are set to `"(&(|(cn=%v)(uid=%v))(objectclass=person))"` and `"(&(cn=%v)(|(objectclass=groupOfUniqueNames)(objectclass=groupOfNames)(objectclass=group)))"`, respectively. By configuring the fields with the default values, the WebSphere Portal wizard allows the `wpsadmin` Portal administrator to successfully login and existing LDAP entries can be used once the Portal server is successfully configured and secured. However, if the LDAP server has a large number of entries, or if many additional users are added to the LDAP server, Portal's authentication performance may be noticeably impacted.

### Solution

If you determine that the filters, as configured by the WebSphere Portal wizard, are causing authentication performance problems, complete the following steps:

1. Start the Web Administration for i interface.
2. Click the **Manage** tab.
3. Click the **Application Servers** subtab.
4. Expand **Tools**.
5. Click **Launch Administrative Console**.
6. Login to the console and click **OK**.
7. Expand **Security**.
8. Expand **User Registries**.
9. Click **LDAP**.
10. Click **Advanced LDAP Settings** in the **Additional Properties** table.
11. Edit the **User Filter** and the **Group Filter** properties values to more precise values to increase authentication performance. For more information about this syntax, see the IBM Tivoli Directory Server for i (LDAP) and the WebSphere Portal and Lotus Web Content Management  Web site.
  1. Edit the **User Filter** and the **Group Filter** properties values to more precise values to increase authentication performance. For more information about this syntax, see the IBM Tivoli Directory Server for i (LDAP) and the WebSphere Portal and Lotus Web Content Management  Web site.
  2. Click **OK**.
  3. Click **Save** to apply changes to the master configuration.
  4. Click **Save** again on the next page.



**Note:** You may need to restart your WebSphere Application Server for these changes to take affect.

**Related information:**

Application servers

Business solutions

 [IBM HTTP Server for i FAQs](#)

 [IBM HTTP Server for i Support](#)

 [WebSphere Portal and Lotus Web Content Management](#)

## Troubleshooting CGI programs

This topic lists common CGI program problems and solutions.


You can use the Work with Active Jobs (WRKACTJOB) command to check on the status of server jobs. To start Work with Active Jobs command, type the following in during a 5250 session on a command line:

```
WRKACTJOB SBS(QHTTSPVR) JOB(server_instance)
```

Where *server\_instance* is the name of your HTTP Server instance.

When the server is not processing a request, the Work with Active Jobs display will show several server jobs. The first job is the manager job for the server instance. (Function PGM-QZHBMAIN). Server jobs showing PGM - QZSRLOG are logging jobs. Server jobs showing PGM - QZSRHTTP are primary jobs. (There will be 2 of these unless you specify HotBackup Off in your configuration.) Only one of these jobs will be actively handling requests. Jobs showing PGM -QZSRCGI are CGI jobs.

To find out if server jobs have ended abnormally, check the spooled files that contain the job logs (QPJOBLOG) for the user profile QTMHHTTP.

More CGI troubleshooting tips and hints can be found at the Troubleshooting your CGI program  Web page on the HTTP Server Web site.

The symptoms that are described in this section would be seen running a request to the server at a browser.

### Symptom: Connection abandoned, dropped, or no data sent

**Note:** Different browser issues different messages when no data is returned to the browser. Abandoned, dropped or no data will be displayed at the browser.

**Cause:** The system has incorrectly formatted a CGI program that writes data to standard output. The data that is written to stdout may have one of the following problems:

- 
- No data written to stdout
- No "Content-type", "Location", or "Status" line
- No new line character after HTTP response header
- No data after HTTP response header.

**Solution:** Write the data to stdout with "Content-type: " line with two new line characters ("\n") and the data to be returned to the client. For example:

```
Content-type: text/plain\n\nThis data is returned to the client
```

**Cause:** CGI program caused an exception message that was not handled by the CGI program.

**Solution:** If the system does not indicate a message in the joblog for the active server jobs, do a WRKSPLF QTMHHTTP. Check for server jobs that ended when the system ran the CGI program. Change the program to monitor for the message not being handled.

**Cause:** The program being called does not exist in the library.

**Solution:** Check the library for the correct name.

**Cause:** There is a bug in your user-created CGI program.

**Solution:** You need to set up a scaffolding environment to debug the CGI application prior to integration with server:

1. Issue the command `ENDTCPSVR *HTTP HTTPSVR(server_instance)`
2. Issue the command `STRTCPSVR *HTTP HTTPSVR(server_instance '-minat 1 -maxat 1')`

**Note:** You also may need to change `script_timeout` and `output_timeout` to be larger. If you are stepping through your code, it may take too long and `script_timeout` or `output_timeout` may expire. This causes the server to terminate the job you are debugging.

Ending and starting the server ensures that only one worker job is running.

- a. Issue the command `WRKACTJOB JOB(server_instance)`

Look for the CGI jobs as described above.

Select **option 10** to display the job log.

If your CGI program is single thread capable, message HTP2001 will be in the job log. If your CGI program is multithread capable, message HTP2002 will be in the job log.

Record the *Number:*, *User:*, and *Job:* values for your CGI program job.

Press **F12**.

Issue the command `STRSRVJOB <Number/User/Job>`.

- b. For the user CGI program, issue the command `STRDBG <usercgilib/cgipgm>`

If the program accesses a database file on the server, you must specify `UPDPROD(*YES)`. See the help for the `STRDBG` command.

**Note:** You will need additional authority to troubleshoot the CGI program. For example, you will need authority to the QTMHHTTP user profile.

- c. Set breakpoints in the program.
- d. On the browser, issue a URL that would run the CGI program.
- e. After the system issues an HTTP request on the browser, return to the session that ran `STRSRVJOB`. It should have stopped at a program breakpoint.

Ending and starting the server ensures that only one worker thread is running.

3. When finished with debug, reset the server values:

- a. Issue the command `ENDDBG`
- b. Issue the command `ENDSRVJOB`
- c. Issue the command `WRKACTJOB SBS(QHTTPSVR) JOB(server_instance)`
- d. Issue the command `STRTCPSVR *HTTP HTTPSVR(server_instance)`

## **Symptom: The system is not converting or handling special characters as expected**

**Cause:** The browser inserts special characters using escape sequences which requires special handling by the CGI program.

**Solution:** Browsers create escape sequences (ISO 8859) for special characters (for example, : , ! @ # \$ % \* , and so on.) These characters come into standard input or into the QUERY\_STRING environment variable in the form “%xx”, where “xx” is the two characters representing the ASCII hexadecimal value. (For example, a comma comes in as “%2C”. For CGI input mode %%MIXED%%, these three characters “%xx” are converted to EBCDIC, but the values of “xx” are not changed to the corresponding EBCDIC code points.

There are two approaches to handling escape sequences:

1. Convert the EBCDIC representation of the ASCII escape sequence to an EBCDIC escape sequence or use CGI input mode %%EBCDIC%%. This is necessary because the QtmhCvtDB API assumes that escape sequences represent EBCDIC code points, and the API converts them to the corresponding EBCDIC character. For example, %2C, which represents an ASCII comma, is converted to EBCDIC X'2C', which is not an EBCDIC comma.
2. Convert the EBCDIC representation of the ASCII escape sequence to the EBCDIC equivalent character.

The following approach outlined in the first conversion technique listed above:

**Note:** The hex representation of the %2C from the browser was 0x253243. When this escape sequence is converted to EBCDIC, it ends up as 0x6CF2C3.

1. Convert the “xx” in “%xx” to the corresponding EBCDIC character. In this case 0xF2C3 is converted to 0x2C.
2. For the first approach, convert the EBCDIC character to the two-byte form. Then you can reinsert the two bytes back into the input stream in the same place they originally appeared. The 0x6B would be converted to 0xF6C2, and the resultant escape sequence would be 0x6CF6C2. For the second approach, leave the data in its EBCDIC form and replace the original escape sequence (three characters) with the single character. In this case, replace 0x6CF2C3 with 0x6B.

**Note:** The CGI program should preserve an escape sequence that represents the character “%”.

3. Call QtmhCvtDB to convert the input stream.

**Note:** 7-bit ASCII CCSID 367 is standard on browsers.

## **Symptom: Error 500: Bad script request -- script '/qsys.lib/qsyscgi.lib/progname.pgm' not found or not executable**

**Cause:** Configuration or authority error.

This message can appear for the following reasons:

- The script does not exist.
- There is a problem with the script, for example, a send error or function check.
- The user QTMHHTTP1 does not have authority to run this program.

**Solution:** Check the configuration and authorities given to the CGI program.

## **Symptom: A browser request that runs a CGI program runs longer than expected. The browser keeps waiting for a response**

**Cause:** The CGI application that was running has taken a function check.

**Solution:** Look at the QSYSOPR message queue for a message that requires a reply sent from the CGI program that was running. Note the statement where the program is failing. Use the procedure described under “Symptom: Error 500”.

### **Symptom: A CGI written form is not cached in the browser**

Using the back button on the browser results in a request to the server. The form contains no headers or meta tags telling the browser to request (not cache) the page.

**Cause:** The server is sending a last-modified header.

**Solution:** Use the `—nolastmod` HTTP Server startup value to specify that the server should not send a last-modified header.

### **Symptom: The configuration uses the CGIConvMode value of %%MIXED/MIXED%% and the input characters your CGI program receives are incorrect**

**Cause:** The file CCSID language for your server has characters that do not match the EBCDIC code page 37. Use the EBCDIC mode rather than the MIXED mode.

**Solution:** Configure CGIConvMode for `%%EBCDIC/MIXED%%`.

#### **Related information:**

“Troubleshooting HTTP Server” on page 206

This topic lists common problems and solutions for the IBM HTTP Server for i and other features associated with the product.

 [IBM HTTP Server for i FAQs](#)

 [IBM HTTP Server for i Support](#)

---

## **Reference information for HTTP Server**

This topic provides additional reference documentation for IBM HTTP Server for i and the IBM Web Administration for i interface.

See “Related information for HTTP Server” on page 507 for additional reference documentation.

### **Directives for HTTP Server**

This topic provides information about the supported directives for IBM HTTP Server for i.

The supported modules can be found in the HTTP Server directive finder.

See “Directives no longer supported on HTTP Server” on page 216 for modules no longer supported for this version of HTTP Server.

**Note:** This information is provided for reference only. Use the IBM Web Administration for i to set up and manage your HTTP Server.

### **Directive term definitions for HTTP Server**

This topic provides information about the directive terms used for IBM HTTP Server for i.

Each configuration directive is described using the following attributes:

**Module:** directive existence

**Syntax:** directive\_name arguments

**Default:** directive\_name default\_value

**Context:** context\_list

**Override:** directive override activation

**Origin:** origin

**Usage Considerations:** important usage considerations required in the server configuration file

**Example:** example of directive and its arguments

## Module

This attribute identifies the module the directive is associated with.

## Syntax

This attribute indicates the format of the directive as it would appear in a configuration file. This syntax is directive-specific, so refer to the text of the directive's other attributes for details. Strings should be quoted. The string ("word1 word2") contains spaces. If the strings do not contain spaces they do not need to be quoted.

## Default

This attribute specifies if the directive has a default value. For example, if you omit the directive from your configuration entirely, HTTP Server will behave as though you set it to a particular value. If there is no default value, this attribute says "none".

## Context

This attribute indicates where in the server's configuration the directive is supported. It's a comma-separated list of one or more of the following values:

### server config

The directive is valid in the global server configuration.

### virtual host

The directive is valid in <VirtualHost> containers.


### directory

The directive is valid in <Directory>, <Location>, and <Files> containers, subject to the restrictions outlined in the "Fundamental directive, context, and server area concepts on HTTP Server" on page 13" topic.

### directory (but not location)

The directive is valid in <Directory>, <Files> containers, subject to the restrictions outlined in the "Fundamental directive, context, and server area concepts on HTTP Server" on page 13" topic, but is not valid in the <Location> container.

### .htaccess

The directive is valid in per-directory .htaccess files. It may not be processed, however, depending upon the overrides currently active. For more information on how to use .htaccess files, see the Apache HTTP Server Project  Web site.

### Not in Limit

The directive is not valid in <Limit> containers, subject to the restrictions outline in the "Fundamental directive, context, and server area concepts on HTTP Server" on page 13" topic.

**All** The directive is valid in all contexts.

**Note:** The directive is only allowed within its supported context; if you try to use it elsewhere, you will receive a configuration error that will either prevent the server from handling requests, or will keep the server from starting. The valid context for a directive is actually the result of a "Boolean

OR" of all of the listed contexts. In other words, a directive that is marked as being valid in "server config, .htaccess" can be used in the server configuration file and in .htaccess files, but not within any <Directory> or <VirtualHost> containers.

## Override

This attribute indicates which configuration override must be active in order for the directive to be processed when it appears in a .htaccess file. If the directive's context does not permit it to appear in .htaccess files, this attribute is none.

## Origin

This attribute reveals the origin of an HTTP directive. Possible values for this attribute include:

**IBM** A new directive created for the IBM HTTP Server for i Web server.

### Modified

An Apache server directive modified to support the IBM HTTP Server for i Web server.

### Apache

An unmodified Apache server directive.

## Usage Considerations

This attribute specifies if important usage considerations such as a LoadModule are required in the server configuration file prior to using the directive. If this attribute is not available, the directive does not require any usage considerations.

## Example

This attribute specifies at least one example for directives that take a file path name as an argument. It will include both a root example and a QSYS.LIB example, if both apply.

## Directives no longer supported on HTTP Server

This topic provides information about what directives are no longer supported by IBM HTTP Server for i.

The following directives are no longer supported on HTTP Server.

### Directives

- "AddModule"
- "ClearModuleList" on page 217
- "IconPath" on page 217
- "Port" on page 218

### AddModule:

**Module:** core

**Syntax:** AddModule *module* [*module* ...]

**Default:** none

**Context:** server config

**Override:** none

**Origin:** Apache

**Example:** AddModule mod\_cgi

The AddModule directive allows the server to activate specific modules in the server after a ClearModuleList has been performed. The server comes with a pre-loaded list of active modules. Only

those modules are valid. A list of valid modules can be obtained using the '-l' option on the command line. The example above would activate the module `mod_cgi`. If this module is already active then the directive will be ignored.

**Parameter:** *module*

- *Module* is any valid module in the pre-loaded list that came with the HTTP Server.

See also “ClearModuleList.”

### **ClearModuleList:**

**Module:** core

**Syntax:** ClearModuleList

**Default:** none

**Context:** server config

**Override:** none

**Origin:** Apache

**Example:** ClearModuleList

The ClearModuleList directive will clear the built-in list of active modules provided by the server. To reactivate this module list use the “AddModule” on page 216 directive.

### **IconPath:**

**Module:** mod\_auto\_index

**Syntax:** IconPath

**Default:** IconPath /icons

**Context:** server config, virtual host, directory, .htaccess

**Override:** none

**Origin:** IBM

**Example:** IconPath /myicons/small/

The IconPath directive to specify URL information to be added at the beginning of each icon-URL specified on the following directives:

- AddIcon
- AddIconByType
- AddIconByEncoding
- DefaultIcon

The value that you specify on this directive is added to the icon-URL value on each of the other directives to form the full request URL for each icon. The following path and directory is the default location for icons:

`/QIBM/ProdData/HTTPPA/icons`

### **Special Usage Considerations:**

- You must enable your server for serving the icons from the default location by adding the following statement to your configuration:

```
Alias /icons /QIBM/ProdData/HTTPPA/icons
```

- You must use this directive in your configuration before any of the other icon directives that are to use the path (DefaultIcon, AddIcon, AddIconByType, and AddIconByEncoding).

For example, a configuration containing:

```
Alias /icons/small /QIBM/ProdData/HTTPPA/icons/small
IconPath /icons/small/
AddIcon blank.gif ^^BLANKICON^^
```

This causes the server to generate a request for the directory list icon as `/icons/small/blank.gif`. The server uses the alias directive to resolve the request to the proper file. This is different from Apache than on other platforms.

On another platform you would use:

```
Alias /icons /full/icon/path
AddIcon /icons/blank.gif ^^BLANKICON^^
```

IconPath is an IBM i specific directive for Apache; therefore, precautions must be taken if the Apache configuration file is modified manually. On the IBM i server, you would use:

```
Alias /icons /QIBM/ProdData/HTTPPA/icons
AddIcon blank.gif ^^BLANKICON^^
```

Since IconPath is set to `/icons/` by default, it will be prepended to 'blank.gif' when the AddIcon directive is used.

## Port:

**Module:** core

**Syntax:** Port *number*

**Default:** Port 80

**Context:** server config

**Override:** none

**Origin:** Apache

**Example:** Port 8080

The Port directive has two behaviors:

- In the absence of any Listen directives specifying a port number, a Port directive given in the "main server" (for example, outside any `<VirtualHost>` section) sets the network port on which the server listens. If there are any Listen directives specifying the port number then Port has no effect on what address the server listens at. The use of the Listen directive causes all Port directives to be ignored.
- The Port directive sets the SERVER\_PORT environment variable (for CGI and SSI), and is used when the server must generate a URL that refers to itself (for example when creating an external redirect to itself). This behavior is modified by UseCanonicalName.

In no event does a Port setting affect what ports a VirtualHost responds on, the VirtualHost directive itself is used for that. The primary behavior of Port should be considered to be similar to that of the ServerName directive. The ServerName and Port together specify what you consider to be the *canonical* address of the server. (See also UseCanonicalName.)

**Parameter:** *number*

- Where *number* is a number from 0 to 65535; some port number (especially below 1024) are reserved for particular protocols. The standard port for http protocol is 80.

**Note:** The "Listen" on page 310 directive is used as an alternative to Port.

## Module mod\_access

Module mod\_access contains directives for the IBM HTTP Server for i Web server.

### Summary

The module mod\_access provides access control based on a client's hostname or IP address.



## Directives

- “Allow”
- “Deny”
- “Order” on page 220

### Allow:

**Module:** mod\_access

**Syntax:** allow from *all* | *env*=[!]*envvar* | *host* [*host* ...]

**Default:** none

**Context:** directory, .htaccess

**Override:** Limit

**Origin:** Apache

**Example:** allow from all

**Example:** allow from env=go\_away

**Example:** allow from 10.10.10.10 .ibm.com

**Example:** allow from 2001:db8::a00:20ff:fea7:ccea

**Example:** allow from 2001:db8::a00:20ff:fea7:ccea/10

The Allow directive affects which hosts can access a given directory.

**Parameter:** *host*

- If *all*, all hosts are allowed access.
- If *full* or *partial domain-name*, hosts whose names match or end in this string are allowed access.
- If *full IP address*, only IP address of a host are allowed access.
- If *partial IP address*, only the first 1 to 3 bytes of an IP address, for subnet restriction.
- If *network/netmask*, a network a.b.c.d. And a netmask w.x.y.z. Can be used for fine-grained subnet restriction (for example, 10.2.0.0/255.255.0.0).
- If *network/nnn CIDR specification*, it is similar to the previous case, except the netmask consists of nnn higher-order 1 bits (for example, 10.1.0.0/16 is the same as 10.1.0.0/255.255.0.0).

**Note:** This compares whole components, *ibm.com*<sup>®</sup> would not match *QIBMibm.com*.

The allow from *env* option controls access to a directory by the existence (or nonexistence) of an environment variable. For example:

```
BrowserMatch ^KnockKnock/2.0 let_me_in
<Directory /docroot>
    order deny,allow
    deny from all
    allow from env=let_me_in
</Directory>
```

In this case browsers with the user-agent string KnockKnock/2.0 will be allowed access, and all others will be denied.

See also “Deny,” “Order” on page 220, and BrowserMatch.

### Deny:

**Module:** mod\_access

**Syntax:** deny from *all* | *env*=[!]*envvar* | *host* [*host* ...]

**Default:** none

**Context:** directory, .htaccess

**Override:** Limit

**Origin:** Apache

**Example:** deny from env=go\_away

**Example:** deny from 10.10.10.10 .ibm.com

The deny directive affects which hosts can access a given directory.

**Parameter:** *host*

- If *all*, all hosts are denied access.
- If *full* or *partial domain-name*, hosts whose names match or end in this string are denied access.
- If *full IP address*, only IP address of a host are denied access.
- If *partial IP address*, only the first 1 to 3 bytes of an IP address, for subnet restriction.
- If *network/netmask*, a network a.b.c.d. And a net mask w.x.y.z. Can be used for fine-grained subnet restriction (for example, 10.2.0.0/255.255.0.0).
- If *network/nnn CIDR specification*, it is similar to the previous case, except the netmask consists of nnn higher-order 1 bits (for example, 10.1.0.0/16 is the same as 10.1.0.0/255.255.0.0).

**Note:** This compares whole components (ibm.com would not match *QIBMibm.com*).

The deny from env option controls access to a directory by the existence (or nonexistence) of an environment variable. For example:

```
BrowserMatch ^BadRobot/0.9 go_away
<Directory /docroot>
    order allow,deny
    allow from all
    deny from env=go_away
</Directory>
```

In this case browsers with the user-agent string BadRobot/0.9 will be denied access, and all others will be allowed.

See also "Allow" on page 219 and "Order."

**Order:**

**Module:** mod\_access

**Syntax:** order *ordering*

**Default:** order deny,allow

**Context:** directory, .htaccess

**Override:** Limit

**Origin:** Modified

**Example:** order deny,allow

The order directive controls the order in which Allow and Deny directives are evaluated. .

**Parameter:** *ordering*

- If *deny,allow*, the deny directives are evaluated before the allow directives (the initial state is OK).
- If *allow,deny*, the allow directives are evaluated before the deny directives (the initial state is FORBIDDEN).
- If *mutual-failure*, only those hosts which appear on the allow list and do not appear on the deny list are granted access (the initial state is irrelevant).

Keywords may only be separated by a comma; no whitespace is allowed between them. Note: that in all cases every allow and deny statement is evaluated, there is no "short-circuiting". For Example:

```
order deny,allow
deny from all
allow from .ibm.com
```

In this example, the first container's intent is to keep everyone out. The next container overrides for the appropriate subdirectory.

```
<Directory/>
  Order deny,allow
  deny from all
  allow from none
</Directory>

Alias /root /bobtest/xyz/html
<Directory /bobtest/xyz/html/>
  Order allow,deny
  allow from all
  AuthType Basic
  AuthName "root and %%SYSTEM%%"
  PasswdFile %%SYSTEM%%
  Require valid-user
  UserID %%SYSTEM%%
</Directory>
```

Hosts in the ibm.com domain are allowed access; all other hosts are denied access.

## Module mod\_actions

Module mod\_actions contains directives for the IBM HTTP Server for i Web server.

### Summary

The module mod\_actions provides for executing CGI scripts based on media type or request method.

### Directives

- "Action"
- "Script" on page 222

### Action:

**Module:** mod\_actions

**Syntax:** Action *action-type* *cgi-script*

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** FileInfo

**Origin:** Apache

**Example:** Action application/x-www-form-urlencoded /cgi-bin/file.pgm

The Action directive adds an action, which will activate CGI script when action-type is triggered by the request.

### Parameter One: *action-type*

- The action-type can be either a handler or a MIME content type. It sends the URL and file path of the requested document using the standard CGI PATH\_INFO and PATH\_TRANSLATED environment variables. See "Handler for HTTP Server" on page 200 for more information on handlers.

### Parameter Two: *CGI-script*

- The CGI-script can be any valid CGI script or other resource that is capable of handling the requested action-type.

## Script:

**Module:** mod\_actions

**Syntax:** Script *method CGI-script*

**Default:** none

**Context:** server config, virtual host, directory

**Override:** none

**Origin:** Apache

**Example:** Script PUT /cgi-bin/bob.pgm

The Script directive adds an action, which will activate *CGI-script* when a file is requested using the method of *method*. It sends the URL and file path of the requested document using the standard CGI PATH\_INFO and PATH\_TRANSLATED environment variables. Method names are case-sensitive, so Script *PUT* and Script *put* have two entirely different effects.

### Parameter One: *method*

- The *method* names listed can be one or more of the following: GET, POST, PUT, DELETE, CONNECT, OPTIONS, PATCH, PROPFIND, PROPPATCH, MKCOL, COPY, MOVE, LOCK and UNLOCK. User defined method names can also be used. The *method* name is case-sensitive. If GET is used it will also handle HEAD requests.

### Parameter Two: *CGI-script*

- The *CGI-script* can be any valid CGI script or other resource that is capable of handling the requested *method*.

**Note:** The *CGI-script* command defines default actions only. If a CGI script is called, or some other resource that is capable of handling the requested method internally, it will do so. Also note that CGI script with a *method* of GET will only be called if there are query arguments present (for example, bob.html?hi). Otherwise, the request will proceed normally.

## Module mod\_alias

Module mod\_alias contains directives for the IBM HTTP Server for i Web server.

### Summary

The module mod\_alias provides mapping for different parts of the host file system in the document tree and also for URL redirection.

### Directives

- “Alias”
- “AliasMatch” on page 223
- “MapMatch” on page 224
- “Redirect” on page 224
- “RedirectMatch” on page 225
- “RedirectPermanent” on page 226
- “RedirectTemp” on page 227
- “ScriptAlias” on page 227
- “ScriptAliasMatch” on page 228

### Alias:

**Module:** mod\_alias

**Syntax:** Alias *url-path directory-filename*

**Default:** none

**Context:** server config, virtual host  
**Override:** none  
**Origin:** Apache  
**Example:** Alias /image /QIBM/UserData/pub/image  
**Example:** Alias /httpfile/ /QSYS.LIB/AS400LIB.LIB/HTML.FILE/

This directive allows documents to be stored in the local filesystem other than under the “DocumentRoot” on page 293. URLs with a (%-decoded) path beginning with the value of the URL-path parameter will be mapped to local files beginning with the value of directory-filename. Alias also allows you to hide the file system path from users, enhancing both security of your server and the ability to change the filesystem structure or paths without impacting the end users.

**Parameter One:** *url-path*

- The *url-path* parameter is any valid URL path. If you include a trailing '/' in the URL path, then the server will require a trailing '/' in order to expand the alias. That is, if you use 'Alias /icons/ /www/images/i/icons/' then the URL '/icon' will not be aliased.

**Parameter Two:** *directory-filename*

- The *directory-filename* parameter is any valid directory/filename combination on the system.

**Note:** You may need to specify additional “<Directory>” on page 291 containers that cover the destination of aliases. Aliasing occurs before <Directory> containers are checked, so only the destination of aliases are affected. “<Location>” on page 312 containers are run through once before aliases are performed, so they will apply.

See “ScriptAlias” on page 227 for more information.

**AliasMatch:**

**Module:** mod\_alias  
**Syntax:** AliasMatch *regex directory-filename*  
**Default:** none  
**Context:** server config, virtual host  
**Override:** none  
**Origin:** Apache  
**Example:** AliasMatch ^/icons(\*) /www/images/HTTP\_Server/icons\$1  
**Example:** AliasMatch ^/lib/docs(\*) /QSYS.LIB/DOCLIB.LIB/HTMLDOC.FILE/\$1.MBR

This directive is equivalent to “Alias” on page 222, but makes use of standard regular expressions, instead of simple prefix matching. The supplied regular expression is matched against the URL, and if it matches, the server will substitute any parenthesized matches into the given string and use it as a filename.

**Parameter One:** *regex*

- The *regex* parameter is a regular expression that is matched against the URL. Subexpressions are grouped within parentheses. Then parenthetically enclosed regular expressions will be substituted in a subsequent \$n statement.

**Parameter Two:** *directory-filename*

- The *directory-filename* parameter is any valid directory/filename that is supported on the IBM i server. If there is a \$ symbol (followed by a digit) that is not a substitution variable in the *directory-filename* parameter, or there is an & symbol in the *directory-filename* parameter that is part of the directory or filename, the symbol must be escaped (\).

If the directory-filename is `/usr/local/apache/icons&gifs/` the `&` would need to be escaped as follows on the `AliasMatch` directive:

```
AliasMatch ^/icons(.*) /usr/local/apache/icons\gifs/
```

If the directory-filename is `/usr/local/apache/icon$1/` the `$` would need to be escaped as follows on the `AliasMatch` directive:

```
AliasMatch ^/icons(.*) /usr/local/apache/icon\$1/
```

See “Regular expression notation for HTTP Server” on page 491 for more information.

### MapMatch:

**Module:** `mod_alias`

**Syntax:** `MapMatch regex URI`

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Example:** `MapMatch ^/icons(.*) /www/apache/icons\&gifs/`

The `MapMatch` directive uses standard regular expressions to change a URI to a different URI. The supplied regular expression is matched against the URL, and if it matches, the server will substitute any parenthesized matches into the given string and use it as the URI. This is not a terminating directive. The server will use the new URI as input to `Alias`, `Redirect` or other `MapMatch` directives.

#### Parameter One: *regex*

- The *regex* parameter is a regular expression that is matched against the URL. Subexpressions are grouped within parentheses. The parenthetically enclosed regular expressions will be substituted in a subsequent `$n` statement.

#### Parameter Two: *URI*

- The *URI* parameter is any valid URI that is supported on the IBM i server. If there is a `$` symbol (followed by a digit) that is not a substitution variable in the URI parameter, or there is an `&` symbol in the URI parameter that is part of the URI, the symbol must be escaped (`\`).

If the target URI is `/www/apache/icons\&gifs/` the `&` would need to be escaped as follows on the `MapMatch` directive:

```
MapMatch ^/icons(.*) /www/apache/icons\&gifs/
```

If the target URI is `/www/apache/icon$1/` the `$` would need to be escaped as follows on the `MapMatch` directive:

```
MapMatch ^/icons(.*) /www/apache/icon\$1/
```

See “Regular expression notation for HTTP Server” on page 491 for more information.

### Redirect:

**Module:** `mod_alias`

**Syntax:** `Redirect [status] url-path url`

**Default:** none

**Context:** server config, virtual host, directory, `.htaccess`

**Override:** `FileInfo`

**Origin:** Apache

**Example:** `Redirect /service http://foo2.bar.com/service`

The Redirect directive maps an old URL into a new one. The new URL is returned to the client, who then attempts to access the page with the new address. URL-path is a (%-decoded) path; any requests for documents beginning with this path will be returned with a redirect error to a new (%-encoded) URL beginning with *url*.

**Parameter One:** *status*

- The *status* parameter is used to return the below HTTP status codes:

Status	Description
permanent	Returns a permanent redirect status (301) indicating that the resource has moved permanently.
temp	Returns a temporary redirect status (302). This is the default.
seeother	Returns a "See Other" status (303) indicating that the resource has been replaced.
gone	Returns a "Gone" status (410) indicating that the resource has been permanently removed. When this status is used the <i>url</i> argument should be omitted.

If no status argument is given, the redirect will be "temporary" (HTTP status 302). This indicates to the client that the resource has moved temporarily. Other status codes can be returned by giving the numeric status code as the value of status. If the status is between 300 and 399, the url argument must be present, otherwise it must be omitted. Regardless, any HTTP status given must be known to HTTP Server.

**Parameter Two:** *url-path*

- If the url-path has a trailing slash ('/'), the *url* should also have a trailing slash. If the *url-path* does not contain a trailing slash, the *url* should not either. Double check the designated *url-path* and the *url*, or a double-slash ('//') may appear in the resulting URL. The *url-path* must be an absolute path, not a relative path, even when used with .htaccess files or inside of "<Directory>" on page 291 containers. The *url-path* must match the requested resource exactly or be a proper ancestor of it.

**Parameter Three:** *url*

- The *url* parameter should be a complete URL string, including the scheme ('http://...') and the 'server:host' portion. When the status parameter is "gone", the url argument should be omitted.

**Note:** Redirect directives take precedence over Alias and ScriptAlias directives, regardless of their order in the configuration file.

**RedirectMatch:**

**Module:** mod\_alias

**Syntax:** RedirectMatch [*status*] *regex url*

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** FileInfo

**Origin:** Apache

**Example:** RedirectMatch (\*.\*)\.gif\$ http://www.anotherserver.com\$1.jpg

This directive is equivalent to "Redirect" on page 224, but makes use of standard regular expressions, instead of simple prefix matching. The supplied regular expression is matched against the URL, and if it matches, the server will substitute any parenthesized matches into the given string and use it as a filename.

**Parameter One:** *status*

- The *status* parameter is used to return the below HTTP status codes:

Status	Description
permanent	Returns a permanent redirect status (301) indicating that the resource has moved permanently.
temp	Returns a temporary redirect status (302). This is the default.
seeother	Returns a "See Other" status (303) indicating that the resource has been replaced.
gone	Returns a "Gone" status (410) indicating that the resource has been permanently removed. When this status is used the <i>url</i> argument should be omitted.

If no status argument is given, the redirect will be "temporary" (HTTP status 302). This indicates to the client that the resource has moved temporarily. Other status codes can be returned by giving the numeric status code as the value of status. If the status is between 300 and 399, the url argument must be present, otherwise it must be omitted. Regardless, any HTTP status given must be known to HTTP Server.

**Parameter Two:** *regex*

- The *regex* parameter is a regular expression that is matched against the URL. Subexpressions are grouped within parentheses. Then, parenthetically enclosed regular expressions will be substituted in a subsequent \$n statement.

**Parameter Three:** *url*

- The *url* parameter should be a complete URL string, including the scheme ('http://...') and the 'server:port' portion. If there is a \$ symbol (followed by a digit) that is not a substitution variable in the url parameter, or there is an & symbol in the url parameter that is part of the URL, the symbol must be escaped (\).

If the URL to redirect to is `http://www.anotherserver.com/cgi-bin/welcome.cgi?parm1=login&parm2=mainlist` the & would need to be escaped as follows on the RedirectMatch directive:  
`RedirectMatch (.*) http://www.anotherserver.com/cgi-bin/welcome.cgi?parm1=login&parm2=mainlist`

If the URL to redirect to is `http://www.anotherserver.com/htdocs/welcome$2login.html` the \$2 would need to be escaped as follows on the RedirectMatch directive:  
`RedirectMatch (.*) http://www.anotherserver.com/htdocs/welcome\$2login.html`

See "Regular expression notation for HTTP Server" on page 491 for more information.

**RedirectPermanent:**

**Module:** mod\_alias

**Syntax:** RedirectPermanent *url-path url*

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** FileInfo

**Origin:** Apache

**Example:** RedirectPermanent /payroll http://payroll.server.com/payroll

The RedirectPermanent directive notifies the client that the Redirect is permanent (status 301). This is the exact equivalent to Redirect permanent.

**Parameter One:** *url-path*



- The *url-path* parameter is any valid URL path. If you include a trailing '/' in the URL path, then the server will require a trailing '/' in order to expand the alias. That is, if you use 'Alias /icons/ /www/images/i/icons/' then the URL '/icon' will not be aliased.

**Parameter Two:** *url*

- The *url* parameter should be a complete URL string, including the scheme ('http://...') and the 'server:host' portion. When the status parameter is "gone", the *url* argument should be omitted.

See “Regular expression notation for HTTP Server” on page 491 for more information.

**RedirectTemp:**

**Module:** mod\_alias

**Syntax:** RedirectTemp *url-path url*

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** FileInfo

**Origin:** Apache

**Example:** RedirectTemp /service http://foo2.bar.com/service

The RedirectTemp directive notifies the client that the Redirect is only temporary (status 302). This is the exact equivalent to Redirect temp.

**Parameter One:** *url-path*

- The *url-path* parameter is any valid URL path. If you include a trailing '/' in the URL path, then the server will require a trailing '/' in order to expand the alias. That is, if you use 'Alias /icons/ /www/images/i/icons/' then the URL '/icon' will not be aliased.

**Parameter Two:** *url*

- The *url* parameter should be a complete URL string, including the scheme ('http://...') and the 'server:host' portion. When the status parameter is "gone", the *url* argument should be omitted.

See “Regular expression notation for HTTP Server” on page 491 for more information.

**ScriptAlias:**

**Module:** mod\_alias

**Syntax:** ScriptAlias *url-path directory-filename*

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Example:** ScriptAlias /cgi-bin/ /web/cgi-bin/

**Example:** ScriptAlias /cgi-bin/ /QSYS.LIB/QSYSCGI.LIB/

The ScriptAlias directive has the same behavior as the “Alias” on page 222 directive, except that in addition it marks the target directory as containing CGI scripts, and then executes the CGI program. URLs with a (%-decoded) path beginning with *url-path* will be mapped to scripts beginning with *directory-filename*. Additional “<Directory>” on page 291 containers that cover the destination of the ScriptAlias may need to be specified. Aliasing occurs before <Directory> containers are checked, so only the destination of Aliases are affected.

**Parameter One:** *url-path*

- The *url-path* parameter is any valid url-path. It must end with a slash ( '/') character so that any files in the directory will be routed.

**Parameter Two:** *directory-filename*

- The *directory-filename* parameter is any valid directory/filename on the IBM i server.

**Note:** If the URL ends in a slash ("/") character, the ScriptAlias must also end in a slash character.

**ScriptAliasMatch:**

**Module:** mod\_alias

**Syntax:** ScriptAliasMatch *regex directory-filename*

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Example:** ScriptAliasMatch `^/cgi-bin/(.*)\.cgi /QSYS.LIB/QSYSCGI.LIB/$1.PGM`

This directive is equivalent to “ScriptAlias” on page 227, but makes use of standard regular expressions, instead of simple prefix matching. The supplied regular expression is matched against the URL, and if it matches, the server will substitute any parenthesized matches into the given string and use it as a filename.

**Parameter One:** *regex*

- The *regex* parameter is a regular expression that is matched against the URL. Subexpressions are grouped within parentheses. Then, parenthetically enclosed regular expressions will be substituted in a subsequent \$n statement.

**Parameter Two:** *directory-filename*

- This is any valid directory/filename that is supported on the IBM i server. If there is a \$ symbol (followed by a digit) that is not a substitution variable in the directory-filename parameter, or there is a & symbol in the directory-filename parameter that is part of the directory or filename, the symbol must be escaped (\).

If the directory-filename is `/usr/local/apache/cgi-bin&sym/$1.pgm`, where the \$1 is a substitution variable, the & would need to be escaped as follows on the ScriptAliasMatch directive:

```
ScriptAliasMatch ^/cgi-bin/(.*)\.cgi /usr/local/apache/cgi-bins&sym/$1.pgm
```

If the directory-filename is `/usr/local/apache/cgi-bin$2sym/ $1.pgm`, where the \$1 is a substitution variable, the \$2 would need to be escaped as follows on the ScriptAliasMatch directive:

```
ScriptAliasMatch ^/cgi-bin/(.*)\.cgi /usr/local/apache/cgi-bin\$2sym/$1.pgm
```

## Module ap\_charset

Module mod\_ap\_charset contains directives for the IBM HTTP Server for i Web server.

### Summary

The module ap\_charset provides support for performing ASCII to EBCDIC and EBCDIC to ASCII codepage conversions.

### Directives

- “DefaultFsCCSID”
- “UseJCD” on page 229

### DefaultFsCCSID:

**Module:** ap\_charset

**Syntax:** DefaultFsCCSID *server-character-set-identification-number*

**Default:** dependent on server settings  
**Context:** server config  
**Override:** none  
**Origin:** IBM  
**Example:** DefaultFsCCSID 37

The DefaultFsCCSID directive specifies the CCSID that your server runs under, the server character set environment, and the EBCDIC CCSID that is used when the server converts:

- Input request data for user CGI programs or Apache modules.
- Output response data from user CGI programs, or Apache modules, to be sent back to the requester (client browser).

A configuration file can contain more than one DefaultFsCCSID directive, but the last directive in the configuration file determines the CCSID.

If the HTTP Server startup value *-fscsid* is specified on the STRTCPSVR command or as a parameter on the HTTP Administration's start server , the value specified overrides all other settings and is used for the server CCSID.

If there is no startup value specified, but there is a DefaultFsCCSID directive in the configuration file, the directive value will be used for the server CCSID.

If there is no startup value specified and there is no DefaultFsCCSID directive in the configuration file, then the QCCSID system value is used. If the QCCSID system value is set to 65535, then the server job will be started with that CCSID. However, the CCSID that the server actually uses for conversions will be the job default *ccsid* which is set to an appropriate value based on the language (LANGID) of the server job.

To display the CCSID of the server, complete the following task:

1. Start a 5250 session on your IBM i server.
2. Type WRKACTJOB (Work Active Job).
3. Type a 5 (Work with...) next to your server job.
4. Type a 2 (Display job definition attributes) on the **Work with Job** screen.
5. Page down until you see the job CCSID fields.

**Example**

In this case, the QCCSID system value was used to start the server job. We see that the Coded character set identifier is 65535. However, the Default coded character set identifier has been set to 37 because the Language identifier is ENU (United States English). The server will use CCSID 37 as the EBCDIC CCSID.

```
Language identifier . . . . . : ENU
Country or region identifier . . . . . : US
Coded character set identifier . . . . . : 65535
Default coded character set identifier . . . . . : 37
```

**UseJCD:**

**Module:** ap\_charset  
**Syntax:** UseJCD *On* | *Off*  
**Default:** UseJCD Off  
**Context:** server config  
**Override:** none  
**Origin:** IBM  
**Example:** UseJCD Off

This directive is used to instruct the server to perform Japanese codepage detection on the request body.

Japanese browsers can potentially send data in one of three code pages, JIS (ISO-2022-JP), S-JIS (PC-Windows), or EUC (UNIX). If this directive is set to *On*, the server uses a well-known JCD utility to determine which codepage to use (if not explicitly specified by a charset tag) to convert the request body.

**Parameter:** *On* | *Off*

- When *On* is specified, the server uses a well-known JCD utility to determine which codepage to use (if not explicitly specified by a charset tag) to convert the request body.
- When *Off* is specified, Japanese codepage detection on the request body is disabled.

This directive is intended for module writers that need the server to detect JCD on the request body. CGI writers can use the CGIConvMode value "EBCDIC\_JCD" to instruct the server to perform JCD.

## Module mod\_arm4\_ap20

Module mod\_arm4\_ap20 contains directives for the IBM HTTP Server for i Web server.

### Summary

The module mod\_arm4\_ap20 uses the ARM (Application Response Measurement) 4.0 APIs to classify requests and record the time spent for each one. Configuring these directives enables ARM services for the IBM HTTP Server for i.

To enable ARM on IBM HTTP Server for i, perform these steps:

1. Ensure that the EWLM managed server is configured and started, and that it is communicating properly with its EWLM domain manager.
2. Ensure that IBM HTTP Server for i is installed and configured.
3. Ensure you have the latest required PTFs installed for EWLM to monitor the HTTP Server for IBM i application.
4. Add the following directives to the configuration file:

```
LoadModule arm4_module /QSYS.LIB/QHTTPSVR.LIB/QZSRARM.SRVPGM
ArmLoadLibrary /QSYS.LIB/QSYS2.LIB/LIBARM4.SRVPGM
```

To edit the configuration file, follow these steps:

- a. Start the IBM Web Administration for i interface.
- b. Click the **Manage** tab.
- c. Click the **HTTP Servers** subtab.
- d. Select your HTTP Server from the **Server** list.
- e. Select **System Resources**.
- f. Change **Activate Application Response Measurement (ARM) instrumentation** to **Enabled**.
- g. Click **OK** when you finish editing the configuration file.
- h. Stop and restart the HTTP Server.

### Directives

- "ArmApplicationName" on page 231
- "ArmInstrumentHandler" on page 231
- "ArmLoadLibrary" on page 231
- "ArmTransactionName" on page 231

### ArmApplicationName:

**Module:** mod\_arm4\_ap20

**Syntax:** ArmApplicationName *application\_name*

**Default:** ArmApplicationName "IBM Webserving Plugin"

**Context:** server

**Override:** none

**Origin:** IBM

**Usage:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule arm4\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRARM.SRVPGM

**Example:** ArmApplicationName "IBM Webserving Plugin"

IBM HTTP Server for i is an ARM-instrumented application. ARM 4.0 enables the real time measurement of transactions, transaction components, and underlying resource usage associated with the execution of an application. Use this directive to set specific information passed by the ARM API function calls which will be used in the filtering criteria that EWLM uses for transaction classification.

### ArmInstrumentHandler:

**Module:** mod\_arm4\_ap20

**Syntax:** ArmInstrumentHandler *on|off*

**Default:** ArmInstrumentHandler off

**Context:** server

**Override:** none

**Origin:** IBM

**Usage:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule arm4\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRARM.SRVPGM

**Example:** ArmInstrumentHandler off

When the ArmInstrumentHandler directive is turned on, arm\_block|unblock\_transaction is called across content handlers to notify the IBM ARM implementation that a blocking condition is finished.

### ArmLoadLibrary:

**Module:** mod\_arm4\_ap20

**Syntax:** ArmLoadLibrary *arm4-api-service-program-name*

**Default:** ArmLoadLibrary /QSYS.LIB/QSYS2.LIB/LIBARM4.SRVPGM

**Context:** server

**Override:** none

**Origin:** IBM

**Usage:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule arm4\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRARM.SRVPGM

**Example:** ArmLoadLibrary /QSYS.LIB/QSYS2.LIB/LIBARM4.SRVPGM

This directive is needed to activate the EWLM (Enterprise Workload Management) instrumentation module for HTTP Server. It uses the ARM (Application Response Measurement) 4.0 APIs to classify requests and record the time spent for each one.

### ArmTransactionName:

**Module:** mod\_arm4\_ap20

**Syntax:** ArmTransactionName *transaction\_name*

**Default:** ArmTransactionName WebRequest

**Context:** server

**Override:** none

**Origin:** IBM

**Usage:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule arm4\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRARM.SRVPGM

**Example:** ArmTransactionName WebRequest

IBM HTTP Server for i is an ARM-instrumented application. ARM 4.0 enables the real time measurement of transactions, transaction components, and underlying resource usage associated with the execution of an application. Use this directive to set specific information passed by the ARM API function calls, which are used in the filtering criteria that EWLM uses for transaction classification.

## Module mod\_as\_auth

Module mod\_as\_auth contains directives for the IBM HTTP Server for i Web server.

### Summary

The module mod\_as\_auth provides user authentication using IBM i system profiles, Internet users (through validation lists), or LDAP users.

### Directives

- “AsAuthAuthoritative”
- “GroupFile”
- “PasswdFile” on page 233
- “UserID” on page 234

#### AsAuthAuthoritative:

**Module:** mod\_as\_auth

**Syntax:** AsAuthAuthoritative *On* | *Off*

**Default:** AsAuthAuthoritative On

**Context:** directory

**Override:** none

**Origin:** IBM

**Example:** AsAuthAuthoritative Off

Setting the AsAuthAuthoritative directive explicitly to off allows for both authentication and authorization to be passed on to lower level modules (if there is no userid or rule matching the supplied userid).

**Parameter:** *On* | *Off*

- When *On* is specified, both authentication and authorization are not allowed to be passed on to lower level modules (if there is no userid or rule matching the supplied userid).
- When *Off* is specified, allows for both authentication and authorization to be passed on to lower level modules (if there is no userid or rule matching the supplied userid).

If a userid appears in an authentication realm other than those supported by IBM i (for example, System Userid), or if a valid Require directive applies to more than one module, the first module verifies the credentials and no access is passed on regardless of the AsAuthAuthoritative setting.

#### GroupFile:

**Module:** mod\_as\_auth

**Syntax:** GroupFile *filename*

**Default:** none

**Context:** directory

**Override:** none  
**Origin:** IBM  
**Example:** GroupFile /docs/restrict.group

The GroupFile directive sets the name of a GroupFile to use for a protection setup. Group files are used to classify users into various groups. A protection setup can use groups on limit directives. If a protected directory contains an ACL file, the rules in the ACL file can also use the groups that you define in the group file.

**Parameter:** *filename*

- The *filename* parameter is any valid filename.

**Note:** The GroupFile directive is case-sensitive. If the filename is incorrectly cased, the GroupFile directive will not work properly. Since IBM i user profiles are not case-sensitive, the entries in the GroupFile will be treated as non-case-sensitive if the PasswdFile directive is set to %%SYSTEM%%. For all other values of PasswdFile, the values in the GroupFile will be treated as case-sensitive.

To work correctly this directive must be accompanied by "PasswdFile," "AuthType" on page 290, and Require.

#### **PasswdFile:**

**Module:** mod\_as\_auth  
**Syntax:** PasswdFile *passfile* [*passfile* *passfile* ...]  
**Default:** none  
**Context:** directory  
**Override:** none  
**Origin:** IBM  
**Example:** PasswdFile %%SYSTEM%%  
**Example:** PasswdFile "QUSRSYS/MY\_USERS QGPL/DOC\_USERS"

The PasswdFile directive specifies where the passwords (or certificates) are stored for authentication.

**Parameter:** *passfile*

The different values supported by the passfile parameter value are:

%%SYSTEM%%

The passfile parameter can be in the %%SYSTEM%% format. Using this value indicates that the server should use the IBM i User Profile support to validate username/password.

%%LDAP%%

The passfile can also be in the %%LDAP%% format to validate the LDAP server that has been defined to the server.

%%KERBEROS%%

The passfile parameter should be set to %%KERBEROS%% when the directive AuthType Kerberos is configured.

**passfile** [*passfile* *passfile* ...]

The passfile parameter can be formatted to fit the Internet user list. To use this format, specify QUSRSYS/MY\_USERS as the filename. The HTTP Server allows a space separated list of Internet User lists (for example: 'library/vldl library/fort').

This directive may be configured multiple times in a container. The directives are processed from the first to the last occurrence.

To work correctly this directive must be accompanied by AuthType, AuthName, and Require.

#### UserID:

**Module:** mod\_as\_auth

**Syntax:** Userid *user-profile* | *%%SERVER%%* | *%%CLIENT%%*

**Default:** none

**Context:** directory

**Override:** none

**Origin:** IBM

**Example:** UserID WEBUSER

**Example:** UserID *%%SERVER%%*

**Example:** UserID *%%CLIENT%%*

The UserID directive specifies the IBM i system profile to the server. For a protected resource (one for which Protection directives are defined), the UserID directive specifies which IBM i system profile the server temporarily swaps to while serving that resource. The directive must be a valid user profile.

**Parameter:** *user-profile* | *%%SERVER%%* | *%%CLIENT%%*

- For *user-profile*, a valid IBM i system profile must be specified. The value 'QSECOFR' cannot be specified on the directive. The profile that issued the STRTCPSVR command to start HTTP Server must have \*USE authority to the profile specified on all of the UserID directives and other directives. All UserID directives (and directives specified for a protected resource) are verified during startup. If any UserID directive, or any other directive, does not satisfy the rules listed here, the server instance does not start and a message is sent to the user's interactive job log.
- Entering *%%SERVER%%* uses the default profile QTMHHTTP unless the ServerUserId directive is specified.
- Entering *%%CLIENT%%* causes the user profile from the request to be used on the swap. If Kerberos is specified for the AuthType directive, the server will use Enterprise Identity Mapping (EIM) to attempt to match the user ID associated with the server ticket with an IBM i system profile. If there is no IBM i system profile associated with the server ticket user ID, the HTTP request will fail. This value cannot be used for LDAP or Validation lists authentication. It is valid for IBM i profiles, client certificates, and Kerberos.

The profile that issued the STRTCPSVR command to start HTTP Server must have \*USE authority to the profile specified on all of the UserID directives and other directives. All UserID directives (and directives specified for a protected resource) are verified during startup. If any UserID directive, or any other directive, does not satisfy the rules, the server instance does not start and a message is sent to the user's interactive joblog.

**Note:** Because HTTP Server swaps to the profile that you specify on the UserID directive, you should be careful what profile you specify. For example, if you create a profile MIGHTY1 that is of the class \*SECOFR and use this profile on the UserID directive, then whenever the server invokes a swap to that profile, all IBM i authority checking for the requested resource is based on that profile.

When HTTP Server is running under the QTMHHTTP profile (the QTMHHTTP profile is the default) and a UserID directive is not in effect, the server switches to the QTMHHTTP1 profile before starting a CGI program. However, when a CGI program is running on servers where the UserID directive is in effect or within a protection setup where the UserID directive has been specified, the program is run under the specified profile, unless the profile is QTMHHTTP. In which case, QTMHHTTP1 is used. If the profile does not have authority to the specified program, the request is rejected.



There are two special values you can use on the UserID directive. Entering `%%SERVER%%` uses the default profile QTMHHTTP unless a protection setup has a different UserID specified. Entering `%%CLIENT%%` causes the server to challenge the client on each and every request for a user ID and password.

See also ServerUserID.

To work correctly, this directive must be accompanied by the PasswdFile, AuthType, AuthName, and Require directives.

## Module mod\_as\_cache

Module mod\_as\_cache contains directives for the IBM HTTP Server for i Web server.

### Summary

The module mod\_as\_cache provides support for caching frequently referenced files. It can be used to cache file content, file descriptors or both, or mmap the file.

### Directives

- "CacheLocalFD" on page 236
- "CacheLocalFile" on page 236
- "CacheLocalFileMmap" on page 237
- "CacheLocalFilePublic" on page 237
- "CacheLocalFileSizeLimit" on page 238
- "CacheLocalSizeLimit" on page 238
- "DynamicCache" on page 238
- "FRCACacheLocalFileRunTime" on page 239
- "FRCACacheLocalFileSizeLimit" on page 240
- "FRCACacheLocalFileStartUp" on page 240
- "FRCACacheLocalSizeLimit" on page 241
- "FRCACookieAware" on page 241
- "FRCAEnableFileCache" on page 242
- "FRCAEnableProxy" on page 242
- "FRCAEndofURLMarker" on page 243
- "FRCAMaxCommBufferSize" on page 243
- "FRCAMaxCommTime" on page 243
- "FRCAProxyCacheEntitySizeLimit" on page 244
- "FRCAProxyCacheExpiryLimit" on page 244
- "FRCAProxyCacheRefreshInterval" on page 244
- "FRCAProxyCacheSizeLimit" on page 245
- "FRCAProxyPass" on page 245
- "FRCARandomizeResponse" on page 246
- "LiveLocalCache" on page 246
- "PublicCache" on page 247

### CacheLocalFD:

**Module:** mod\_as\_cache

**Syntax:** CacheLocalFD *filename*

**Default:** none

**Context:** server config

**Override:** none

**Origin:** IBM

**Example:** CacheLocalFD some\_image.gif

The CacheLocalFD directive is used to specify the names of ASCII/BINARY stream files whose descriptors you want to cache at server startup. The file is opened (share read) and remains open while the server is active. The configuration file can contain multiple directive occurrences. Include a separate directive for each file that you want to remain open. By keeping your most frequently requested files/images opened at server startup, you can improve your server's response time for those files. For example, if you open your server's welcome page files at startup, the server can handle requests for the page much more quickly than if it had to open the files each time they are requested.

**Parameter:** *filename*

- The *filename* parameter specifies the names of ASCII/BINARY stream files whose descriptors are cached at server startup.

The advantage of using CacheLocalFD directive over CacheLocalFile is that it does not cache the content of the file, and therefore does not allocate a large amount of memory, yet provides similar performance. The disadvantage of using CacheLocalFD directive over CacheLocalFile is that it only caches the file descriptors of ASCII/BINARY stream files and it keeps the file open (share read) while the server is active.

The LiveLocalCache directive setting does not apply to this directive and if a cached file is updated, the cached entity is discarded and the updated file is served from the file system. If a cached file is modified while at the same time being served, the content of the response body is unpredictable.

**Note:** You can use an asterisk (\*) as a wildcard character on the file names (for example, CacheLocalFD \*.gif). File name matching is not recursive through subdirectories. The server will only cache files in the specified directory. No files in subdirectories are affected.

### CacheLocalFile:

**Module:** mod\_as\_cache

**Syntax:** CacheLocalFile *filename*

**Default:** none

**Context:** server config

**Override:** none

**Origin:** IBM

**Example:** CacheLocalFile bobwelcome.html

The CacheLocalFile directive is used to specify the names of files that you want to load into the server's memory each time that you start the server, and is the recommended file cache method. You can have multiple occurrences of this directive in the configuration file. Include a separate directive for each file that you want to load into memory. By keeping your most frequently requested files loaded in the server's memory, you can improve your server's response time for those files. For example, if you load your server's welcome page into memory at startup, the server can handle requests for the page much more quickly than if it had to read the file from the file system.

**Parameter:** *filename*

- The *filename* parameter specifies the names of files that you want to load into the server's memory each time that you start the server.

**Note:** You can use an asterisk (\*) as a wildcard character on the file names (for example, CacheLocalFile \*.html). File name matching is not recursive through subdirectories. The server will only cache files in the specified directory. No files in subdirectories are affected.

### CacheLocalFileMmap:

**Module:** mod\_as\_cache

**Syntax:** CacheLocalFileMmap *filename*

**Default:** none

**Context:** server config

**Override:** none

**Origin:** IBM

**Example:** CacheLocalFileMmap bobwelcome.html

The CacheLocalFileMmap directive is used to specify the names of files that you want to map to the server's memory each time that you start the server. This directive is similar to the CacheLocalFile directive. Whereas CacheLocalFile allocates storage and copies (read/write) the content of the file to the allocated storage, CacheLocalFileMmap maps the file content to the process storage space without actually allocating storage.

The LiveLocalCache directive setting does not apply to this directive and if a cached file is updated, the cached entity is discarded and the updated file is served from the file system. If a cached file is modified while at the same time being served, the content of the response body is unpredictable.

**Parameter:** *filename*

- The *filename* parameter specifies the names of files that you want to map to the server's memory each time that you start the server.

You can have multiple occurrences of this directive in the configuration file. Include a separate directive for each file that you want to load into memory. By keeping your most frequently requested files mapped in the server's address space, you can improve your server's response time for those files. For example, if you map your server's welcome at startup, the server can handle requests for the page much more quickly than if it had to read the file from the file system.

**Note:** You can use an asterisk (\*) as a wildcard character on the file names (for example, CacheLocalFileMmap \*.html). File name matching is not recursive through subdirectories. The server will only cache files in the specified directory. No files in subdirectories are affected. The relative/absolute path rules apply to this directive, meaning that a path that begins without a leading (/) character is considered to be absolute. Otherwise, the path is based on the server's document root.

### CacheLocalFilePublic:

**Module:** mod\_as\_cache

**Syntax:** CacheLocalFilePublic *filename*

**Default:** none

**Context:** server

**Override:** none

**Origin:** IBM

**Example:** CacheLocalFilePublic bobwelcome.html

The CacheLocalFilePublic directive is used to specify the names of files that you want to load into the server's memory each time that you start the server. The files cached here are files that are served without any server authentication. This directive is used by SSL sites which have pages that are publicly available. This simulates the FRCA function completed by the server for non-SSL publicly available files. You can

have multiple occurrences of this directive in the configuration file. Include a separate directive for each file that you want to load into memory. By keeping your most frequently requested public files loaded in the server's memory, you can improve your server's response time for those files. For example, if you load your server's welcome page into memory at startup, the server can handle requests for the page much more quickly than if it had to read the file from the file system.

**Note:** You can use an asterisk ("\*") as a wildcard character on the file names, (for example, `CacheLocalFile *.html`).

File name matching is not recursive through subdirectories. The server only caches files in the specified directory. No files in subdirectories are affected. There is no authentication or authorization done before any files in this cache are served.

### **CacheLocalFileSizeLimit:**

**Module:** `mod_as_cache`

**Syntax:** `CacheLocalFileSizeLimit size`

**Default:** `CacheLocalFileSizeLimit 90000`

**Context:** server config

**Override:** none

**Origin:** IBM

**Example:** `CacheLocalFileSizeLimit 5000000`

The `CacheLocalFileSizeLimit` directive is used to specify, in bytes, the largest file that will be placed in the local memory cache. A file larger than the value specified for `CacheLocalFileSizeLimit` will not be placed in the cache. This prevents the cache from being filled by only a small number of very large files. The upper limit for this directive is capped at 16,000,000. If you specify a larger value the value 16,000,000 will be used.

### **CacheLocalSizeLimit:**

**Module:** `mod_as_cache`

**Syntax:** `CacheLocalSizeLimit size`

**Default:** `CacheLocalSizeLimit 2000`

**Context:** server config

**Override:** none

**Origin:** IBM

**Example:** `CacheLocalSizeLimit 25000`

The `CacheLocalSizeLimit` directive is used to specify the maximum amount of memory, in kilobytes, that you want to allow for file caching. You must specify the files that you want cached with the `CacheLocalFile` directive or by setting `DynamicCache` to `on`. The number you specify is the upper limit (the maximum upper limits 93 gigabytes (100,000,000,000 bytes)); the storage is allocated as a file is cached.

**Parameter:** *size*

- The *size* parameter specifies the maximum amount of memory, in kilobytes, that you want to allow for file caching.

**Note:** `CacheLocalSizeLimit` can help limit your cache size when you are using the wildcard character to specify the files on the `CacheLocalFile` directive.

### **DynamicCache:**

**Module:** `mod_as_cache`

**Syntax:** `DynamicCache on | off`

**Default:** DynamicCache off  
**Context:** server config  
**Override:** none  
**Origin:** IBM  
**Example:** DynamicCache on

The DynamicCache directive is used to specify if you want the server to dynamically cache frequently accessed files. Setting the dynamic cache directive to "on" instructs the server to cache the most frequently accessed files. This results in better performance and system throughput.

**Parameter:** *on* | *off*

- If the parameter is set to *on* the server will dynamically cache frequently accessed files.
- If the parameter is set to *off* the server will not dynamically cache frequently accessed files.

**Note:** Note requires links. If you know the files that are frequently accessed or you have a large number of files, then it is better to use CacheLocalFile, CacheLocalFD, or CacheLocalFileMmap to cache them at server startup.

### **FRCACacheLocalFileRunTime:**

**Module:** mod\_as\_cache  
**Syntax:** FRCACacheLocalFileRunTime *filename*  
**Default:** none  
**Context:** server config  
**Override:** none  
**Origin:** IBM  
**Example:** FRCACacheLocalFileRunTime /www/html/index.html

The FRCACacheLocalFileRunTime directive specifies the name of a file that you want to load into the SLIC NFC during server run time if and when it is requested by a client. The configuration file can contain multiple directive occurrences.

**Parameter:** *filename*

- The *filename* parameter value specifies the name of a file that you want to load into the SLIC NFC during server run time if and when it is requested by a client.

During server run time, the below example caches the specified file in FRCA NFC if it is requested by a client.

```
FRCACacheLocalFileRunTime /www/html/index.html
```

**Note:** You can use an asterisk (\*) as a wild card character on the file name. Filename matching is not recursive through subdirectories. The server only caches files in the specified directory. No files in other sub directories are affected.

During server run time, the below example caches in the FRCA NFC any .gif file in the /www/images directory that is requested by a client. For example,

```
FRCACacheLocalFileRunTime /www/images/*.gif
```

**Note:** You can use an asterisk (\*) as a wild card character on the file name. Filename matching is not recursive through subdirectories. The server will only cache files in the specified directory and its subdirectories.

During server run time, the below example will dynamically cache in the SLIC NFC (based on the file usage) any file that is in a directory path that starts with /www/imag and its subdirectories. For example,

```
FRCACacheLocalFileRunTime /www/imag*
```

**Note:** If directory name begins with / it is absolute, otherwise it is relative to the server's document root.

During server run time, the below example will dynamically cache in the SLIC NFC (based on the file usage) any file in any directory.

```
FRCACacheLocalFileRunTime /*
```

**Note:** If a directory name begins with / it is absolute, otherwise it is relative to the server's document root.

For caching files at the server run time, only specify the path name of the files that are intended for public viewing. That is, do not specify or configure file names containing sensitive information which is not intended for general users. FRCACacheLocalFileRunTime only caches files that do not require conversion. (IFS binary or ASCII files).

### **FRCACacheLocalFileSizeLimit:**

**Module:** mod\_as\_cache

**Syntax:** FRCACacheLocalFileSizeLimit *size*

**Default:** FRCACacheLocalFileSizeLimit 92160

**Context:** server config

**Override:** none

**Origin:** IBM

**Example:** FRCACacheLocalFileSizeLimit 32000

The FRCACacheLocalFileSizeLimit directive specifies the maximum file size (in bytes) that you want to allow for file caching. The directive can control cache storage for a number of smaller files when using wild card characters to specify files in the FRCACacheLocalFileStartUp and FRCACacheLocalFileDynamic directives

**Parameter:** *size*

- The *size* parameter value specifies the maximum file size (in bytes) that you want to allow for file caching.

The below example allows only caching of files that are equal to or less than 32000 bytes. Files greater than 32000 bytes are not cached.

```
FRCACacheLocalFileSizeLimit 32000
```

### **FRCACacheLocalFileStartUp:**

**Module:** mod\_as\_cache

**Syntax:** FRCACacheLocalFileStartUp *filename*

**Default:** none

**Context:** server config

**Override:** none

**Origin:** IBM

**Example:** FRCACacheLocalFileStartUp /www/html/index.html

The FRCACacheLocalFileStartUp directive specifies the file name that you want to load into the SLIC NFC each time you start the server. The configuration file can contain multiple directive occurrences.

**Parameter:** *filename*

- The *filename* parameter value specifies the file name that you want to load into the SLIC NFC each time you start the server.

The below example caches a specific file.

```
FRCACacheLocalFileStartUp /www/html/index.html
```

**Note:** You can use an asterisk (\*) as a wild card character on the file name. Filename matching is not recursive through subdirectories. The server only caches files in the specified directory. No files in other sub directories are affected.

The below example caches all .gif files in the /www/images directory.

```
FRCACacheLocalFileStartUp /www/images/*.gif
```

**Note:** If a directory name begins with / it is absolute, otherwise it is relative to the server's document root.

FRCACacheLocalFileStartUp only caches files that do not require conversion. (IFS binary or ASCII files). FRCA Proxy does not perform authentication or authorization checking. Therefore, do not specify or configure file names containing sensitive information that is not intended for public viewing.

### **FRCACacheLocalSizeLimit:**

**Module:** mod\_as\_cache

**Syntax:** FRCACacheLocalSizeLimit *size*

**Default:** FRCACacheLocalSizeLimit 2000

**Context:** server config

**Override:** none

**Origin:** IBM

**Example:** FRCACacheLocalSizeLimit 5000

The FRCACacheLocalSizeLimit directive specifies the maximum amount of storage (in kilobytes) that you want to allow for FRCA file caching.

**Parameter:** *size*

- The *size* parameter value specifies the maximum amount of storage (in kilobytes) that you want to allow for FRCA file caching.

The below example caches files until the accumulated size reaches 5000 kilobytes.

```
FRCACacheLocalSizeLimit 5000
```

**Note:** The specified value is the upper limit, the actual amount of allocated storage is the accumulated size of the cached files. This directive can limit the cache size when using wild card character to specify the files in the FRCACacheLocalFileStartUp directive.

If the specified directive size is greater than the amount of available storage in the FRCA network file cache, the FRCA network file only caches as many files as it has space for.

### **FRCACookieAware:**

**Module:** mod\_as\_cache

**Syntax:** FRCACookieAware *<path>*

**Default:** none

**Context:** server config

**Override:** none

**Origin:** IBM

**Example:** FRCACookieAware /some\_path\_segment

This FRCACookieAware directive indicates URL prefix for which the cookie should be included in cache lookup. This directive makes it possible to serve a cached entity only for the requests with the same cookie

**Parameter:** <path>

- The <path> parameter value specifies a valid path name.

### FRCAEnableFileCache:

**Module:** mod\_as\_cache

**Syntax:** FRCAEnableFileCache *on* | *off*

**Default:** FRCAEnableFileCache *off*

**Context:** server config

**Override:** none

**Origin:** IBM

**Example:** FRCAEnableFileCache *on*

The FRCAEnableFileCache directive enables or disables FRCA file caching support for the specified server.

**Parameter:** *on* | *off*

- If the parameter value is *on*, FRCA file caching support is enabled for the specified server.
- If the parameter value is *off*, all other FRCA file cache related directives in the configuration file are ignored.

**Note:** FRCA does not perform authentication or authorization checking for the HTTP requests that are served from the FRCA cache.

### FRCAEnableProxy:

**Module:** mod\_as\_cache

**Syntax:** FRCAEnableProxy *on* | *off*

**Default:** FRCAEnableProxy *off*

**Context:** server config, virtual host

**Override:** none

**Origin:** IBM

**Example:** FRCAEnableProxy *on*

The FRCAEnableFileCache directive enables or disables FRCA proxy support.

**Parameter:** *on* | *off*

- If the parameter value is *on*, FRCA proxy support is enabled for the specified container.
- If the parameter value is *off*, only FRCA directives in the server configuration section are ignored. If FRCAEnableProxy is set to *off* in a virtual host container, only FRCA directives in that virtual host container are ignored.

FRCA proxy does not perform authentication or authorization checking for the HTTP requests that are served by the FRCA Proxy support.

**Note:** Virtual host containers do not inherit the FRCAEnableProxy setting from the server configuration.



### FRCAEndofURLMarker:

**Module:** mod\_as\_cache  
**Syntax:** FRCAEndofURLMarker #  
**Default:** none  
**Context:** server config  
**Override:** none  
**Origin:** IBM  
**Example:** FRCAEndofURLMarker #

The FRCAEndofURLMarker directive specifies the unique string that identifies the end of URLs. Suppose a link in an html page is `http://some.org/some_path/some_parms#`. Before the client sends this request to the server, it may pad the URL with data such as `client_padded_data`. The `some.org` server will receive the path `/some_path/some_parms#client_padded_data`.

By specifying FRCAEndofURLMarker #, FRCA support can identify the end of the original URL (link) before it was modified or padded by the client.

### FRCAMaxCommBufferSize:

**Module:** mod\_as\_cache  
**Syntax:** FRCAMaxCommBufferSize *size*  
**Default:** FRCAMaxCommBufferSize 8000  
**Context:** server config  
**Override:** none  
**Origin:** Apache  
**Example:** FRCAMaxCommBufferSize 4000000

The FRCAMaxCommBufferSize directive sets the communication buffer size (in bytes) in FRCA for performance. The data being sent to HTTP Server consists of log data, message data, and collection services data. FRCA will buffer the size of data specified until the buffer is full. Once the buffer is full, the data will be transmitted to Apache for processing.

**Parameter:** *size*

- The *size* parameter value sets the communication buffer size (in bytes) in FRCA for performance.

### FRCAMaxCommTime:

**Module:** mod\_as\_cache  
**Syntax:** FRCAMaxCommTime *time*  
**Default:** FRCAMaxCommTime 120  
**Context:** server config  
**Override:** none  
**Origin:** Apache  
**Example:** FRCAMaxCommTime 240

The FRCAMaxCommTime directive sets the maximum number of seconds to wait before the data buffer is sent from FRCA to HTTP Server. The data being sent to HTTP Server consists of log data, message data, and collection services data. Once the time limit has been reached, the data will be transmitted to HTTP Server for processing. Valid values include integers 0 through 65,535.

**Parameter:** *time*

- The *time* parameter value sets the maximum number of seconds to wait before the data buffer is sent from FRCA to HTTP Server.

### **FRCAProxyCacheEntitySizeLimit:**

**Module:** mod\_as\_cache

**Syntax:** FRCAProxyCacheEntitySizeLimit *size*

**Default:** FRCAProxyCacheEntitySizeLimit 92160

**Context:** server config

**Override:** none

**Origin:** IBM

**Example:** FRCAProxyCacheEntitySizeLimit 8000

The FRCAProxyCacheEntitySizeLimit directive specifies the maximum proxy response entity size (in bytes) for FRCA to cache.

**Parameter:** *size*

- The *size* parameter value specifies the maximum proxy response entity size (in bytes) for FRCA to cache.

The below example only allows caching of proxy responses that are equal to, or less than, 8000 bytes.

```
FRCAProxyCacheEntitySizeLimit 8000
```

### **FRCAProxyCacheExpiryLimit:**

**Module:** mod\_as\_cache

**Syntax:** FRCAProxyCacheExpiryLimit *<time>*

**Default:** FRCAProxyCacheExpiryLimit 86400

**Context:** server config

**Override:** none

**Origin:** IBM

**Example:** FRCAProxyCacheExpiryLimit 3600

The FRCAProxyCacheExpiryLimit directive sets the expiration (in seconds) for FRCA proxy cached HTTP documents. Expiry time for FRCA proxy cached HTTP documents will be set to at most *nnn* number of seconds into the future. FRCA proxy cached HTTP documents can be at most *nnn* seconds out of date. If the expire header is present with the document in the response, then the lower of the two values is used.

**Parameter:** *<time>*

- The *<time>* parameter value sets the expiration (in seconds) for FRCA proxy cached HTTP documents.

FRCA proxy cached HTTP documents are limited by the specified time interval (in seconds). This restriction is enforced even if an expiration date is supplied with the HTTP document.

### **FRCAProxyCacheRefreshInterval:**

**Module:** mod\_as\_cache

**Syntax:** FRCAProxyCacheRefreshInterval *<proxy>* *<time>*

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** IBM

**Example:** FRCAProxyCacheRefreshInterval /mirror/ibm/test 30

The FRCAProxyCacheRefreshInterval directive sets the time period (in seconds) to use each cached entity, for the specified URI, before refreshing the cache.

**Parameter One:** *<path>*

- The *<path>* parameter value specifies the URI associated with cached entity.

**Parameter Two:** *<time>*

- The *<time>* parameter value sets the time period (in seconds) to use each cached entity, for the specified URI, before refreshing the cache. Possible values include integers 0 through 2,147,483,647.

**Note:** If the value specified for *<time>* is zero, then the document for the specified path are always current. That is the document is not cached.

FRCA proxy cached HTTP documents are limited by the specified interval (in seconds). This restriction is enforced even if an expiration date is supplied with the HTTP document.

### **FRCAProxyCacheSizeLimit:**

**Module:** mod\_as\_cache

**Syntax:** FRCAProxyCacheSizeLimit *size*

**Default:** FRCAProxyCacheSizeLimit 2000

**Context:** server config

**Override:** none

**Origin:** IBM

**Example:** FRCAProxyCacheSizeLimit 5000

The FRCAProxyCacheSizeLimit directive specifies the maximum amount of storage (in kilobytes) that FRCA proxy caching uses for the specified server.

**Parameter One:** *size*

- The *size* parameter value specifies the maximum amount of storage (in kilobytes) that FRCA proxy caching uses for the specified server.

The below example caches proxy response entities until the accumulated size reaches 5000 kilobytes.

```
FRCAProxyCacheSizeLimit 5000
```

**Note:** The specified value is the upper limit; the actual amount of allocated storage is the accumulated proxy entity cache size.

### **FRCAProxyPass:**

**Module:** mod\_as\_cache

**Syntax:** FRCAProxyPass *<path>* *<URL>*

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** IBM

**Example:** FRCAProxyPass /mirror/foo/ http://foo.com/

The FRCAProxyPass directive allows remote servers to map into the local server space. The local server does not act as a proxy in the conventional sense; it acts a mirror of the remote server.

**Parameter One:** *<path>*

- The *<path>* parameter value specifies the name of a local virtual path. The value is case sensitive.

**Parameter Two:** *<URL>*

- The *<URL>* parameter value specifies the partial URL for the remote server.

If the local server address is `http://ibm.com/` then `FRCAProxyPass /mirror/ibm1/ http://ibm1.com/` causes a local request for the `http://ibm.com/mirror/ibm1/location` to be internally converted into a proxy request of `http://ibm1.com/location`.

**Note:** FRCA Proxy does not perform authentication or authorization checking. Therefore, do not specify or configure paths or URLs that would result in responses with sensitive information that is not intended for public viewing.

### **FRCARandomizeResponse:**

**Module:** `mod_as_cache`

**Syntax:** `FRCARandomizeResponse <path> <string> <nnn> <mmm>`

**Default:** none

**Context:** server config

**Override:** none

**Origin:** IBM

**Example:** `FRCARandomizeResponse /some_path/fileNNN.html NNN 1 1000`

**Example:** `FRCARandomizeResponse /some_path/fileXXX.html XXX 200 300`

The `FRCARandomizeResponse` directive specifies the path template, the replacement string marker, and the random number range that you would like FRCA to randomly use when selecting and serving files of that template. For example, if you have 1000 files with names `file1.html` through `file1000.html`. By configuring `FRCARandomizeResponse /document_root_alias_path/fileNNN.html NNN 1 1000` and requesting the URL `http://some_host:port/dirpath/fileNNN.html`, FRCA will randomly select and serve one of the 1000 files.

#### **Parameter One:** `<path>`

- The `<path>` parameter value specifies valid paths in the form of `/some_path_segment/some_partial_file_name#.ext` where `"#"` is replaced with a randomly generated number by FRCA before serving the response.

**Note:** the path must begin with `'/'`. It cannot be a relative path.

#### **Parameter Two:** `<string>`

- Text `<string>` parameter value specifies the replacement string marker ("`NNN`") in the path.

#### **Parameter Three:** `<nnn>`

- The `<nnn>` parameter value specifies the lower limit for random numbers.

#### **Parameter Four:** `<mmm>`

- The `<mmm>` parameter value specifies the upper limit for random numbers.

### **LiveLocalCache:**

**Module:** `mod_as_cache`

**Syntax:** `LiveLocalCache on | off`

**Default:** `LiveLocalCache on`

**Context:** server config

**Override:** none

**Origin:** IBM

**Example:** `LiveLocalCache off`

The `LiveLocalCache` directive is used to specify if the cache is updated when a cached file is modified. Set this directive to `"on"` if you want users, requesting a cached file, to receive the file with the latest updates. Setting this directive to `"off"` is the optimum setting for performance. When `LiveLocalCache` directive is set to `"on"`, before responding to a request for a file that is stored in memory, the server

checks to see if the file has changed since the server was started. If the file has changed, the server responds to this request with the updated file. The server then deletes the older file version from memory. Restart the server to load the new file into memory.

**Parameter:** *on* | *off*

- If the parameter value is set to *on*, the cache is updated when a cached file is modified.
- If the parameter value is set to *off*, the cache is not updated when a cached file is modified.

#### | **PublicCache:**

| **Module:** mod\_as\_cache

| **Syntax:** PublicCache *on* | *off*

| **Default:** PublicCache *off*

| **Context:** server config

| **Override:** none

| **Origin:** IBM

| **Example:** PublicCache *on*

| The PublicCache directive enables caching with the CacheLocalFilePublic directive on an SSL enabled server. It is used in conjunction with the CacheLocalFilePublic directive and has a server-wide scope. See the CacheLocalFilePublic directive for additional details.

| **Parameter:** *on* | *off*

- | • If the parameter value is set to *on*, caching is turned on for directive CacheLocalFilePublic.
- | • If the parameter value is set to *off*, caching is turned off for directive CacheLocalFilePublic.

## **Module mod\_asis**

Module mod\_asis contains directives for the IBM HTTP Server for i Web server.

### **Summary**

This module allows file types to be defined such that HTTP Server sends them without adding HTTP headers. The mod\_asis module supports EBCDIC files. The module converts file content from EBCDIC to ASCII. This can be used to send any kind of data from the server, including redirects and other special HTTP responses, without requiring a CGI script.

### **Usage**

In the server configuration file, define a new mime type called httpd/send-as-is . For example:

```
AddType httpd/send-as-is asis
```

This defines the .asis file extension as being of the new httpd/send-as-is mime type. The contents of any file with a .asis extension are then be sent by HTTP Server to the client with almost no changes. Clients will need HTTP headers to be attached. A Status: header is also required; the data should be the 3-digit HTTP response code, followed by a textual message.

Here is an example of a file whose contents are sent asis, telling the client that a file has redirected.

```
Status: 301 Now where did I leave that URL
Location: http://xyz.abc.com/roch/bar.html
Content-type: text/html
```

```
<HTML>
<HEAD>
<TITLE>Fred's Page</TITLE>
</HEAD>
<BODY>
<H1>Fred's exceptionally wonderful page has moved to
```

```
<A HREF="http://xyz.abc.com/roch/bar.html">Joe's</A> site.  
</H1>  
</BODY>  
</HTML>
```

**Note:** The server adds a Date: and Server: header to the data returned to the client.

## Module mod\_autoindex

Module mod\_autoindex contains directives for the IBM HTTP Server for i Web server.

### Summary

The module mod\_autoindex provides for automatic directory indexing. The index of a directory can come from one of two sources:

- A file written by the user, typically called index.html. The “DirectoryIndex” on page 342 directive sets the name of this file. This is controlled by mod\_dir.
- A listing generated by the server. The other directives control the format of this listing. The AddIcon, AddIconByEncoding and AddIconByType are used to set a list of icons to display for various file types; for each file listed, the first icon listed that matches the file is displayed. These are controlled by mod\_autoindex.

If the FancyIndexing keyword is present on the IndexOptions directive, the column headers are links that control the order of the display. If you select a header link, the listing will be regenerated, sorted by the values in that column. Selecting the same header repeatedly toggles between ascending and descending order.

For all mod\_autoindex directives that specify a file name (AddDescription, AddIcon, and so on), case sensitivity is handled based on the file system. If the object is in the QOpenSys file system, the name is handled in a case sensitive manner. If the object is a file system other than QOpenSys, the name is handled in a case insensitive manner.

**Note:** When the display is sorted by "Size", it is the actual size of the files that's used, not the displayed value - so a 1010-byte file will always be displayed before a 1011-byte file (if in ascending order) even though the size of both files could be displayed as "1K".

### Directives

- “AddAlt” on page 249
- “AddAltByEncoding” on page 249
- “AddAltByType” on page 249
- “AddDescription” on page 250
- “AddIcon” on page 251
- “AddIconByEncoding” on page 251
- “AddIconByType” on page 251
- “DefaultIcon” on page 252
- “HeaderName” on page 252
- “IndexHeadInsert” on page 253
- “IndexIgnore” on page 253
- “IndexOptions” on page 254
- “IndexOrderDefault” on page 256
- “IndexStyleSheet” on page 257
- “ReadmeName” on page 257

## AddAlt:

**Module:** mod\_autoindex

**Syntax:** AddAlt *string file [file...]*

**Default:** none

**Context:** server config, virtual host, directory (but not location), .htaccess

**Override:** Indexes

**Origin:** Apache

**Example:** AddAlt "IMG" jpg gif

The AddAlt directive sets the alternate text to display for automatic directory indexing.

### Parameter One: *string*

- The *string* parameter is enclosed in double quotes ("..."). This alternate text is displayed if the client is image-incapable or has image loading disabled.

### Parameter Two: *file*

- The *file* parameter is either `^^DIRECTORY^^` for child directories, `^^PARENT^^` for parent directories, `^^BLANKICON^^` for blank lines (to format the list correctly), a file extension, a wildcard expression, a partial file, or a complete filename. It could also be a QSYS.LIB member type if this directive is being used to set alternate text for QSYS.LIB members. For example:

```
AddAlt "IMG" .jpg .gif
AddAlt " " ^^BLANKICON^^
AddAlt "BAK" *~
```

**Note:** This directive is not supported in "<Location>" on page 312 containers.

## AddAltByEncoding:

**Module:** mod\_autoindex

**Syntax:** AddAltByEncoding *string MIME-encoding [MIME-encoding...]*

**Default:** none

**Context:** server config, virtual host, directory (but not location), .htaccess

**Override:** Indexes

**Origin:** Apache

**Example:** AddAltByEncoding "CMP" x-compress

The AddAltByEncoding directive sets the alternate text to display for a file, instead of an icon, for automatic directory indexing.

### Parameter One: *string*

- The *string* parameter is enclosed in double quotes ("..."). This alternate text is displayed if the client is image-incapable or has image loading disabled.

### Parameter Two: *MIME-encoding*

- The *MIME-encoding* parameter is a valid content-encoding, such as x-compress.

**Note:** This directive is not supported in "<Location>" on page 312 containers.

## AddAltByType:

**Module:** mod\_autoindex

**Syntax:** AddAltByType *string MIME-type [MIME-type ...]*

**Default:** none

**Context:** server config, virtual host, directory (but not location), .htaccess

**Override:** Indexes

**Origin:** Apache

**Example:** AddAltByType "HTM" text/html

The AddAltByType directive sets the alternate text to display for a file, instead of an icon, for automatic directory indexing.

**Parameter One:** *string*

- The *string* parameter is enclosed in double quotes ("..."). This alternate text is displayed if the client is image-incapable or has image loading disabled.

**Parameter Two:** *MIME-type*

- The *MIME-type* parameter is a valid content-type, such as text/html.

**Note:** This directive is not supported in "<Location>" on page 312 containers.

### AddDescription:

**Module:** mod\_autoindex

**Syntax:** AddDescription *string file [file...]*

**Default:** none

**Context:** server config, virtual host, directory (but not location), .htaccess

**Override:** Indexes

**Origin:** Apache

**Example:** AddDescription "Famous People" /web/pics/famous\*

**Example:** AddDescription "My pictures" /QSYS.LIB/MYLIB/MYFILE.FILE/pic\*

The AddDescription directive sets the description to display for a file, for automatic directory indexing. File is a file extension, partial filename, QSYS.LIB member type, wildcard expression or full filename for files to describe. String is enclosed in double quotes (""). For example:

```
AddDescription "The planet Mars" /web/pics/mars.gif
```

By default, the description field is 23 bytes wide. Seven more bytes may be added if the directory is covered by an IndexOptions SuppressSize, and 19 bytes may be added if IndexOptions SuppressLastModified is in effect. The widest this column can be is therefore 49 bytes, unless configured differently using IndexOptions DescriptionMaxWidth.

The DescriptionWidth IndexOptions keyword allows you to adjust this width to any arbitrary size.

The following order of precedence will be used to search for a directory listing file description. The first mechanism from this list that applies will be used to generate the file description:

1. The file matches one of those specified on an AddDescription directive. The string from the directive is displayed. This option is the least CPU intensive.
2. The file system contains a description for the file. The file system description information is displayed. Note that if the file is a QSYS.LIB member, the member text is displayed.
3. If IndexOptions ScanHTMLTitles is configured, the title is extracted from HTML documents for fancy indexing. This is CPU and disk intensive.

**Note:** Descriptive text defined with AddDescription may contain HTML markup, such as tags and character entities. If the width of the description column should happen to truncate a tagged element (such as cutting off the end of a bold phrase), the results may affect the rest of the directory listing. This directive is not supported in "<Location>" on page 312 containers.



## AddIcon:

**Module:** mod\_autoindex

**Syntax:** AddIcon *icon name* [*name ...*]

**Default:** none

**Context:** server config, virtual host, directory (but not location), .htaccess

**Override:** Indexes

**Origin:** Apache

**Example:** AddIcon (IMG,icons/image) .gif .jpg

The AddIcon directive sets the icon to display next to a file ending in name for automatic directory indexing.

**Parameter One:** *icon*

- The *icon* parameter is either a (%-escape) relative URL to the icon or of the format (alttext,url) where alttext is the text tag given for an icon for non-graphical browsers.

**Parameter Two:** *name*

- The *name* parameter is either `^^DIRECTORY^^` for child directories, `^^PARENT^^` for parent directories, `^^BLANKICON^^` for blank lines (to format the list correctly), a file extension, a wildcard expression, a partial file or a complete filename. For example

```
AddIcon (IMG,icons/image) .gif .jpg
AddIcon (PAR, icons/parent .gif) ^^PARENT^^
AddIcon /dir.gif ^^DIRECTORY^^
AddIcon backup.gif *~
```

“AddIconByType” should be used in preference to AddIcon, when possible.

**Note:** This directive is not supported in “<Location>” on page 312 containers.

## AddIconByEncoding:

**Module:** mod\_autoindex

**Syntax:** AddIconByEncoding *icon MIME-encoding* [*MIME-encoding ...*]

**Default:** none

**Context:** server config, virtual host, directory (but not location), .htaccess

**Override:** Indexes

**Origin:** Apache

**Example:** AddIconByEncoding /compress.xbm x-compress

The AddIconByEncoding directive sets the icon to display next to files with MIME-encoding for automatic directory indexing.

**Parameter One:** *icon*

- The *icon* parameter is either a (%-escaped) relative URL to the icon or of the format (alttext,url) where alttext is the text tag five for an icon for non-graphical browsers.

**Parameter Two:** *MIME-encoding*

- The *MIME-encoding* parameter is a wildcard expression matching required content-encoding.

**Note:** This directive is not supported in “<Location>” on page 312 containers.

## AddIconByType:

**Module:** mod\_autoindex

**Syntax:** AddIconByType *icon MIME-type* [*MIME-type ...*]

**Default:** none  
**Context:** server config, virtual host, directory (but not location), .htaccess  
**Override:** Indexes  
**Origin:** Apache  
**Example:** AddIconByType (IMG,image.gif) image/\*

The AddIconByType directive sets the icon to display next to files of type MIME-type for FancyIndexing. Icon is either a (%-escaped) relative URL to the icon, or of the format (alttext,url) where alttext is the text tag given for an icon for non-graphical browsers.

**Parameter One:** *icon*

- The icon parameter is either a (%-escaped) relative URL to the icon or of the format (alttext,url) where alttext is the text tag given for an icon for non-graphical browsers.

**Parameter Two:** *MIME-type*

- The MIME-type parameter is a wildcard expression matching the required MIME types.

**Note:** This directive is not supported in “<Location>” on page 312 containers.

**DefaultIcon:**

**Module:** mod\_autoindex  
**Syntax:** DefaultIcon *url*  
**Default:** none  
**Context:** server config, virtual host, directory (but not location), .htaccess  
**Override:** Indexes  
**Origin:** Modified  
**Example:** DefaultIcon /icon/unknown.gif

The DefaultIcon directive sets the icon to display for files when no specific icon is known, for automatic directory indexing.

**Parameter:** *url*

- The url parameter is either a (%-escaped) relative URL to the icon or of the format (alttext,url) where alttext is the text tag given for an icon for non-graphical browsers. For example:  
DefaultIcon (UNK,unknown.gif)

**Note:** This directive is not supported in “<Location>” on page 312 containers.

**HeaderName:**

**Module:** mod\_autoindex  
**Syntax:** HeaderName *filename*  
**Default:** none  
**Context:** server config, virtual host, directory (but not location), .htaccess  
**Override:** Indexes  
**Origin:** Apache  
**Example:** HeaderName headerfile  
**Example:** HeaderName PREAMBLE.MBR

The HeaderName directive sets the name of the file that will be inserted at the top of the index listing.

**Parameter:** *filename*

- The *filename* parameter is the name of the file to include.

Filename is treated as a URI path relative to the one used to access the directory being indexed, and must resolve to a document with a major content type of "text" (for example, text/html, text/plain). This means that filename may refer to a CGI script if the script's actual file type (as opposed to its output) is marked as text/html such as with a directive like:

```
AddType text/html .cgi
```

Content negotiation will be performed if the MultiViews option is enabled. See "Content negotiation for HTTP Server" on page 17 for more information.

If filename resolves to a static text/html document (not a CGI script) and the Includes Option is enabled, the file will be processed for server-side includes. See `mod_include` for more information.

See also "ReadmeName" on page 257.

**Note:** This directive is not supported in "<Location>" on page 312 containers.

#### | **IndexHeadInsert:**

| **Module:** mod\_autoindex  
| **Syntax:** IndexHeadInsert *markup*  
| **Default:** none  
| **Context:** server config, virtual host, directory, .htaccess  
| **Override:** Indexes  
| **Origin:** Apache  
| **Example:**  
| IndexHeadInsert "<link rel=\"sitemap\" href=\"/sitemap.html\">"  
|

| The IndexHeadInsert directive specifies a string to insert in the <head> section of the HTML generated for the index page.

| **Parameter:** *markup*

- The *markup* parameter is a string to be inserted in the <head> section of the HTML generated for the index page. The *markup* parameter must be enclosed in double quotes ("...").

#### **IndexIgnore:**

**Module:** mod\_autoindex  
**Syntax:** IndexIgnore file [file ...]  
**Default:** none  
**Context:** server config, virtual host, directory (but not location), .htaccess  
**Override:** Indexes  
**Origin:** Apache  
**Example:** IndexIgnore README .htaccess  
**Example:** IndexIgnore README.MBR

The IndexIgnore directive adds to the list of files to hide when listing a directory. Multiple IndexIgnore directives add to the list, rather than replacing the list of ignored files. By default, the dot directory (.) is ignored.

**Parameter:** *file*

- The *file* parameter is a file extension, QSYS.LIB member type, partial filename, wildcard expression or full filename for files to ignore.

**Note:** This directive is not supported in "<Location>" on page 312 containers.

## IndexOptions:

**Module:** mod\_autoindex

**Syntax:** IndexOptions [+|-]option [+|-]option ...

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** Indexes

**Origin:** Apache

**Example:** IndexOptions FancyIndexing ShowOwner FoldersFirst

The IndexOptions directive specifies the behavior of the directory indexing. The option parameter can be one of the following:

### DescriptionWidth= [n | \*]

The DescriptionWidth keyword allows you to specify the width of the description column in characters. -DescriptionWidth (or unset) allows mod\_autoindex to calculate the best width. DescriptionWidth=n fixes the column width to n characters wide. DescriptionWidth=\* grows the column to the width necessary to accommodate the longest description string. See the section on AddDescription for dangers inherent in truncating descriptions.

### FancyIndexing

This option turns on fancy indexing of directories. With FancyIndexing, the column headers are links that control the order of the display. If you select a header link, the listing will be regenerated, sorted by the values in that column. Selecting the same header repeatedly toggles between ascending and descending order.

### FoldersFirst

If this option is enabled, subdirectories in a FancyIndexed listing will always appear first, followed by normal files in the directory. The listing is broken into two components, the files and the subdirectories, and each is sorted separately and then displayed (subdirectories-first). For instance, if the sort order is descending by name, and FoldersFirst is enabled, subdirectory Zed will be listed before subdirectory Beta, which will be listed before normal files Gamma and Alpha. This option only has an effect if FancyIndexing is also enabled

### IconsAreLinks

This makes the icons part of the anchor for the filename, for fancy indexing.

### IconHeight=[pixels]

Presence of this option, when used with IconWidth, will cause the server to include HEIGHT and WIDTH attributes in the IMG tag for the file icon. This allows browser to precalculate the page layout without having to wait until all the images have been loaded. If no value is given for the option, it defaults to the standard height of the icons supplied with the the HTTP Server software.

### IconWidth=[pixels]

Presence of this option, when used with IconHeight, will cause the server to include HEIGHT and WIDTH attributes in the IMG tag for the file icon. This allows browser to precalculate the page layout without having to wait until all the images have been loaded. If no value is given for the option, it defaults to the standard width of the icons supplied with the HTTP Server software.

### IgnoreCase

If this option is enabled, names are sorted in a case-insensitive manner. For instance, if the sort order is ascending by name, and IgnoreCase is enabled, file zeta will be listed after file Alpha. Likewise, if IgnoreCase is disabled, file zeta will be listed before file Alpha. By default IgnoreCase is disabled. This option only has an effect if FancyIndexing is also enabled. The new IgnoreCase value replaces the IndexOrderDefault CaseSense | NoCaseSense parameter.

**IgnoreClient**

This option causes mod\_autoindex to ignore all query variables from the client, including sort order (implies SuppressColumnSorting.)

**NameWidth=[n | \*]**

The NameWidth keyword allows you to specify the width of the filename column in characters. If the keyword value is '\*', then the column is automatically sized to the length of the longest filename in the display. -NameWidth (or unset) allows mod\_autoindex to calculate the best width. NameWidth=n fixes the column width to n characters wide. The minimum value allowed is 5.

**NameMinWidth=[n]**

The NameMinWidth keyword allows you to specify the minimum width that will always be reserved for the filename column in characters. The default setting is 15. The minimum value allowed is 5. If NameMinWidth is greater than NameWidth, then the filename column will have a length=NameMinWidth.

**ScanHTMLTitles**

This enables the extraction of the title from HTML documents for fancy indexing. If the file does not have a description given by AddDescription then the HTTP Server will read the document for the value of the TITLE tag. This is CPU and disk intensive.

**SelectiveDirAccess**

This option will cause the server to return directory listings only for directories that contain a wwwbrws file. The contents of wwwbrws file are not important. The server only checks for its existence. The object is a member name of an IBM i physical file or a type of object in an integrated file system directory. For case-sensitive file systems such as /QOpenSys, the wwwbrws name is lowercase. \*\*SelectiveDirAccess is an AS400 specific option. This specific option works on a "per directory" basis, in other words you must specify the +/-SelectiveDirAccess on a Directory container.

**ShowForbidden**

This option is new for Apache 2.2. If you use this option, Apache will show files normally hidden because the subrequest returned HTTP\_UNAUTHORIZED or HTTP\_FORBIDDEN.

**ShowOwner.**

This directive determines whether directory listings should include the owner ID for each file.

**SuppressColumnSorting**

If specified, the HTTP Server will not make the column headings in a FancyIndexed directory listing into links for sorting. The default behavior is for them to be links; selecting the column heading will sort the directory listing by the values in that column.

**SuppressDescription**

This will suppress the file description in fancy indexing listings. See AddDescription for information about setting the file description. See also the DescriptionWidth index option to limit the size of the description column.

**SuppressHTMLPreamble**

If the directory actually contains a file specified by the HeaderName directive, the module usually includes the contents of the file after a standard HTML preamble (<HTML> <HEAD>). The SuppressHTMLPreamble option disables this behavior, causing the module to start the display with the header file contents. The header file must contain appropriate HTML instructions in this case. If there is no header file, the preamble is generated as usual.

**SuppressIcon**

This directive suppresses the display of icons on directory listings. The default is that no options are enabled.

**SuppressLastModified**

This directive will suppress the display of the last modification date, in fancy indexing listings.

### **SuppressRules**

This directive will suppress the horizontal rule lines (HR tags) in directory listings. Combining both SuppressIcon and SuppressRules yields proper HTML 3.2 output, which by the final specification prohibits IMG and HR tags from the PRE block used to format FancyIndexed listings.

### **SuppressSize**

This directive will suppress the file size in fancy indexing listings.

### **TrackModified**

This returns the Last-Modified and ETag values for the listed directory in the HTTP header. It is only valid if the operating system and file system return appropriate stat() results. Once this feature is enabled, the client or proxy can track changes to the list of files when they perform a HEAD request. Changes to the size or date stamp of an existing file will not update the Last-Modified header on all Unix platforms. If this is a concern, leave this option disabled.

### **VersionSort**

The VersionSort keyword causes files containing version numbers to sort in a natural way. Strings are sorted as usual, except that substrings of digits in the name and description are compared according to their numeric value. For example:

- foo-1.73
- foo-1.7.2
- foo-1.7.12
- foo-1.8.2
- foo-1.8.2a
- foo-1.12

If the number starts with a zero, then it is considered to be a fraction:

- foo-1.001
- foo-1.002
- foo-1.030
- foo-1.04

### **XHTML**

This keyword is available in Apache 2.0.49, or later. The XHTML keyword forces mod\_autoindex to emit XHTML 1.0 code instead of HTML 3.2.

Multiple IndexOptions directives for a single directory are merged together. The directive allows incremental syntax (i.e., prefixing keywords with '+' or '-'). Whenever a '+' or '-' prefixed keyword is encountered, it is applied to the current IndexOptions settings (which may have been inherited from an upper-level directory). However, whenever a non-prefixed keyword is processed, it clears all inherited options and any incremental settings encountered so far. Consider the following example:

```
IndexOptions +ScanHTMLTitles -IconsAreLinks FancyIndexing
IndexOptions +SuppressSize
```

The net effect is equivalent to IndexOptions FancyIndexing +SuppressSize, because the non-prefixed FancyIndexing discarded the incremental keywords before it, but allowed them to start accumulating again afterward. To unconditionally set the IndexOptions for a particular directory, clearing the inherited settings, specify keywords without either '+' or '-' prefixes.

**Note:** IndexOptions directive is not supported in <Location> containers.

### **IndexOrderDefault:**

**Module:** mod\_autoindex

**Syntax:** IndexOrderDefault [ *ascending* | *descending* ] [ *name* | *date* | *size* | *owner* | *description* ] [ *CaseSense* | *NoCaseSense* ]

**Default:** IndexOrderDefault Ascending Name CaseSense

**Context:** server config, virtual host, directory (but not location), .htaccess

**Override:** Indexes

**Origin:** Modified

**Example:** IndexOrderDefault descending size

The IndexOrderDefault directive is used in combination with the FancyIndexing index option. By default, FancyIndexed directory listings are displayed in ascending order by filename; the IndexOrderDefault allows you to change this initial display order.

IndexOrderDefault takes two required arguments and a third optional argument.

**Parameter One:** *ascending* | *descending*

- The *ascending* and *descending* parameter indicates the direction of the sort.

**Parameter Two:** *name* | *date* | *size* | *owner* | *description*

- The *name*, *date*, *size*, *owner*, and *description* parameter arguments must be used and identifies the primary key. The secondary key is always ascending filename.

**Parameter Three:** *CaseSense* | *NoCaseSense*

- The *CaseSense* and *NoCaseSense* parameters are optional third keywords that allow you to choose if the column sort is case sensitive. This keyword is valid if the second keyword is *name*, *owner* or *description* only. If the second keyword is *date* or *size*, then this parameter is ignored. The default for keyword is *CaseSense*.

You can force a directory listing to only be displayed in a particular order by combining this directive with the SuppressColumnSorting index option; this will prevent the client from requesting the directory listing in a different order.

**Note:** This directive is not supported “<Location>” on page 312 containers. The directive may be inherited in a “<Directory>” on page 291 context, but not in a “<VirtualHost>” on page 332 context.

## IndexStyleSheet:

**Module:** mod\_autoindex

**Syntax:** IndexStyleSheet *url-path*

**Default:** none

**Context:** Server, Virtual Host, Directory, .htaccess

**Override:** Indexes

**Origin:** Apache

**Example:** IndexStyleSheet "/css/style.css"

The IndexStyleSheet directive sets the name of the file that will be used as the CSS for the index listing.

## ReadmeName:

**Module:** mod\_autoindex

**Syntax:** ReadmeName *filename*

**Default:** none

**Context:** server config, virtual host, directory (but not location), .htaccess

**Override:** Indexes

**Origin:** Apache

**Example:** ReadMeName readme  
**Example:** ReadMeName README.MBR

The ReadmeName directive sets the name of the file that will be appended to the end of the index listing.

**Parameter:** *filename*

- The *filename* parameter is the name of the file to include and is taken to be relative to the location being indexed. Details of how its handled may be found under the description of the “HeaderName” on page 252 directive, which uses the same mechanism as ReadmeName.

**Note:** This directive is not supported in “<Location>” on page 312 containers.

## Module **mod\_cern\_meta**

Module mod\_autoindex supports directives for the IBM HTTP Server for i Web server.

### Summary

The module mod\_cern\_meta provides for CERN httpd meta file semantics.

### Directives

- “MetaFiles”
- “MetaDir”
- “MetaSuffix” on page 259

### MetaFiles:

**Module:** mod\_cern\_meta  
**Syntax:** MetaFiles *on* | *off*  
**Default:** MetaFiles off  
**Context:** directory  
**Override:** none  
**Origin:** Apache  
**Example:** MetaFiles on

This directive allows you to emulate the CERN Meta file semantics. Meta files are HTTP headers that can be output in addition to the normal range of headers for each file accessed. They appear like the .asis files, and are able to influence the Expires: header.

**Parameter:** *on* | *off*

- Turns *on* or *off* the meta file processing on a per-directory basis.

### MetaDir:

**Module:** mod\_cern\_meta  
**Syntax:** MetaDir *directoryname*  
**Default:** MetaDir .web  
**Context:** directory  
**Override:** none  
**Origin:** Apache  
**Example:** MetaDir .meta Specifies the name of the



Specifies the name of the directory where HTTP Server can find meta information files. The directory is usually a hidden subdirectory of the directory that contains the file being accessed. Set directory name to "." in order to look in the same directory as the file. Hidden directories begin with "." so the default directory will be hidden.

**Parameter:** *directoryname*

- The *directoryname* parameter specifies the name of the directory in which HTTP Server can find meta information files.

### **MetaSuffix:**

**Module:** mod\_cern\_meta

**Syntax:** MetaSuffix *suffix*

**Default:** MetaSuffix .meta

**Context:** directory

**Override:** none

**Origin:** Apache

**Example:** MetaSuffix .stuff

Specifies the file name suffix for the file containing the meta information. A request will cause the server to look in the file with the MetaSuffix in the "MetaDir" on page 258 directory, and will use its contents to generate additional MIME header information.

**Parameter:** *suffix*

- The *suffix* parameter is the file name suffix of the file containing the meta information.

## **Module mod\_cache**

Module mod\_cache contains directives for the IBM HTTP Server for i Web server.

### **Summary**

This module contains directives that define support for the HTTP Proxy function which includes the proxy caching function.

### **Cache Expiry Times**

Cache expiry times are different than expiry times provided in HTTP response data. Cache expiry times are calculated by caching agents (such as a proxy server), whereas expiry times in HTTP response data are provided by content servers (for example, via HTTP Expires headers). If cacheable data from content servers contains expiry times, a caching agent (or proxy) must use cache expiry times that are no later than the corresponding data expiry times. In other words, caching agents may not serve data from cache after it has expired, however they may stop serving it from cache prior to such time.

If content servers do not provide expiry times for cacheable data, the caching agent (or proxy) may try to use other response information to calculate acceptable cache expiry times, or it may use some arbitrary default value, as determined by the administrator.

**Note:** Response data is considered cacheable for the proxy function if it satisfies criteria described under "Criteria for Local Proxy Cache" on page 260.

The proxy function follows these rules to determine which directive settings to use to calculate cache expiry times for HTTP proxy response data stored in the local proxy cache.

1. If HTTP response data contains expiry times (via Expires header for HTTP requests only) these times are also used as cache expiry times.

2. If HTTP response data does not contain expiry times, but does contain information pertaining to when it was last modified (via Last-Modified header for HTTP requests, or MDTM command for FTP requests), the CacheLastModifiedFactor and CacheMaxExpire directive settings are used to calculate cache expiry times.
3. If HTTP response data does not contain expiry times, nor does it contain information pertaining to when it was last modified, the CacheDefaultExpire directive setting is used to calculate arbitrary cache expiry times.

**Note:** The first rule has one exception. If response code 304 (Not Modified) is received for HTTP requests, Expires headers (if any) are not used to set new cache expiry time. The second rule is applied (for 304 responses) if last modified times from cached data are available to recalculate new cache expiry times. If last modified times are not available from cached data, the third rule is applied.

## Criteria for Local Proxy Cache

When configured, the server handles certain requests using the proxy function to obtain data from remote servers, which it then serves as HTTP proxy response data. It does this when acting as either a forward proxy or a reverse proxy (see ProxyRequests or ProxyReverse). By default, the proxy function obtains and handles data separately for each request. The server may be made more efficient, however, by using a local proxy cache to store HTTP proxy response data locally, which it then serves multiple times for multiple requests. The server is more efficient since remote servers need only be contacted when data in the local proxy cache expires.

Not all response data obtained by the server is cached and served for multiple requests, due mostly for reasons involving privacy, version control (frequency of change), and negotiable content. This type of response data is not considered cacheable and must be obtained from remote servers for each request.

### Standard Criteria

Standard criteria for the server's local proxy cache and proxy function, in regards to response data obtained using specified protocols, is described in the following lists. This criteria is used to determine whether HTTP proxy response data is cacheable and may be served multiple times for multiple requests.

HTTP response data:

- Only data requested using the GET method is cacheable.
- Only data received on a request that does not end with a '/' is cacheable.
  - 200 (OK)
  - 203 (Non Authoritative)
  - 300 (Multiple Choices)
  - 301 (Moved Permanently)
  - 304 (Not Modified)
- If data contains an Expires header, the header must be valid.

**Note:** This does not apply to data that does not contain an Expires header.

- If data contains an Expires header, the header must not specify a time that has already past (according to local system time).
- If data contains an Expires header, the expiration time must be greater than the configured minimum expiration time.
- Data received with response code 200 (OK) must contain either a Last-Modified header or an ETag header. This requirement is waived if on is specified for the CacheIgnoreNoLastMod directive.
- Data received with response code 304 (Not Modified) is not cacheable if a previous version is not already in cache.

- If data contains a Cache-Control header, the header must not specify the value "no-store" or "private".
- If data contains a Pragma header, the header must not specify the value "no-cache".
- If the request provides an Authorization header (possibly used by the remote server), response data must contain a Cache-Control header that specifies one or more of the following values: "s-maxage", "must-revalidate" or "public".
- If data contains a Content-Length header, the header must not specify a value that exceeds the minimum or maximum data size limits set by the CacheMinFileSize and CacheMaxFileSize directives. See Additional Criteria for more information.

FTP response data:

- Only data requested using the GET method is cacheable.
- Data is only cached if LIST or RETR commands return one of the following response codes:
  - 125 (OK, Data Transfer Starting)
  - 150 (OK, Opening Data Connection)
  - 226 (OK, Closing Data Connection)
  - 250 (OK)

**Note:** The LIST command is used to retrieve directory listings. The RETR command is used to retrieve data files.

- Data must contain information for an HTTP Last-Modified header (produced via MDTM command with response code 213, see Notes<sup>®</sup>: below). This requirement is waived if on is specified for the CacheIgnoreNoLastMod directive.
- If data contains information for an HTTP Content-Length header (produced via SIZE command with response code 213), the header must not specify a value that exceeds the minimum or maximum data size limits set by the CacheMinFileSize and CacheMaxFileSize directives, respectively. See Additional Criteria for more information.

HTTPS (or SSL-tunneling over HTTP) response data:

- Data requested using SSL-tunneling over HTTP is not cacheable.

No other protocols are supported by the proxy function.

### Additional Criteria

Additional criteria for the server's local proxy cache and proxy functions may be imposed by the function providing underlying cache support. Currently, this includes only the disk cache function.

The following list describes additional restrictions on HTTP proxy response data stored in a local proxy cache, imposed by the mod\_disk\_cache module:

- Cache data must not exceed the minimum or maximum data size limits set by the CacheMinFileSize and CacheMaxFileSize directives. This restriction applies regardless of Content-Length header values (if any) in HTTP proxy response data.
- Data with cache expiry times that will expire within the minimum time margin set by the CacheTimeMargin directive is not cached. This restriction applies to HTTP proxy response data, using cache expiry times calculated according to rules described in the Cache Expiry Times. See mod\_disk\_cache for other restriction that may apply.

### Directives

- "CacheDefaultExpire" on page 262
- "CacheDisable" on page 263
- "CacheEnable" on page 263

- “CacheExpiryCheck” on page 264
- “CacheIgnoreCacheControl” on page 264
- “CacheIgnoreNoLastMod” on page 265
- “CacheIgnoreHeaders” on page 265
- “CacheLastModifiedFactor” on page 266
- “CacheMaxExpire” on page 267
- “CacheStoreNoStore” on page 270
- “CacheStorePrivate” on page 270
- “CacheTimeMargin” on page 271

### CacheDefaultExpire:

**Module:** mod\_cache

**Syntax:** CacheDefaultExpire *period*

**Default:** CacheDefaultExpire 3600

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule cache\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** CacheDefaultExpire 3

The CacheDefaultExpire directive specifies the default number of seconds in which cacheable HTTP proxy response data will be set to expire within the local proxy cache, starting from the time it is obtained by the server.

**Parameter:** *period*

- The *period* parameter defines the default cache expiry period, in seconds.

This setting is used to calculate arbitrary cache expiry times for HTTP proxy response data stored in the local proxy cache. See Cache Expiry Times for more information on how the server determines which settings to use to calculate cache expiry times. See the CacheIgnoreNoLastMod directive for information relating to how cache criteria may be waived for this setting to take affect.

If this setting is used, cache expiry times are calculated by adding the specified number of seconds to the time that data is received by the proxy function.

**Example:**

```
ProxyRequests on
CacheRoot proxyCache
CacheDefaultExpire 3600
CacheMaxExpire 86400
CacheLastModifiedFactor 0.3
```

In the example, if a cacheable data is retrieved from a server that does not provide an expiry time (via HTTP Expires header), nor does it indicate when the data was last modified (via HTTP Last-Modified header, or FTP MDTM command), the server will cache and serve the data for 3600 seconds (since CacheDefaultExpire is set to 3600 and "on" is specified for CacheIgnoreNoLastMod). If an expiry time or last-modified time is provided, CacheDefaultExpire would not be used (see Cache Expiry Times).

**Note:** Response data is considered cacheable for the proxy function if it satisfies criteria described under Criteria for Local Proxy Cache.

- Setting ProxyRequests and ProxyReverse to off negates this directive.
- Setting ProxyNoConnect to on negates this directive.

- This directive is used only if CacheRoot is set.

### CacheDisable:

**Module:** mod\_cache

**Syntax:** CacheDisable url-string

**Default:** none

**Context:** server, virtual host

**Override:** none

**Origin:** Apache

**Example:** CacheDisable /local\_files

The CacheDisable directive instructs mod\_cache to not cache urls at or below url-string. Implementing CacheDisable will not make ProxyNoCache obsolete. CacheDisable and ProxyNoCache can be used in conjunction with each other (CacheDisable will have precedence)

### CacheEnable:

**Module:** mod\_cache

**Syntax:** CacheEnable cache\_type url-string

**Default:** none

**Context:** server, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule cache\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGMcache\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** CacheEnable disk /

The CacheEnable directive instructs mod\_cache to cache urls at or below url-string. The cache storage manager is specified with the cache\_type attribute (note: we do not support types fd or mem - mod\_mem\_cache). cache\_type disk instructs mod\_cache to use the disk based storage manager implemented by mod\_disk\_cache.

When acting as a forward proxy server, url-string can also be used to specify remote sites and proxy protocols which caching should be enabled for.

#### Example 1

```
# Cache url's
CacheEnable disk /
```

#### Example 2

```
# Cache FTP-proxied url's
CacheEnable disk ftp://
```

#### Example 3

```
#Cache content from www.apache.org
CacheEnable disk http://www.apache.org/
```

The IBM i server only supports cache\_type disk (mod\_disk\_cache). If you wish to improve caching performance, use FRCA or memory based local cache mechanisms (CacheLocal directives). Implementing the CacheEnable directive will not make DynamicCache obsolete. URL's specified by CacheEnable will take precedence over dynamic cache, and will mark the request as not being a candidate for Dynamic Cache.

## CacheExpiryCheck:

**Module:** mod\_cache

**Syntax:** CacheExpiryCheck *on* | *off*

**Default:** CacheExpiryCheck on

**Context:** server config, virtual host

**Override:** none

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule cache\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** CacheExpiryCheck on

The CacheExpiryCheck directive specifies whether the server is to observe cache expiry times when cached data is requested using the disk cache function (see CacheRoot).

**Parameter:** *on* | *off*

- If *on* is specified (the default), the server will perform and apply all cache expiry time checks for data currently available in cache.
- If *off* is specified, cache expiry times will not be observed and cached data (if any) will always be available.

Cache expiry time checks may be disabled (*off*) when the content of the cache is managed by an application or process other than the server itself. If the content of the cache is not managed by an application or process other than the server, this setting must be set to *on* (the default) to prevent the disk cache function from making expired data appear valid.

**Note:** When the disk cache function is used to support a local proxy cache, this setting determines whether cache expiry times are observed for the proxy function. Once cached, data is usually available from cache until its respective cache expiry times has passed. However, if cache expiry time checks are disabled (CacheExpiryCheck *off*), the proxy function will serve cached HTTP proxy response data regardless of whether it has expired. This effectively causes the disk cache function to ignore cache expiry times calculated using the CacheDefaultExpire, CacheMaxExpire, and CacheLastModifiedFactor directives for a local proxy cache, as well as any expiry time provided via Expires headers (for HTTP requests).

See the CacheRoot directive for more information on how the disk cache function is used to support a local proxy cache.

## CacheIgnoreCacheControl:

**Module:** mod\_cache

**Syntax:** CacheIgnoreCacheControl *on* | *off*

**Default:** CacheIgnoreCacheControl off

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule cache\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** CacheIgnoreCacheControl on

The CacheIgnoreCacheControl directive specifies whether the server is to observe certain cache controlling request headers (for example, Cache-Control and Pragma) when handling requests using the proxy function.

**Parameter:** *on* | *off*

- If *on* is specified, the server will not observe cache controlling request headers.
- If *off* is specified (the default), the server will observe cache controlling request headers when HTTP proxy response data is available from the local proxy cache.

By default, the server observes certain cache controlling request headers (for example, "Cache-Control : no-store" and "Pragma : no-cache") when handling requests using the proxy function. If such headers are present in HTTP request data sent to the server, the proxy function will not serve HTTP proxy response data from the local proxy cache since these headers indicate that cached data is not wanted. However, if *on* is specified for this setting, the proxy function will ignore cache controlling request headers and serve HTTP proxy response data from cache, if it is available.

- Setting ProxyRequests and ProxyReverse to off negates this directive.
- This directive is used only if CacheRoot is set.

### CacheIgnoreHeaders:

**Module:** mod\_cache

**Syntax:** CacheIgnoreHeaders *header-string* [*header-string*]

**Default:** CacheIgnoreHeaders None

**Context:** server, virtual host

**Override:** none

**Origin:** Apache

**Example 1:** CacheIgnoreHeaders Set-Cookie

**Example 2:** CacheIgnoreHeaders None

Do not store the given HTTP headers in the cache. According to RFC 2616, hop-by-hop HTTP headers are not stored in the cache. The following HTTP headers are hop-by-hop headers and thus do not get stored in the cache in any case regardless of the setting of CacheIgnoreHeaders:

- Connection
- Keep-Alive
- Proxy-Authenticate
- TE
- Trailers
- Transfer-Encoding
- Upgrade

CacheIgnoreHeaders specifies additional HTTP headers that should not be stored in the cache. For example, it makes sense in some cases to prevent cookies from being stored in the cache.

CacheIgnoreHeaders takes a space separated list of HTTP headers that should not be stored in the cache. If only hop-by-hop headers should not be stored in the cache (the RFC 2616 compliant behaviour), CacheIgnoreHeaders can be set to None.

**Warning:** If headers like Expires which are needed for proper cache management are not stored due to a CacheIgnoreHeaders setting, the behaviour of mod\_cache is undefined.

### CacheIgnoreNoLastMod:

**Module:** mod\_cache

**Syntax:** CacheIgnoreNoLastMod *on* | *off*

**Default:** CacheIgnoreNoLastMod off

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule cache\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** CacheIgnoreNoLastMod on

The CacheIgnoreNoLastMod directive specifies whether the server may cache HTTP proxy response data in the local proxy cache, if it does not contain a Last-Modified header or an ETag header.

**Parameter:** *on* | *off*

- If *off* is specified (the default), the server requires either an ETag header or a Last-Modified header to be present in all HTTP proxy response data cached in the local proxy cache.
- If *on* is specified, the server will not require an ETag header or Last-Modified header to be present in HTTP proxy response data cached in the local proxy cache.

By default, if data does not contain either an ETag header or a Last-Modified header, the server does not consider it cacheable. Specifying *on* for this setting waives this criteria. See Criteria for Local Proxy Cache for more information.

**Example One:**

```
ProxyRequests on
CacheRoot proxyCache
CacheIgnoreNoLastMod off
CacheDefaultExpire 1
```

In the example, if data is received from a server that does not provide an expiry time (via HTTP Expires header), nor does it have an ETag or Last-Modified header, it is not considered cacheable since *off* is specified for CacheIgnoreNoLastMod. The server serves the data for the current request, but does not cache it for subsequent requests. The settings for CacheDefaultExpire is not used.

**Example Two:**

```
ProxyRequests on
CacheRoot proxyCache
CacheIgnoreNoLastMod on
CacheDefaultExpire 1
```

In this example, if data is received from a server that does not provide an expiry time (via HTTP Expires header), nor does it have an ETag or Last-Modified header (as in example one), it is still considered cacheable since *on* is specified for CacheIgnoreNoLastMod. The server serves the data for the current request, and may calculate a cache expiry time using CacheDefaultExpire to cache it for subsequent requests, assuming it satisfies all other cache criteria.

**Note:** Response data is considered cacheable for the proxy function if it satisfies criteria described under Criteria for Local Proxy Cache.

- Setting ProxyRequests and ProxyReverse to off negates this directive.
- Setting ProxyNoConnect to on negates this directive.
- This directive is used only if CacheRoot is set.

**CacheLastModifiedFactor:**

**Module:** mod\_cache

**Syntax:** CacheLastModifiedFactor *factor*

**Default:** CacheLastModifiedFactor 0.1

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache



**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule cache\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** CacheLastModifiedFactor 0.3

The CacheLastModifiedFactor directive specifies a multiplication factor used in the formula:

```
period = time-since-last-modification *<factor>
```

**Parameter:** *factor*

- The *factor* parameter specifies the multiplication factor used in the formula (described above) to calculate cache expiry times.

This formula is used and setting is used along with CacheMaxExpire to calculate cache expiry times for HTTP proxy response data store in the local proxy cache, based on when the data was last modified. See Cache Expiry Times for more information on how the server determines which settings to use when calculating cache expiry times.

If this setting is used, cache expiry times are calculated by adding the lesser of the calculated period (using the formula above) and the period specified for CacheMaxExpire to the time that data is received by the proxy function. Using this method, data that has not changed recently is served from cache longer than data that has changed recently, since its last-modified time is older and will produce a greater cache expiry period. This assumes that both responses yield calculated cache expiry periods that are less than the CacheMaxExpire directive setting.

**Example:**

```
ProxyRequests on
CacheRoot proxyCache
CacheMaxExpire 86400
CacheLastModifiedFactor 0.3
```

In this example, if cacheable data is received from a server that does not provide an expiry time (via HTTP Expires header), but does indicate that the data was last changed 10 hours ago (via HTTP Last-Modified header, or FTP MDTM command), the server would calculate a period of 3 hours using CacheLastModifiedFactor (10 \* 0.3) and would cache and serve the data for the same period of time since it is less than the maximum limit of 24 hours set by CacheMaxExpire.

**Note:** Response data is considered cacheable for the proxy function if it satisfies criteria described under Criteria for Local Proxy Cache.

If a similar response for this example indicates that the data was last changed 8 days ago (or 192 hours), the server would calculate a period of 57.6 hours using CacheLastModifiedFactor (192 \* 0.3), but it would cache and serve the data for a period of only 24 hours since CacheMaxExpire sets a limit on the maximum period for the CacheLastModifiedFactor formula.

- Setting ProxyRequests and ProxyReverse to off negates this directive.
- Setting ProxyNoConnect to on negates this directive.
- This directive is used only if CacheRoot is set.

**CacheMaxExpire:**

**Module:** mod\_cache

**Syntax:** CacheMaxExpire *period*

**Default:** CacheMaxExpire 86400

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule cache\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** CacheMaxExpire 43200

The CacheMaxExpire directive specifies the maximum number of seconds in which cacheable HTTP proxy response data will be set to expire within the local proxy cache (when the CacheLastModifiedFactor directive setting is used). This setting has no affect on other settings used to calculate cache expiry times.

**Parameter:** *period*

- The *period* parameter specifies the maximum cache expiry period, in seconds, that may be used when expiry times are calculated using the CacheLastModifiedFactor directive setting.

This setting is used along with the CacheLastModifiedFactor directive setting to calculate expiry times for HTTP proxy response data stored in the local proxy cache, based on when data was last modified. See Cache Expiry Times for more information on how the server determines which settings to use when calculating cache expiry times. If this setting is used, cache expiry times are calculated by adding the lesser of the specified period and the period calculated using CacheLastModifiedFactor to the time that data is received by the proxy function.

**Example**

```
ProxyRequests on
CacheRoot proxyCache
CacheMaxExpire 86400
CacheLastModifiedFactor 0.3
```

In this example, if cacheable data is received from a server that does not provide an expiry time (via HTTP Expires header), but does indicate that the data was last changed 5 days ago (via HTTP Last-Modified header, or FTP MDTM command), the server would calculate a period of 1.5 days using CacheLastModifiedFactor (5 \* 0.3), but it would cache and serve the data for a period of only 86400 seconds (1 day) since CacheMaxExpire sets a maximum limit of 86400 seconds.

**Note:** Response data is considered cacheable for the proxy function if it satisfies criteria described under Criteria for Local Proxy Cache.

- 
- Setting ProxyRequests and ProxyReverse to off negates this directive.
- Setting ProxyNoConnect to on negates this directive.
- This directive is used only if CacheRoot is set

**CacheMaxFileSize:**

**Module:** mod\_cache

**Syntax:** CacheMaxFileSize *size*

**Default:** CacheMaxFileSize 1000000

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule cache\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM, disk\_cache\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** CacheMaxFileSize 4000000

The `CacheMaxFileSize` directive specifies the maximum amount of data that may be stored in the proxy disk cache for a single URL, in bytes. This setting effectively places a maximum data size limit on individual cache entries. If the disk cache function is disabled (see `CacheRoot`), this setting has no effect.

**Parameter:** *size*

- The *size* parameter specifies the maximum number of bytes allowed for cache data entries. A minimum document size limits specified using `CacheMinFileSize`.

**Notes for local proxy cache:**

When the disk cache function is used to support a local proxy cache, this setting places a maximum data size limit on HTTP proxy responses which remain in the cache after cache maintenance has run. See the `CachGcDaily` or `CacheGcInterval` directives for more information on how the disk cache maintenance function is used to support a local proxy cache.

**Example**

```
ProxyRequests on
CacheRoot proxyCache
CacheMaxFileSize 5000000
CacheMinFileSize 400000
```

For this example, if 7.2 megabytes of cacheable HTTP proxy response data is available for a single proxy request but will be removed during the next cache maintenance cycle since it is larger than the 5000000 byte maximum data size limit imposed by `CacheMaxFileSize`. A 3.8 megabyte HTTP proxy response may be cached for subsequent proxy requests and will remain in the cache after the cache maintenance cycle has run since it is smaller than the 5000000 byte maximum data size limit and larger than the 400000 byte minimum data size limit (set by `CacheMinFileSize`).

If the values specified for `CacheMinFileSize` and `CacheMaxFileSize` are changed once they have been used to cache data, the server will discard existing cache data that does not adhere to the new limits when it runs disk cache maintenance. See `CachGcDaily` or `CacheGcInterval` for more details on the disk cache maintenance process.

**CacheMinFileSize:**

**Module:** `mod_cache`

**Syntax:** `CacheMinFileSize size`

**Default:** `CacheMinFileSize 1`

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A `LoadModule` is required in the configuration file prior to using the directive. The statement should be as follows: `LoadModule cache_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM, LoadModule disk_cache_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM`

**Example:** `CacheMinFileSize 40`

The `CacheMinFileSize` directive specifies the minimum amount of data that may be stored in the proxy disk cache for a single URL, in bytes. This setting effectively places a minimum data size limit on individual cache entries. If the disk cache function is disabled (see `CacheRoot`), this setting has no effect.

**Parameter:** *size*

- The *size* parameter specifies the minimum number of bytes allowed for cache data entries. A maximum document size limits specified using `CacheMaxFileSize`.

**Notes for local proxy cache:**

When the disk cache function is used to support a local proxy cache, this setting places a minimum data size limit on HTTP proxy responses which remain in the cache after cache maintenance has run. See `CachGcDaily` and `CacheGcInterval` directives for more details on the how the disk cache maintenance function is used to support a local proxy cache.

### Example

```
ProxyRequests on
CacheRoot proxyCache
CacheMaxFileSize 5000000
CacheMinFileSize 400000
```

For this example, if 240 kilobytes of cacheable HTTP proxy response data is available for a single proxy request, but will be removed during the next cache maintenance cycle since it is less than the 400000 byte minimum data size limit imposed by `CacheMinFileSize`. A 2.7 megabyte HTTP proxy response may be cached for subsequent proxy requests and will remain in the cache after the cache maintenance cycle has run since it is larger than the 400000 byte minimum data size limit and smaller than the 5000000 byte maximum data size limit (set by `CacheMaxFileSize`).

If the values specified for `CacheMinFileSize` and `CacheMaxFileSize` are changed once they have been used to cache data, the server will discard existing cache data that does not adhere to the new limits when it runs disk cache maintenance. See `CachGcDaily` or `CacheGcInterval` for more details on the disk cache maintenance process.

- If set greater than `CacheMaxFileSize`, the size parameter will be ignored and the default value will be used.
- The default for this directive is dependent on the value set for `CacheMaxFileSize`. If `CacheMaxFileSize` is set greater than 200 the default for this directive is 200, otherwise the default is 0.

### CacheStoreNoStore:

**Module:** mod\_cache

**Syntax:** `CacheStoreNoStore on | off`

**Default:** `CacheStoreNoStore Off`

**Context:** server, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A `LoadModule` is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule cache_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

**Example:** `CacheStoreNoStore On`

Attempt to cache requests or responses that have been marked as no-store. Ordinarily, requests or responses with `Cache-Control: no-store` header values will not be stored in the cache. The `CacheStoreNoCache` directive allows this behavior to be overridden. `CacheStoreNoCache On` tells the server to attempt to cache the resource even if it contains no-store header values. Resources requiring authorization will never be cached.

### CAUTION:

As described in RFC 2616, the no-store directive is intended to "prevent the inadvertent release or retention of sensitive information (for example, on backup tapes)." Enabling this option could store sensitive information in the cache. You are hereby warned.

### CacheStorePrivate:

**Module:** mod\_cache

**Syntax:** `CacheStorePrivate on | off`

**Default:** CacheStorePrivate Off

**Context:** server, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule cache_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

Ordinarily, responses with Cache-Control: private header values will not be stored in the cache. The CacheStorePrivate directive allows this behavior to be overridden. CacheStorePrivate On tells the server to attempt to cache the resource even if it contains private header values. Resources requiring authorization will never be cached.

#### **CAUTION:**

**This directive will allow caching even if the upstream server has requested that the resource not be cached. This directive is only ideal for a private cache.**

#### **CacheTimeMargin:**

**Module:** mod\_cache

**Syntax:** CacheTimeMargin *period*

**Default:** CacheTimeMargin 120

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule cache_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

```
LoadModule disk_cache_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

**Example:** CacheTimeMargin 300

The CacheTimeMargin directive specifies the minimum number of seconds remaining prior to data expiration, as indicated in the expires response header, in order for data to be cached by the server using the disk cache function. If the disk cache function is disabled (see CacheRoot), this setting has no affect.

#### **Parameter:** *period*

- The *period* parameter specifies the minimum time margin for cache update requests (in seconds).

The server calculates cache time margins (or periods) for cache update requests by subtracting the current system time from the computed expiry time. Data for cache update requests that produce cache time margins, that are less than the specified minimum time margin is not cached by the server.

#### **Notes for local proxy cache:**

The disk cache function uses CacheDefaultExpire, CacheLastModifiedFactor, and CacheMaxExpire directives which may produce cache time margins that are less than the minimum time margin specified by the CacheTimeMargin directive. In this case, the CacheTimeMargin directive will also be used to determine if the file will be cached. See the CacheRoot directive for more information on how the disk cache function is used to support a local proxy cache.

#### **Example**

```
ProxyRequests on
CacheRoot proxyCache
CacheTimeMargin 120
```

In this example, if cacheable HTTP proxy response data is available, the data will be served (by proxy), but it will not be cached for subsequent requests if set to expire in less than 120 seconds (CacheTimeMargin 120). If the HTTP proxy response data is set to expire in more than two minutes, the data will be served (by proxy) and will also be cached for subsequent proxy requests.

## Module mod\_cgi


Module mod\_cgi supports directives for the IBM HTTP Server for i Web server.

### Summary

The module mod\_cgi provides for execution of CGI scripts. This module will process any file with mime type application/x-httpd-cgi. Any file that has the mime type application/x-httpd-cgi or handler cgi-script will be treated as a CGI script, and run by the server, with its output being returned to the client. Files acquire this type either by having a name containing an extension defined by the “AddType” on page 419 directive, or by being in a “ScriptAlias” on page 227 directory.

When the server invokes a CGI script, it will add a variable called DOCUMENT\_ROOT to the environment. This variable will contain the value of the “DocumentRoot” on page 293 configuration variable.

### CGI Environment variables

The server will set the CGI environment variables as described in the CGI specification  with the following provisions. See “Environment variables set by HTTP Server” on page 493 for a list of environment variables.

#### REMOTE\_HOST

This will only be set if “HostNameLookups” on page 301 is set to *double* (it is *off* by default), and if a reverse DNS lookup of the accessing hosts address indeed finds a host name.

#### REMOTE\_IDENT

This will only be set if “IdentityCheck” on page 304 is set to *on* and the accessing host supports the ident protocol.

**Note:** The contents of this variable cannot be relied upon because it can easily be faked. If there is a proxy between the client and the server, the variable is not useful.

#### REMOTE\_USER

This will only be set if the CGI script is subject to authentication.

### CGI Debug

Debugging CGI scripts has traditionally been difficult, mainly because it has not been possible to study the output (standard output and error) for scripts which are failing to run properly. However, the HTTP Server runs CGI programs in previously started jobs (not prestart jobs) and it also reuses these jobs to run many CGI program invocations. Therefore, debugging your CGI program is simple. You simply need to find the job that runs CGI programs. It will have a jobname the same as the server instance name. The joblog will contain either HTP2001 or HTP2002 indicating whether it is a CGI single threaded only job, or a CGI multi-thread capable job. If you use a dedicated server instance, when you invoke your CGI from a browser, the first job in the WRKACTJOB list for CGI, will be the job chosen to run the CGI request. Therefore, you can use STRSRVJOB against this job and STRDBG against your CGI program. From here, you have full debug capabilities provided with the IBM i debugger. You can also use standard error (stderr) for debug information. The debug information written to STDERR is written to the ScriptLog if one is configured or to the ErrorLog if a ScriptLog is not configured. The ScriptLog and ErrorLog are both created with CCSID 1208 UTF-8. For CGI conversion mode EBCDIC, debug information is assumed to be in the CCSID of the CGI job. The logging process handles the conversion from CGI job CCSID to UTF-8. For CGI conversion mode BINARY, debug information is written as is.

## ScriptLog Format

When configured, the ScriptLog logs any CGI that does not execute properly. Each CGI script that fails to operate causes several lines of information to be logged. The first two lines are always of the format:

```
%% [time] request-line %% HTTP-status CGI-script-filename
```

If the error is that CGI script cannot be run, the log file will contain an extra two lines:

```
%%error error-message
```

Alternatively, if the error is the result of the script returning incorrect header information (often due to a bug in the script), the following information is logged:

```
%request All HTTP request headers received POST or PUT entity (if any) %response  
All headers output by the CGI script %stdout CGI standard output %stderr CGI standard error
```

**Note:** The %stdout and %stderr parts may be missing if the script did not output anything on standard output or standard error

### Directives

- “CGIConvMode”
- “CgiInitialUrl” on page 275
- “CGIJobCCSID” on page 276
- “CGIJobLocale” on page 277
- “CGIMultiThreaded” on page 277
- “CGIRecyclePersist” on page 278
- “DefaultNetCCSID” on page 278
- “MaxCGIJobs” on page 279
- “MaxPersistentCGI” on page 279
- “MaxPersistentCGITimeout” on page 279
- “MaxThreadedCGIJobs” on page 280
- “PersistentCGITimeout” on page 280
- “ScriptLog” on page 281
- “ScriptLogBuffer” on page 281
- “ScriptLogLength” on page 282
- “StartCGI” on page 282
- “StartThreadedCGI” on page 283
- “ThreadedCgiInitialUrl” on page 283

### CGIConvMode:

**Module:** mod\_cgi

**Syntax:** CGIConvMode *mode*

**Default:** CGIConvMode EBCDIC

**Context:** server config, virtual host, directory, .htaccess, Not in Limit

**Override:** FileInfo

**Origin:** IBM

**Example:** CGIConvMode BINARY

The CGIConvMode directive is used to specify the conversion mode that your server will use when processing CGI programs.

**Parameter:** *mode*

- Valid modes include the following:

Table 17. Valid conversion modes

Mode	Description
EBCDIC	The server converts everything into the EBCDIC CCSID of the CGI job. If the directive CGIJobCCSID exists, it has precedence in its context over the job CCSID of the server. The server assumes the header output and encoded characters "%xx" are in the EBCDIC CCSID of the CGI job. The server assumes that the body output is in the EBCDIC CCSID of the CGI job unless specified otherwise using the Content-type header.
EBCDIC_JCD	The server will use the Japanese Codepage Detection utility to determine which Japanese ASCII CCSID to convert from. Otherwise, this option is the same as the EBCDIC option.
BINARY	The server performs no conversion on QUERY_STRING or STDIN data. Environment variables are encoded in the CGI job's EBCDIC CCSID.  The server expects the header output and encoded characters "%xx" in ASCII. The server assumes that the body output is in ASCII unless specified otherwise using the Content-type header. This differs from no parse header CGI in that the server will still build the HTTP headers and perform conversion on the body output if necessary.

- The following modes are used for compatibility with IBM HTTP Server (original).

Table 18. Legacy conversion modes

Mode	Description
%%MIXED/MIXED%%	The server converts CGI environment variables to EBCDIC CCSID 37, including QUERY_STRING. The server converts STDIN data to the CCSID of the server job. However, the encoded characters "%xx" are still represented by the EBCDIC 37 representation of the ASCII 819 octet. The server expects the header output to be in EBCDIC CCSID 37. However, the encoded characters "%xx" must be represented by the EBCDIC 37 representation of the ASCII 819 octet. The server assumes that the body output is in the default CCSID of the server job unless specified otherwise using the Content-type header. The header most affected by this value is the location header (for example, to send a plus sign '+' in the location header you would send %).
%%EBCDIC/MIXED%%	The server converts everything into the EBCDIC CCSID of the job. In addition, the server converts escaped octets from ASCII to EBCDIC. The server expects the header output to be in EBCDIC CCSID 37. However, the encoded characters "%xx" must be represented by the EBCDIC 37 representation of the ASCII 819 octet. The server assumes that the body output is in the default CCSID of the server job unless specified otherwise using the Content-type header. The header most affected by this value is the location header (for example, to send a plus sign '+' in the location header you would send %).
%%BINARY/MIXED%%	The server converts environment variables into the EBCDIC CCSID of the job, but performs no conversions on either QUERY_STRING or STDIN data. The server expects the header output to be in EBCDIC CCSID 37. However, the encoded characters "%xx" must be represented by the EBCDIC 37 representation of the ASCII 819 octet. The server assumes that the body output is in the default CCSID of the server job unless specified otherwise using the Content-type header. The header most affected by this value is the location header (for example, to send a plus sign '+' in the location header you would send %).
%%EBCDIC_JCD/MIXED%%	The server uses the Japanese Codepage Detection utility to determine which Japanese CCSID to convert from. Otherwise, this option is the same as the %%EBCDIC/MIXED%% option.



Table 18. Legacy conversion modes (continued)

Mode	Description
%%EBCDIC/EBCDIC%%	The server converts everything into the EBCDIC CCSID of the job. In addition, the server converts escaped octets from ASCII to EBCDIC. The server expects the header output and encoded characters "%xx" to be in EBCDIC CCSID 37. The server assumes that the body output is in the default CCSID of the server job unless specified otherwise using the Content-type header. The header most affected by this value is the location header (for example, to send a plus sign '+' in the location header you would send %).
%%BINARY/BINARY%%	The server converts environment variables into the EBCDIC CCSID of the job, but performs no conversions on either QUERY_STRING or STDIN data. The server expects the header output and encoded characters "%xx" to be in ASCII 819. The server assumes that the body output is in ASCII 819 unless specified otherwise using the Content-type header. The header most affected by this value is the location header (for example, to send a plus sign '+' in the location header you would send %).
%%BINARY/EBCDIC%%	The server converts environment variables into the EBCDIC CCSID of the job, but performs no conversions on either QUERY_STRING or STDIN data. The server expects the header output and the encoded characters "%xx" to be in EBCDIC CCSID 37. The server assumes that the body output is in the default CCSID of the server job unless specified otherwise using the Content-type header.
%%EBCDIC_JCD/EBCDIC%%	The server uses the Japanese Codepage Detection utility to determine which Japanese CCSID to convert from. Otherwise, this option is the same as the %%EBCDIC/EBCDIC%% option.

### CgiInitialUrl:

**Module:** mod\_cgi

**Syntax:** CgiInitialUrl *url userid*

**Default:** none

**Context:** server config

**Override:** none

**Origin:** IBM

**Example:** CgiInitialUrl /qsys.lib/qsyscgi.lib/db2www.pgm/mymacros/macro.ndm/initial \*

**Example:** CgiInitialUrl /qsys.lib/cgi.lib/mycgi.pgm QTMHHTTP1

**Example:** CgiInitialUrl /QOpenSys/mypacedir/paccgi USER1

**Example:** CgiInitialUrl /qsys.lib/cgi.lib/mycgi.pgm?init=yes

This directive is used to load and initialize CGI programs when the server starts. At server startup, when we are processing the StartCgi directive, we are starting jobs to run CGI programs in. This new directive will enable the server to run a CGI request to the CGI job enabling the CGI program to be loaded and initialized. This is beneficial for Net.Data users and other CGI programs built to use "named" activation groups. The initialization of the "named" activation group is a performance issue that the first user of the CGI job has to endure. This function will enable the performance issue to be moved to when the server starts, so the first user does not have to pay the performance penalty.

If there are no StartCgi directives, an error will be posted and the server will not start.

#### Parameter One: *url*

- The *url* parameter value is actually the physical path URL, not the logical path URL. It should not be fully qualified (do not use http://system:port/). It must start with a / and contains the physical path to the CGI program and any path info needed by the CGI program, including query-string. If a URL is specified that is not valid, the server will not start.

**Parameter Two:** *userid*

- The *userid* parameter value is either a valid IBM i userid or \* where \* means all of the userids specified on the StartCgi directive. To check for valid values, follow the rules for IBM i user profiles. The userid is optional.

**CGIJobCCSID:**

**Module:** mod\_cgi

**Syntax:** CGIJobCCSID *cgi-job-character-set-identification-number*

**Default:** CGIJobCCSID *Dependent upon server-character-set-identification-number*

**Context:** server config, virtual host, directory, not in limit, .htaccess

**Override:** none

**Origin:** IBM

**Example:** CGIJobCCSID 37

**Example:** To run one CGI program in CCSID 37 (English):

```
ScriptAlias /cgi-english/ /QSYS.LIB/ENGLISH.LIB/  
<Directory /QSYS.LIB/ENGLISH.LIB/>  
    Allow From all  
    Options +ExecCGI  
    DefaultNetCCSID 819  
    CGIJobCCSID 37  
    CGIConvMode EBCDIC  
</Directory>
```

**Example:** To run a different CGI program in CCSID 284 (Spanish):

```
ScriptAlias /cgi-spanish/ /QSYS.LIB/SPANISH.LIB/  
<Directory /QSYS.LIB/SPANISH.LIB/>  
    Allow From all  
    Options +ExecCGI  
    DefaultNetCCSID 819  
    CGIJobCCSID 284  
    CGIConvMode EBCDIC  
</Directory>
```

**Example:** For GET and POST – Use the URI to determine the language of the user:

Enter: <http://www.mydomain.com/cgi-bin/ENG/819/...>

The configuration file would have this container configuration:

```
<Location /cgi-bin/ENG/819/>  
    DefaultNetCCSID 819  
    CGIJobCCSID 37  
</Location>  
ScriptAlias /cgi-bin/ /QSYS.LIB/CGI.LIB/  
<Directory /QSYS.LIB/CGI.LIB/>  
    Allow From all  
    Options +ExecCGI  
    CGIConvMode EBCDIC  
</Directory>
```

The same configuration can handle this URI for a Japanese speaking user.

Enter: <http://www.mydomain.com/cgi-bin/JAP/942/>

The configuration file would also have this container configuration:

```
<Location /cgi-bin/JAP/942/>  
    DefaultNetCCSID 942  
    CGIJobCCSID 5035  
    CGIConvMode EBCDIC_JCD  
</Location>  
ScriptAlias /cgi-bin/ /QSYS.LIB/CGI.LIB/  
<Directory /QSYS.LIB/CGI.LIB/>  
    Allow From all  
    Options +ExecCGI  
    CGIConvMode EBCDIC  
</Directory>
```

The CGIJobCCSID directive specifies the CCSID under which CGI jobs run, the CGI job character set environment, and the EBCDIC CCSID that is used when the server converts:

- Input request data for user CGI programs
- Output response data from user CGI programs to be sent back to the requester (client browser)

If this directive is not specified, the default behavior is to have the CGI job run under the same CCSID as the main server jobs. See the DefaultFsCCSID directive for detailed information on how this is determined.

### CGIJobLocale:

**Module:** mod\_cgi

**Syntax:** CGIJobLocale *locale\_path\_name*

**Default:** none

**Context:** server config, virtual host, directory, not in limit, .htaccess

**Override:** none

**Origin:** IBM

**Example:** CGIJobLocale /QSYS.LIB/LOCALELIB.LIB/EN\_US.LOCALE

**Example:** To run one CGI program in CCSID 37 with an English based locale (English):

```
ScriptAlias /cgi-english/ /QSYS.LIB/ENGLISH.LIB/
```

```
<Directory /QSYS.LIB/ENGLISH.LIB/>
  Allow From all
  Options +ExecCGI
  DefaultNetCCSID 819
  CGIJobCCSID 37
      CGIJobLocale /QSYS.LIB/LOCALELIB.LIB/EN_US.LOCALE
  CGIConvMode EBCDIC
</Directory>
```

**Example:** To run a different CGI program in CCSID 273 and with a German based locale (German):

```
ScriptAlias /cgi-german/ /QSYS.LIB/GERMAN.LIB/
<Directory /QSYS.LIB/GERMAN.LIB/>
  Allow From all
  Options +ExecCGI
  DefaultNetCCSID 819
  CGIJobCCSID 273
      CGIJobLocale /QSYS.LIB/LOCALELIB.LIB/DE_DE.LOCALE
  CGIConvMode EBCDIC
</Directory>
```

Applications can be created independent of language, cultural data, or specific characters. Locales can be accessed to provide this type of support to any integrated language environment-based application. The CGIJobLocale directive allows a locale to be set globally or for a specific CGI job. After the locale is set, region specific information such as date or time format can be accessed. Some ILE C/C++ run time functions such as `ctime()` and `localtime()` are locale sensitive. The environment variable `CGI_JOB_LOCALE` is set from the CGIJobLocale directive.

### CGIMultiThreaded:

**Module:** mod\_cgi

**Syntax:** CGIMultiThreaded *on | off*

**Default:** CGIMultiThreaded off

**Context:** server config, virtual host, directory, .htaccess, Not in Limit

**Override:** FileInfo

**Origin:** IBM

**Example:** CGIMultiThreaded on

**Parameter:** *on | off*

- The *on* value indicates that your CGI programs will be run in a job that is multiple thread capable.
- The *off* value indicates that your CGI programs will not be run in a job that is multiple thread capable.

The `CGIMultiThreaded` directive is used to specify whether your CGI programs should be run in a job that is multiple thread capable. HTTP Server uses a pool of pre-started jobs for handling CGI requests. Multiple threaded programs must run in a multiple thread-capable job. The job pool that the job runs in is specified at job startup time. Once the job has started, it cannot be changed to another job pool. Not all IBM i APIs are thread safe, some will issue an error if used in a multiple thread-capable job. This happens even if the program does not actually have multiple threads running. Because of this, HTTP Server must default to non-multiple thread capable jobs for CGI programs for compatibility reasons. If your CGI program uses multiple threads, it must run in a multiple thread capable job. If your CGI does not need multiple threads, you should run it in the single threaded CGI job for performance reasons.

### **CGIRecyclePersist:**

**Module:** mod\_cgi

**Syntax:** CGIRecyclePersist *on* | *off*

**Default:** CGIRecyclePersist *off*

**Context:** server config, virtual host, directory, .htaccess, Not in Limit

**Override:** FileInfo

**Origin:** IBM

**Example:** CGIRecyclePersist *on*

The `CGIRecyclePersist` directive instructs the server what should be done with the job that was being used by a persistent CGI when the persistent CGI exits persistence normally.

**Parameter:** *on* | *off*

- The *on* value indicates that the server can reuse this job for other CGI requests. When this is used, the persistent CGI program is responsible for cleaning up any static data from the persistent CGI transaction. The server will not perform any action other than to remove all environment variables, to clean up any static data. Before using this setting, the CGI programmer need to verify that it does indeed clean up its static data.
- The *off* value indicates that the server will not reuse this job for other CGI requests. This is the default behavior.

### **DefaultNetCCSID:**

**Module:** mod\_cgi

**Syntax:** DefaultNetCCSID *client-character-set-identification-number*

**Default:** Global HTTP Server setting for coded character set identifier.

**Context:** server config, virtual host, directory, .htaccess

**Override:** none

**Origin:** IBM

**Example:** DefaultNetCCSID 819

The `DefaultNetCCSID` directive specifies the client character set environment and defines the ASCII or UTF-8 CCSID that is used when converting:

- Input request data for user CGI programs or Apache modules.
- When serving EBCDIC documents and no ASCII CCSID can be deduced from the file CCSID.
- Output response data from user CGI programs, or Apache modules, to be sent back to the requester (client browser).

A configuration file can contain more than one DefaultNetCCSID directive, but the last directive in the configuration file determines the CCSID. Starting in IBM i 5.4, the use of this directive is expanded to help you configure a single server to handle requests in more than one language. The directive is now allowed in a virtual host container and in directory containers. This directive is supported in the global scope. If the directive is not specified, the global HTTP Server setting for coded character set identifier is used. The shipped value is 00819 (ISO 8859-1 8-bit ASCII). You can view and change global HTTP Server settings using the Change HTTP Attributes (CHGHTTPA) command.

### **MaxCGIJobs:**

**Module:** mod\_cgi

**Syntax:** MaxCGIJobs *number*

**Default:** Value used for the ThreadsPerChild directive

**Context:** server config

**Override:** none

**Origin:** IBM

**Example:** MaxCGIJobs 50

The MaxCGI directive is used to set the maximum number of CGI jobs that the server will concurrently use. The server will only run CGI programs in jobs where the user profile for the CGI job matches the user profile that the request is to run under. If you protect your CGI programs with many different dummy IBM i profiles (profiles with no password) or use %%CLIENT%% (each user has their own IBM i profile and it is used to run the CGI program), then you may want to use this directive to allow the server to start more CGI jobs to handle the CGI programs. The server does reuse the CGI jobs, but only when the profile for the CGI program matches the profile for the CGI job. If you see the server ending and starting CGI jobs regularly, then you may want to use this directive to allow the server to use more CGI jobs. This would improve the capacity and performance of your system and server.

**Parameter:** *number*

- The *number* parameter accepts any positive number. If an invalid value is used, or the number is smaller than the value used for the ThreadsPerChild directive, then the server will use the value used for the ThreadsPerChild directive.

### **MaxPersistentCGI:**

**Module:** mod\_cgi

**Syntax:** MaxPersistentCGI *number*

**Default:** Value used for the ThreadsPerChild directive

**Context:** server config

**Override:** none

**Origin:** IBM

**Example:** MaxPersistentCGI 50

The MaxPersistentCGI directive is used to set the maximum number of active persistent CGI jobs that you want to have active at one time.

**Parameter:** *number*

- The *number* parameter sets the maximum number of active persistent CGI jobs that are active at any one time.

### **MaxPersistentCGITimeout:**

**Module:** mod\_cgi

**Syntax:** MaxPersistentCGITimeout *number*

**Default:** MaxPersistentCGITimeout 1200

**Context:** server config  
**Override:** none  
**Origin:** IBM  
**Example:** MaxPersistentCGITimeout 1800

The MaxPersistentCGITimeout directive specifies the maximum number of seconds that a CGI program can use when overriding the PersistentCGITimeout directive.

**Parameter:** *number*

- The *number* parameter value must be greater than 1 second.

### MaxThreadedCGIJobs:

**Module:** mod\_cgi  
**Syntax:** MaxThreadedCGIJobs *number*  
**Default:** Value used for the ThreadsPerChild directive  
**Context:** server config  
**Override:** none  
**Origin:** IBM  
**Example:** MaxThreadedCGIJobs 50

The MaxThreadedCGIJobs directive is used to set the maximum number of multiple thread capable CGI jobs that the server will concurrently use. The server will only run multiple thread capable CGI programs in jobs where the user profile for the multiple thread capable CGI job matches the user profile that the request is to run under. If you protect your multiple thread capable CGI programs with many different dummy IBM i profiles (profiles with no password) or use %%CLIENT%% (each user has their own IBM i profile and it is used to run the multiple thread capable CGI program), then you may want to use this directive to allow the server to start more multiple thread capable CGI jobs to handle the multiple thread capable CGI programs. The server does reuse the CGI jobs, but only when the profile for the multiple thread capable CGI program matches the profile for the multiple thread capable CGI job. If you see the server ending and starting multiple thread capable CGI jobs regularly, then you may want to use this directive to allow the server to use more multiple thread capable CGI jobs. This would improve the capacity and performance of your system and server.

**Parameter:** *number*

- The *number* parameter value can be any positive number. If an invalid value is used, or the number is smaller than the value used for the ThreadsPerChild directive, then the server will use the value used for the ThreadsPerChild directive.

### PersistentCGITimeout:

**Module:** mod\_cgi  
**Syntax:** PersistentCGITimeout *number*  
**Default:** PersistentCGITimeout 300  
**Context:** server config  
**Override:** none  
**Origin:** IBM  
**Example:** PersistentCGITimeout 120

This directive specifies the number of seconds that your server waits for a client response before ending a persistent CGI session. The CGI program can override the value that you specify on a request-by-request basis.

**Parameter:** *number*

- The *number* parameter can be any amount of time greater than 1 second.

## **ScriptLog:**

**Module:** mod\_cgi

**Syntax:** ScriptLog *filename*

**Default:** none

**Context:** server config

**Override:** none

**Origin:** Modified

**Example:** ScriptLog /QIBM/userdata/httpa/(instance name)

The ScriptLog directive sets the Common Gateway Interface (CGI) script error logfile. If no ScriptLog is given, no CGI error log is created. If a ScriptLog is given, any CGI errors are logged into the filename given as the argument. If this is a relative file or path, it is taken relative to the server root.

This log will be opened as the user the child processes run as, for example the user specified in the main User directive. This means that either the directory the script log is in needs to be writable by that user or the file needs to be manually created and set to be writable by that user. If you place the script log in your main logs directory, do not change the directory permissions to make it writable by the user the child processes run as.

**Note:** The script logging is meant to be a debugging feature when writing CGI scripts, and is not meant to be activated continuously on running servers. It is not optimized for speed or efficiency, and may have security problems if used in a manner other than that for which it was designed.

## **Behavior**

If the filename does not begin with a slash ('/') then it is assumed to be relative to the ServerRoot.

If the path ends with a '/' character, then the path is considered to be the directory that will contain the log file.

The ScriptLog file will be created with CCSID 1208 (UTF8). Customer data written to the script log is assumed to be in the CGI job CCSID and will automatically be converted to CCSID 1208. The data will be written to the log file in binary. Therefore, the customer's data will be written to the ScriptLog without conversion. Information from the CGI request will not need to be translated, as the data will already be in the defaultFSCSID.

## **ScriptLogBuffer:**

**Module:** mod\_cgi

**Syntax:** ScriptLogBuffer *size*

**Default:** ScriptLogBuffer 1024

**Context:** server config

**Override:** none

**Origin:** Apache

**Example:** ScriptLogBuffer 512

The ScriptLogBuffer directive limits the size of any PUT or POST entity body that is logged to the file. This prevents the log file from growing too big too quickly (the case if large bodies are being received).

**Parameter:** *size*

- The *size* parameter is measured in bytes and consists of any positive integer. By default, up to 1024 bytes are logged, but the value can be changed with this directive.

## ScriptLogLength:

**Module:** mod\_cgi

**Syntax:** ScriptLogLength *size*

**Default:** ScriptLogLength 10385760

**Context:** server config

**Override:** none

**Origin:** IBM

**Example:** ScriptLogLength 1024000

The ScriptLogLength directive can be used to limit the size in bytes of the Common Gateway Interface (CGI) script log file. Since the log file logs a significant amount of information per CGI error (all request headers, all script output) it can grow to be quite large. To prevent problems due to unbounded growth, this directive can be used to set a maximum file-size for the CGI logfile. If the file exceeds this size, no more information will be written to it.

**Parameter:** *size*

- The *size* parameter is measured in bytes. This is any positive number.

## StartCGI:

**Module:** mod\_cgi

**Syntax:** StartCGI *number userid*

**Default:** none

**Context:** server config

**Override:** none

**Origin:** IBM

**Example:** StartCGI 5 USER1

The StartCGI directive specifies the number of CGI jobs that are spawned by the server when it starts up and the IBM i user profile to use in these jobs. This allows you to have the server prestart CGI jobs when the server starts so the users do not incur the performance hit of starting a new job. It also allows you to start up jobs for different user profiles. The userid is optional and should only be used to protect your CGI programs so that they run under the %%CLIENT%% profile or under a dummy IBM i profile (a profile with no password).

The cumulative number from all occurrences of this directive cannot exceed MaxCGIJobs, if it does, the server will not start. If the user profile parameter is not specified, the default server profile (QTMHHTTP1) or the value from the global ServerUserID directive is used.

If you are using %%CLIENT%% as the profile in the protection of the CGI programs (meaning that each user authenticates with an IBM i user profile), then it should be noted that %%CLIENT%% is not a valid value on this directive. Using IBM i profiles like this should only be done in an intranet or highly secure server because you would not want to give just anyone an IBM i user profile. Therefore, you would know how many users and also their user profile name, thus you would need to decide how many users will be doing CGI requests and how many concurrent CGI requests you want each user to be able to do. Then you could specify multiple StartCGI directives, one for each user, specifying the number of concurrent CGI requests you expect that user to do.

**Note:** This will NOT limit the number of concurrent CGI requests. This will simply allow CGI jobs to be started at server startup time so the user does not have to incur the performance hit of starting up a new job when they run their first CGI program.



## StartThreadedCGI:

**Module:** mod\_cgi

**Syntax:** StartThreadedCGI *number userid*

**Default:** none

**Context:** server config

**Override:** none

**Origin:** IBM

**Example:** StartThreadedCGI 3

**Example:** StartThreadedCGI 5 USER1

The Start ThreadedCGI directive specifies the number of multiple thread capable CGI jobs that are spawned by the server when it starts up and the IBM i user profile to use in these jobs. This allows you to have the server prestart CGI jobs when the server starts so the users do not incur the performance hit of starting a new job. It also allows you to start up jobs for different user profiles. The userid is optional and should only be used to protect your multiple thread capable CGI programs so that they run under the `%%CLIENT%%` profile or under a dummy IBM i profile (a profile with no password).

The cumulative number from all occurrences of this directive cannot exceed `MaxThreadedCGIJobs`, if it does, the server will not start. If the user profile parameter is not specified, the default server profile (`QTMHHTTP1`) or the value from the global `ServerUserID` directive is used.

If you are using `%%CLIENT%%` as the profile in the protection of the multiple thread capable CGI programs (meaning that each user authenticates with an IBM i user profile), then it should be noted that `%%CLIENT%%` is not a valid value on this directive. Using IBM i profiles like this should only be done in an intranet or highly secure server because you would not want to give just anyone an IBM i user profile. Therefore, you would know how many users and also their user profile name, thus you would need to decide how many users will be doing CGI requests and how many concurrent multiple thread capable CGI requests you want each user to be able to do. Then you could specify multiple `StartThreadedCGI` directives, one for each user, specifying the number of concurrent multiple thread capable CGI requests you expect that user to do.

**Note:** This will NOT limit the number of concurrent multiple thread capable CGI requests. This will simply allow multiple thread capable CGI jobs to be started at server startup time so the user does not have to incur the performance hit of starting up a new job when they run their first multiple thread capable CGI program.

## ThreadedCgiInitialUrl:

**Module:** mod\_cgi

**Syntax:** ThreadedCgiInitialUrl *url userid*

**Default:** none

**Context:** server

**Override:** none

**Origin:** IBM

**Example:** ThreadedCgiInitialUrl /qsys.lib/cgi.lib/mycgi.pgm QTMHHTTP

**Example:** ThreadedCgiInitialUrl /QOpenSys/mypacedir/pac cgi

**Example:** ThreadedCgiInitialUrl /qsys.lib/cgi.lib/mycgi.pgm?init=yes USER1

This directive is used to load and initialize threaded CGI programs when the server starts. At server startup, when processing the `StartThreadedCgi` directive, jobs are started to run CGI programs in. This directive enables the server to run a CGI request to the CGI job enabling the CGI program to be loaded and initialized. This function enables performance issues to be moved to when the server starts, so the first user does not have diminished performance.

If there are no StartThreadedCgi directives, an error is posted and the server does not start.

## Module core

Module mod\_core supports directives for the IBM HTTP Server for i Web server.

### Summary

These directives control the core function of HTTP Server.

### Directives

- "AcceptPathInfo" on page 285
- "AcceptThreads" on page 286
- "AccessFileName" on page 287
- "AddDefaultCharset" on page 287
- "AddOutputFilterByType" on page 288
- "AllowEncodedSlashes" on page 288
- "AllowOverride" on page 289
- "AuthName" on page 289
- "AuthType" on page 290
- "DefaultType" on page 291
- "<Directory>" on page 291
- "<DirectoryMatch>" on page 293
- "DocumentRoot" on page 293
- "EnableSendfile" on page 294
- "ErrorDocument" on page 294
- "ErrorLog" on page 296
- "ErrorLogFormatDDS" on page 298
- "FileETag" on page 298
- "<Files>" on page 299
- "<FilesMatch>" on page 300
- "ForceType" on page 300
- "ForensicLog" on page 300
- "HostNameLookups" on page 301
- "HotBackup" on page 301
- "HTTPSubsystemDesc" on page 302
- "HTTPStartJobQueue" on page 303
- "HTTPStartJobDesc" on page 303
- "HTTPRoutingData" on page 303
- "IdentityCheck" on page 304
- "<IfDefine>" on page 304
- "<IfModule>" on page 305
- "Include" on page 306
- "KeepAlive" on page 306
- "KeepAliveTimeout" on page 306
- "<Limit>" on page 307
- "LimitInternalRecursion" on page 308
- "<LimitExcept>" on page 307

- “LimitRequestBody” on page 308
- “LimitRequestFields” on page 309
- “LimitRequestFieldsize” on page 309
- “LimitRequestLine” on page 310
- “LimitXMLRequestBody” on page 310
- “Listen” on page 310
- “ListenBacklog” on page 311
- “<Location>” on page 312
- “<LocationMatch>” on page 313
- “LogCycle” on page 313
- “LogLevel” on page 315
- “LogLength” on page 315
- “LogLevel” on page 315
- “LogMaint” on page 316
- “LogMaintHour” on page 317
- “LogTime” on page 317
- “MaxKeepAliveRequests” on page 318
- “NameVirtualHost” on page 318
- “Options” on page 319
- “ProfileToken” on page 320
- “ReceiveBufferSize” on page 320
- “Require” on page 321
- “RuleCaseSense” on page 322
- “Satisfy” on page 322
- “SendBufferSize” on page 323
- “SendFileMinSize” on page 323
- “ServerAdmin” on page 324
- “ServerAlias” on page 324
- “ServerName” on page 325
- “ServerPath” on page 325
- “ServerRoot” on page 326
- “ServerSignature” on page 326
- “ServerTokens” on page 327
- “ServerUserID” on page 328
- “SetHandler” on page 328
- “SetInputFilter” on page 329
- “SetOutputFilter” on page 329
- “ThreadsPerChild” on page 330
- “TimeOut” on page 330
- “TraceEnable” on page 331
- “UseCanonicalName” on page 331
- “UseShutdown” on page 332
- “<VirtualHost>” on page 332

**AcceptPathInfo:**

**Module:** core

**Syntax:** AcceptPathInfo *On* | *Off* | *Default*  
**Default:** AcceptPathInfo *Default*  
**Context:** server config, virtual host, directory, .htaccess  
**Override:** none  
**Origin:** Apache  
**Example:** AcceptPathInfo *On*

The AcceptPathInfo directive controls whether requests that contain trailing pathname information, that follows an actual filename or nonexistent file in an existing directory, are accepted or rejected. The trailing pathname information can be made available to scripts in the PATH\_INFO environment variable.

For example, assume the location /test/ points to a directory that contains only the single file here.html. Requests for /test/here.html/more and /test/nothere.html/more both collect /more as PATH\_INFO.

**Parameter:** *On* | *Off* | *Default*

- When set to *On*, a request will be accepted if a leading path component maps to a file that exists. The above example /test/here.html/more will be accepted if /test/here.html maps to a valid file.
- When set to *Off*, a request will only be accepted if it maps to a literal path that exists. Therefore a request with trailing pathname information after the true filename such as /test/here.html/more in the above example will return a 404 NOT FOUND error.
- When set to *Default*, the treatment of requests with trailing pathname information is determined by the handler responsible for the request. The core handler for normal files defaults to rejecting PATH\_INFO. Handlers that serve scripts, such as cgi-script and isapi-isa, generally accept PATH\_INFO by default.

The primary purpose of the AcceptPathInfo directive is to allow you to override the handler's choice of accepting or rejecting PATH\_INFO. This override is required, for example, when you use a filter (such as INCLUDES) to generate content based on PATH\_INFO. The core handler would usually reject the request. You can use the following configuration to enable such a script:

```
<Files "mypaths.shtml">  
Options +Includes  
SetOutputFilter INCLUDES  
AcceptPathInfo on  
</Files>
```

### AcceptThreads:

**Module:** core  
**Syntax:** AcceptThreads *number*  
**Default:** AcceptThreads 4  
**Context:** server config  
**Override:** none  
**Origin:** IBM  
**Example:** AcceptThreads 5

The AcceptThreads directive specifies the maximum number of accept threads per server child process. If a value is not specified, the server will use a limit of four accept threads. The accept threads are used to accept new connections from the client. This number may need to be changed to reflect the number of concurrent connections which are being accepted. If a large number of connections to the Web server start at approximately the same time, the number of accept threads may need to be adjusted to a higher value.

**Note:** The accept threads are created one time, and that is at startup time.

**Parameter:** *number*

- The *number* value specifies the maximum number of accept threads per server child process. Valid values include 1 through 20.

### AccessFileName:

**Module:** core

**Syntax:** AccessFileName *filename* [*filename ...*]

**Default:** AccessFileName .htaccess

**Context:** server config, virtual host, Not in Limit

**Override:** none

**Origin:** Apache

**Example:** AccessFileName index.html

When returning a document to the client, the server looks for the first access control file in the list of names in every document directory path. This only happens if the access control files are enabled for the directory. For example:

```
AccessFileName .acl
```

Before returning the document `/QIBM/UserData/web/index.html`, the server will read `/.acl`, `/QIBM/.acl`, `/QIBM/UserData/.acl` and `/QIBM/UserData/web/.acl` for directives, unless they have been disabled with the following:

```
<Directory/>
  AllowOverride None
</Directory>
```

**Parameter:** *filename*

- *Filename* is any valid filename on the IBM i server.

If multiple occurrences of this directive are configured in a container, only the last occurrence is processed. All other occurrences are ignored.

See also “AllowOverride” on page 289.

### AddDefaultCharset:

**Module:** core

**Syntax:** AddDefaultCharset *on* | *off* | *charset*

**Default:** AddDefaultCharset off

**Context:** server config, virtual host, directory, .htaccess, Not in Limit

**Override:** FileInfo

**Origin:** IBM

**Example:** AddDefaultCharset off

The AddDefaultCharset directive specifies the character set name that will be added to any response that does not have a parameter on the content type in the HTTP headers. This will override any character set specified, in the document body, by a META tag.

**Parameter:** *on* | *off* | *charset*

- AddDefaultCharset *on* enables HTTP Server's internal default charset of iso-8859-1 as required by the directive.
- AddDefaultCharset *off* disables this functionality.
- Alternate *charset* can be specified, for example, AddDefaultCharset on utf-8.

## AddOutputFilterByType:

**Module:** core

**Syntax:** AddOutputFilterByType *filtername content-type*

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** FileInfo

**Origin:** Apache

**Example:** AddOutputFilterByType INCLUDES text/html

**Example:** AddOutputFilterByType INCLUDES;DEFLATE text/html

The AddOutputFilterByType directive matches the MIME content-type of files to a filter which will process responses from the server before they are sent to the client. All files of the given *content-type* will be processed through the filter *filtername*. This is in addition to all defined filters, including those defined in the “SetOutputFilter” on page 329 directive.

**Parameter One:** *filtername*

- The name of a filter which will process responses from the server before they are sent to the client.

**Parameter Two:** *content-type*

- Any valid MIME-type.

To process the the content by more than one filter, the names have to be separated by semicolons. It is also possible to use one AddOutputFilterByType directive for each of these filters. The configuration below causes all script output labeled as text/html to be processed first by the INCLUDES filter and then by the DEFLATE filter.

```
<Location /cgi-bin/>
Options Includes
AddOutputFilterByType INCLUDES;DEFLATE text/html
</Location>
```

## AllowEncodedSlashes:

**Module:** core

**Syntax:** AllowEncodedSlashes *on* | *off*

**Default:** AllowEncodedSlashes off

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Example:** AllowEncodedSlashes on

The AllowEncodedSlashes directive allows URLs which contain encoded path separators (%2F for / and additionally %5C for \ on according systems) to be used. Normally, such URLs are refused with a 404 (Not found) error. Turning AllowEncodedSlashes *on* is useful when used in conjunction with PATH\_INFO environment variable.

**Note:** Allowing encoded slashes does not imply decoding. Occurrences of %2F or %5C (only on according systems) will be left as such in the otherwise decoded URL string.

**Parameter:** *on* | *off*

- The *on* parameter value specifies that URLs with encoded path separators can be used.
- The *off* parameter value specifies that URLs with encoded path separators will result in a 404 (Not found) error.

## AllowOverride:

**Module:** core

**Syntax:** AllowOverride *override* [*override* ..]

**Default:** AllowOverride none

**Context:** directory, Not in Limit

**Override:** none

**Origin:** Modified

**Example:** AllowOverride all

When the server finds an .htaccess file (as specified by "AccessFileName" on page 287) it needs to know which directives declared in that file can override earlier access information.

**Parameter:** *override*

- *Override* can be set to one or more of the following

Override	Description
None	The server will not read the file.
All	The server will allow all directives.
AuthConfig	Allow use of the authorization directives such as AuthName, AuthType, "PasswdFile" on page 233..
FileInfo	Allow use of the directives controlling document types such as AddEncoding, AddLanguage, AddType, "DefaultType" on page 291, "ErrorDocument" on page 294 and LanguagePriority.
Indexes	Allow use of the directives controlling directory indexing such as AddDescription, AddIcon, AddIconByEncoding, AddIconByType, DefaultIcon, DirectoryIndex, IndexOptions, HeaderName, IndexIgnore, IndexOptions and ReadmeName.
Limit	Allow use of the directives controlling host access such as "Allow" on page 219, "Deny" on page 219 and "Order" on page 220.
Options	Allow use of the directives controlling specific directory features such as "Options" on page 319.

**Note:** The use of .htaccess is not supported in QDLS and QSYS. For these file systems the override value must be *None* to avoid errors that keep a Web page from being served.

## AuthName:

**Module:** core

**Syntax:** AuthName *auth-domain*

**Default:** none

**Context:** directory, .htaccess

**Override:** AuthConfig

**Origin:** Modified

**Example:** AuthName "IBM Server"

The AuthName directive sets the name of the authorization realm for a directory. This realm is given to the client during basic authentication to inform the user about which username and password to send. To work properly this directive must be accompanied by "AuthType" on page 290 *Basic*, and directives such as "PasswdFile" on page 233.

**Parameter:** *auth-domain*

- The *auth-domain* parameter values specifies a single argument; if the realm name contains spaces, it must be enclosed in double quotation marks.

## AuthType:

**Module:** core

**Syntax:** AuthType *type*

**Default:** none

**Context:** directory, .htaccess

**Override:** AuthConfig

**Origin:** Modified

**Example:** AuthType Basic

**Example:** AuthType SSL

**Example:** AuthType Kerberos

**Example:** AuthType KerberosOrBasic

The AuthType directive selects the type of user authentication for a directory. For *Basic* authentication to work properly this directive must be accompanied by "AuthName" on page 289. If *Kerberos* is specified, the Require directive must be specified and the PasswdFile directive should be included and set to %%KERBEROS%%. The AuthName, LDAPConfigFile, and LDAPRequire directives may be configured in the same container, but will be ignored.

**Parameter:** *type*

- The *type* parameter value specifies the type of user authentication for a directory. Valid values include:

**Basic** Configuring "AuthType *Basic*" specifies that the server protects resources based on a user ID and password. The user will be prompted for a user ID and password the first time a request is made for a resource protected by this directive. This directive may be used on either a secure or a non-secure HTTP session. On a non-secure HTTP session, the user ID and password are encoded, but not encrypted.

**Note:** Note: In order to use the directive "SSLAuthType CertOrBasic", the AuthType directive must be specified with a value of type *Basic*.

**SSL** Configuring "AuthType *SSL*" specifies that the server will protect resources based on a SSL client certificate that is associated with a user ID. See the SSLAuthType directive for more information.

**Note:** In order to use the directive "SSLAuthType Cert", the AuthType directive must be specified with a value of type *SSL*.

### Kerberos

Configuring "AuthType *Kerberos*" specifies that the server will accept a server ticket from a Kerberos-enabled client to authenticate a user.

### KerberosOrBasic

Configuring "AuthType *KerberosOrBasic*" specifies that the server will give a basic authentication prompt to those browsers who are either not in a kerberos enabled domain, not using Microsoft Internet Explorer, or if kerberos authentication fails for a Microsoft Internet Explorer browser in a kerberos realm. If the browser is Microsoft Internet Explorer configured for kerberos, and in a kerberos domain with the correct kerberos principal and keytab entries, there will be no prompt (uses kerberos HTTP negotiation). To work correctly the intersection of directives for "Kerberos" and "Basic" authority must be used. Kerberos specific directives will not work, because basic authentication can not use kerberos validation. These directives are required when using KerberosOrBasic:

- "AuthName" on page 289
- PasswdFile %%SYSTEM%%



- Require: The parameter *valid-user*, *user* or *group* may be specified. For example:  
Require user *kerbuser@DOMAIN.COM as400userid*

**Notes:**

- The group file must include both the kerberos principal and the as400userid. For example Groupfile: productionusers: johndoe@WIN2003.DOMAIN.COM, jdoe
- If you do not use the *valid-user* you must include both the kerberos client principal and the as400 userid to which it maps.

If you want to have SSL certificate checking, it is recommended that AuthType be set to type *SSL*.

**DefaultType:**

**Module:** core

**Syntax:** DefaultType *MIME-type*

**Default:** DefaultType text/plain

**Context:** server config, virtual host, directory, .htaccess

**Override:** FileInfo

**Origin:** Apache

**Example:** DefaultType image/gif

There will be times when the server is asked to provide a document whose type cannot be determined by its MIME types mappings. The server must inform the client of the document content-type, so in the event of an unknown type it uses the DefaultType.

**Parameter:** *MIME-type*

- The *MIME-type* value specifies the document content-type.

For example:

DefaultType image/gif

This would be appropriate for a directory which contained many gif images with filenames missing the .gif extension. This would also be useful for documents in the QSYS.LIB file system, so you do not have to set the member type field.

**<Directory>:**

**Module:** core

**Syntax:** <Directory *directory*> ... </Directory>

**Default:** none

**Context:** server config, virtual host, Not in Limit

**Override:** none

**Origin:** Apache

**Example:** <Directory /usr/local/httpd/htdocs>

<Directory> and </Directory> are used to enclose a group of directives that only apply to the named directory and subdirectories of that directory. Any directive that is allowed in a directory context may be used.

**Parameter:** *directory*

- A *directory* is either the full path to a directory or a wildcard string. Refer to “<DirectoryMatch>” on page 293 for details regarding wildcard strings. Full path directory example:

```

<Directory /usr/local/httpd/htdocs>
Options Indexes
FollowSymLinks
</Directory>

```

If multiple (non-regular expression) directory sections match the directory (or its parents) containing a document, then the directives are applied in the order of shortest match first, interspersed with the directives from the .htaccess files. See “AccessFileName” on page 287 for more information. For example:

```

<Directory />
    AllowOverride None
</Directory>

<Directory /home/*>
    AllowOverride FileInfo
</Directory>

```

For access to the document /home/web/dir/doc.html the steps are:

- Apply directive AllowOverride None (disabling .htaccess files).
- Apply directive AllowOverride FileInfo (for directory /home/web).
- Apply any FileInfo directives in /home/web/.htaccess.

Regular expressions are not considered until all of the normal sections have been applied. Then all of the regular expressions are tested in the order they appeared in the configuration file. For example:

```

<Directory ~ abc$>
... directives here ..
</Directory>

```

Suppose that the filename being accessed is /home/ABC/public\_html/ABC/index.html. The server considers each of /, /home, /home/ABC, /home/ABC/public\_html and /home/ABC/public\_html/ABC in that order. The regular expression would not be considered until all normal <Directory> and .htaccess files have been applied. Then the regular expression will match on /home/ABC/public\_html/ABC and be applied.

#### Notes:

- The default HTTP Server access for <Directory /> is Allow from All. This means that HTTP Server will serve any file mapped from a URL. The GUI directory wizard automatically creates a root directory that denies access to all and doesn't allow htaccess file usage:

```

<Directory />
    Options None
    AllowOverride None
    order deny,allow
    deny from all
</Directory>

```

Then override this for directories you want accessible. See the “Security tips for HTTP Server” on page 29 or “User profiles and required authorities for HTTP Server” on page 30 pages for more details. <Directory> directives can only be in virtual host and the server configuration see context above. Previously, <Directory> containers were used to enclose groups of directives that applied to proxy requests by appending the prefix “proxy:” to the beginning of the specified directory name. This is no longer supported. The server now has proxy containers for this purpose. The proxy now ignores directives enclosed in directory (or file) containers, and uses proxy containers. See <Proxy> and <ProxyMatch> for more information.

- Directives within location containers (if matched) take precedence over directives within directory containers. See “<Location>” on page 312 and “<LocationMatch>” on page 313 directives for more information on location containers.

## <DirectoryMatch>:

**Module:** core

**Syntax:** <DirectoryMatch *regex*> ... </DirectoryMatch>

**Default:** none

**Context:** server config, virtual host, Not in Limit

**Override:** none

**Origin:** Apache

**Example:** <DirectoryMatch "^/www/.\*/[0-9]{3}">

<DirectoryMatch> and </DirectoryMatch> are used to enclose a group of directives that only apply to the named directory and subdirectories of that directory. It is the same as <Directory>; however, it takes an argument as a regular expression. For example:

```
<DirectoryMatch "^/www/.*/[0-9]{3}">
```

This would match directories in /www/ that consisted of three numbers.

**Note:** The argument to DirectoryMatch does not need to be in quotes unless the regular expression includes a space character.

**Parameter:** *regex*

- *Regex* is a UNIX-style regular expression that is matched against the URL. Subexpressions are grouped within parentheses. Then, parenthetically enclosed regular expressions will be substituted in a subsequent \$*n* statement.

See also "<Directory>" on page 291.

## DocumentRoot:

**Module:** core

**Syntax:** DocumentRoot *directory-path*

**Default:** DocumentRoot /QIBM/UserData/HTTPPA/htdocs

**Context:** server config, virtual host, Not in Limit

**Override:** none

**Origin:** Apache

**Example:** DocumentRoot /QIBM/UserData/mydocs

The DocumentRoot directive sets the directory from which HTTP Server will serve files. If the URL is not matched by a directive like Alias, the server appends the path from the requested URL to the document root and makes the path to the document.

**Parameter:** *directory-path*

- *Directory-path* is any valid directory path on the IBM i server.

For example:

```
DocumentRoot /usr/web
```

An access to <http://www.my.host.com/index.html> refers to /usr/web/index.html.

If the DocumentRoot directive is used in the server context and the directory does not exist, the server will not start. If the DocumentRoot directive is used in a virtual host context and the directory does not exist, that virtual host will inherit the document root from the server context (the server will start).

## EnableSendfile:

**Module:** core

**Syntax:** EnableSendfile *on|off*

**Default:** EnableSendfile on

**Context:** server, virtual host, directory, .htaccess

**Override:** FileInfo

**Origin:** Apache

**Example:** EnableSendfile On

This directive controls whether httpd may use the sendfile support from the kernel to transmit file contents to the client. By default, when the handling of a request requires no access to the data within a file (for example, when delivering a static file) Apache uses sendfile to deliver the file contents without ever reading the file if the operating system supports it. This sendfile mechanism avoids separate read and send operations, and buffer allocations.

## ErrorDocument:

**Module:** core

**Syntax:** ErrorDocument *error-code document*

**Default:** none

**Context:** server config, virtual host, directory, .htaccess, Not in Limit

**Override:** FileInfo

**Origin:** Modified

**Example:** ErrorDocument 404 /cgi-bin/bad\_urls.html

**Example:** ErrorDocument 500 http://QIBM.example.com/cgi-bin/tester

**Example:** ErrorDocument 404 /cgi-bin/bad\_urls.html

**Example:** ErrorDocument 401 /subscription\_info.html

**Example:** ErrorDocument 403 "Sorry, cannot allow you access today."

In the event of a problem or error, HTTP Server can be configured to do one of four things:

1. Output a simple hard coded error message.
2. Output a customized message.
3. Redirect to a local URL to handle the problem/error.
4. Redirect to an external URL to handle the problem/error.

The first option is the default, while options 2 through 4 are configured using the ErrorDocument directive, which is followed by HTTP Server response code and a message or URL.

For option 3, the document parameter must begin with a '/' character and it is assumed to be relative to DocumentRoot. If the document parameter contains a ':' character it is assumed to be an external URL (option 4). If neither of these are true, option 2 is assumed.

### Parameter One: *error-code*

- The *error-code* parameter specifies the error code associated with a hard coded error message, a customized message, a local URL, or an external URL that handles the problem/error.

### Parameter Two: *document*

- The *document* parameter specifies a hard coded error message, a customized message, a local URL, or an external URL that handles the problem/error.

*Messages* in this context begin with a single quote ("), which does not form part of the message itself. The server will sometimes offer additional information regarding the problem/error.

URLs must begin with a slash (/) for local URLs, or be a full URL which the client can resolve. For example:

```
ErrorDocument 500 http://QIBM.example.com/cgi-bin/tester
ErrorDocument 404 /cgi-bin/bad_urls.html
ErrorDocument 401 /subscription_info.html
ErrorDocument 403 "Sorry cannot allow you access today."
```

**Note:** When you specify an ErrorDocument that points to a remote URL (for example, anything with a method such as "http" in front of it) the server will send a redirect to the client to tell it where to find the document, even if the document ends up being on the same server. This has several implications, the most important being that if you use an "ErrorDocument 401" directive then it must refer to a local document. This results from the nature of the HTTP basic authentication scheme.

IBM HTTP Server for i allows error code keywords on this directive, in addition to HTTP response codes. This will allow customers more granularity in their error page customization. To do this, the syntax for ErrorDocument was enhanced to also allow one of these key words as the error\_code. Valid keywords, their equivalent HTTP response codes and the cause are as follows:

Error code	Error meaning
okredirect 302	The document has moved.
badrequest 400	The request is not valid.
badscript 400	The requested script file could not be processed; the request was invalid in some way.
connectfail 400	The server could not connect to the requested partner on the requested port.
nopartner 400	The server could not connect to the requested host name due to bad syntax or an unknown host.
proxyfail 400 or 502	The client tried to use the server as a proxy and, although this is allowed, it did not work. Possibly the destination server doesn't exist or is busy.
proxymterror (any code >= 400)	The server received a response code from a remote server that indicates a remote server problem and the proxy error override function has been invoked (see ProxyErrorOverride for more details).
unknownmethod 400	The request did not include a recognized method.
notauthorized 401	The request requires a user ID and password. Either the user ID and password sent by the client are not valid for this request or the client did not send a user ID and password.
notmember 401	The requested file has a protection rule listing valid user IDs and passwords and the user ID of the requesting client is not included in the list.
pwchanged 401	The password is invalid.
pwexpired 401	The password for the user ID has expired.
badredirect 403	The server is trying to redirect the request and the Redirect directive is invalid or contains a loop.
baduser 403	The client requested a user's home directory that does not exist.
byrule 403	A directive (such as deny or allow directive) or rule was specified that will not allow this request.
dirbrowse 403	The request specified a directory that is turned off for browsing.
dotdot 403	The client request specified a parent (/../) directory which is not allowed.
ipmask 403	The client's IP address is not a valid IP address for the request.
ipmaskproxy 403	The client is trying to use the server as a proxy, however the client is not included in the list of host names or IP addresses that are allowed to do so.
methoddisabled 403	The method requested has been disabled.

Error code	Error meaning
noacl 403	Cannot access the .htaccess file.
noentry 403	The user is not included in the list of valid users for this request.
notallowed 403	The server found the requested file but the protection setup of the server prevented access.
openfailed 403	The file or directory has access restrictions for the current user.
multifail 404	The requested file could not be found on the server.
proxynotauth 407	The request requires a user ID and password for the proxy. Either the user ID and password sent by the client are not valid for this request or the client did not send a user ID and password.
proxynotmember 407	The requested file has a protection rule listing valid user IDs and passwords and the user ID sent by the client is not included in that list.
proxypwchanged 407	The password sent by the client is not valid for the proxy.
proxypwexpired 407	The password sent by the client has expired.
preconfail 412	A precondition specified by the client on this request was not met. For example, this could result from HTTP/1.1 request that contains a condition "if-not-modified-since xxx".
badrange 416	The request either has an invalid content range header or it has incorrect information in the content range header for the file being processed.
upgrade 426	The request was received for a file which must be accessed through SSL. An upgrade to SSL is required before accessing this resource.
scriptio 500	The client requested a CGI script but the server cannot get it to process input or output. The script may contain invalid code.
scriptnotfound 500	The client requested a CGI script that cannot be found.
scriptstart 500	The client requested a CGI script that the server can find but cannot be started. The script may contain invalid code.
systemerror 500	An internal error occurred.
noformat 501	The server cannot interpret the format of the file it is trying to serve. The file may be corrupted or have an unknown or invalid file extension.

An example - a customer puts the following into their configuration file:

```
ErrorDocument byrule "Sorry cannot allow you access."
ErrorDocument openfailed "You do not have authority to this file."
```

When an HTTP response code of 403 (FORBIDDEN) occurs and it is determined that the reason is the client is on the deny list, the response back to the browser will be "Sorry cannot allow you access". If, however, the 403 response code is a result of the user not having authority to the file, the message will be "You do not have authority to this file". This gives the user more granularity to customize error responses to the client.

### ErrorLog:

**Module:** core

**Syntax:** ErrorLog *filename-or-pipe* | *off* | *\*off*

**Default:** ErrorLog logs/error\_log

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Example:** IFS example relative to server root: ErrorLog logs/errorlog

**Example:** Piped log example: ErrorLog |/QSYS.LIB/MYLIB.LIB/ERRPIPE.PGM

**Example:** QSYS example: ErrorLog /QSYS.LIB/MYLIB.LIB/MYLOGS.FILE

The ErrorLog directive sets the name of the file to which the server will log any errors it may encounter. If the filename does not begin with a slash (/) then it is assumed to be relative to the "ServerRoot" on page 326. Specifying a value of *off* or *\*off* will cause the server to not log errors.

**Parameter:** *filename-or-pipe* | *off* | *\*off*

- The *filename* parameter is relative to the ServerRoot or a full path to the file.
- A pipe (|) followed by a program to spawn to handle the error log information. Data written to the pipe from the server will be in the FSCCSID that is in use by the server.
- The *off* or *\*off* value turns off error reading.

**Note:** A new program will not be started for a VirtualHost if it inherits the ErrorLog from the main server. The program is specified in the form "qsys.lib/xxx.lib/xxx.pgm".

All messages logged to the Error log will be logged in the primary language installed for the IBM HTTP Server. The error log file will be created with a coded character set identifier (CCSID) that is compatible with the language. The CCSID value is an ASCII CCSID.

It is recommended that you allow the server to create the log file. Specifically:

- For IFS files, the user must create the directories that contain the log file and must grant the QTMHHTTP user write access to the directory. The server will create the log file.
- For QSYS.LIB logs, the user must create the library that contains the logs. The server will create the file and members in the specified library.
- If the filename does not begin with a slash (/) then it is assumed to be relative to the ServerRoot.
- If "LogCycle" on page 313 is active and if the path ends without a '/' character, then the path is considered to be the complete log file name. In that case, the server will add an extension in the format QCYMMDDHH, where these variables have the following values:
  - Q is a default value that indicates to the server that this is a log file.
  - C is the century indicator (0 for pre-2000, 1 for post-2000)
  - YY is the year indicator
  - MM is the month indicator
  - DD is the day indicator HH is the hour indicator (00 = 00:00 (midnight), 23=23:00)

**Note:** Will not be generated for file system QDL.

For example, a path of "/logs/errorlog" results in a file such as "/logs/errorlog.Q100030300".

- If "LogCycle" on page 313 is active and if the path ends with a '/' character, then the path is considered to be the directory that will contain the log file. In that case, the server will create log files named in the QCYMMDDHH format. For example, a path of "/logs/errorlog/" results in a file such as "/logs/errorlog/Q100030300".
- If "LogCycle" on page 313 is active and the log file is in the QSYS file system, the name must end it the file component of the IFS path. Example:

```
# Config file directives
LogCycle Daily
ErrorLog /QSYS.LIB/MYLIB.LIB/MYLOGS.FILE
```

The resulting daily log rollover files will be of the form /QSYS.LIB/MYLIB.LIB/MYLOGS.FILE/  
Qcyymmddhh.MBR

- "LogCycle" on page 313 Hourly is not valid if the log file is in the QDLS file system as that file system only supports 8 character file names and 3 character extensions.

- If “LogCycle” on page 313 is not active, no special naming is used. The name of the log file given on the ErrorLog directive is used as given for the name of the log file. If the name is a directory, a default name of http.log will be concatenated to the directory name to create the log file. For example:

```
# Config file directives
LogCycle Off
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog /logs/path/ common
```

The resulting log file will be /logs/path/http.log.

### Security:

See “Security tips for HTTP Server” on page 29 details on why your security could be compromised if the directory where log files are stored is writable by anyone other than the user that starts the server. If a program is used, then it will be run under the user who started httpd. This will be root if the server was started by root (be sure that the program is secure).

See also “LogLevel” on page 315.

### ErrorLogFormatDDS:

**Module:** core  
**Syntax:** ErrorLogFormatDDS *on* | *off*  
**Default:** ErrorLogFormatDDS *off*  
**Context:** server, virtual host, not in limit  
**Override:** none  
**Origin:** IBM  
**Example:** ErrorLogFormatDDS *on*

The ErrorLogFormatDDS directive is used to specify whether the server should log information to the ErrorLog in the IBM i data description specification (DDS) format. This directive provides equivalent error logging function to the HTTP Server (original) DDS LogFormat.

If the parameter is set to *on*, the server creates a DDS log file and each record contains the format described by file QHTTSPVR/QAZHBERR. When the parameter is *on*, the path name specified on the ErrorLog directive must be the internal file system syntax for a file in the QSYS.LIB file system, or a server start up error results. To view a description of the fields within a log file created with DDS format, use the command DSPFFD FILE(QHTTSPVR/QAZHBERR).

By default, the IBM i logs is in Apache's standard ErrorLog format.

### FileETag:

**Module:** core  
**Syntax:** FileETag *component* ...  
**Default:** FileETag All  
**Context:** server config, virtual host, directory, .htaccess  
**Override:** FileInfo  
**Origin:** Apache  
**Example:** FileETag INode MTime Size

The FileETag directive configures the file attributes that are used to create the ETag (entity tag) response header field when the document is based on a file. The ETag value is used in cache management to save network bandwidth. In Apache 1.3.22 and earlier, the ETag value was always formed from the file's inode, size, and last-modified time (mtime). The FileETag directive allows you to choose which of these (if any) should be used.



**Parameter:** *component*

- **INode** indicates the file's inode number will be included in the calculation.  
  
**Note:** INode is the file ID number for the object. This number uniquely identifies the object within a file system. It is part of the stat structure (the `st_ino` field of the stat structure).
- **MTime** indicates the date and time the file was last modified will be included.
- **Size** indicates the number of bytes in the file will be included.
- **All** indicates all available fields will be used (equivalent to 'FileETag INode MTime Size').
- **None** indicates that if a document is file-based, no ETag field will be included in the response.

The **INode**, **MTime**, and **Size** keywords may be prefixed with either '+' or '-', which allow changes to be made to the default setting inherited from a higher level context. Any keyword appearing without such a prefix immediately and completely cancels the inherited setting.

If a directory's configuration includes 'FileETag INode MTime Size', and a subdirectory's includes 'FileETag -INode', the setting for that subdirectory (which will be inherited by any sub-subdirectories that don't override it) will be equivalent to 'FileETag MTime Size'.

The **MTime** attribute (if specified) may be used by remote proxy servers to calculate cache expiry times in the event that document expiry times are not available or provided.

See `CacheLastModifiedFactor` for more information.

#### <Files>:

**Module:** core

**Syntax:** <Files *filename*> ... </Files>

**Default:** none

**Context:** server config, virtual host, .htaccess, Not in Limit

**Override:** none

**Origin:** Apache

**Example:** <Files index.html>

The <Files> directive provides for access control by filename. It is comparable to the “<Directory>” on page 291 directive and “<Location>” on page 312 directives. It should be matched with a </Files>. Directives given within this section will be applied to any object with a base-name (last component of filename) matching the specified filename. <Files> sections are processed in the order they appear in the configuration file, after the <Directory> sections and .htaccess files are read, but before <Location> sections. Note that <Files> can be nested inside <Directory> sections to restrict the portion of the file system.

**Parameter:** *filename*

- The *filename* parameter should include a filename or a wildcard string where '?' matches any single character and '\*' matches any sequences of characters. For example:

```
<Files index.html>
  Order allow,deny
  allow from all
</Files>
```

- Extended regular expressions can also be used, with the addition of the '~' character.

**Note:** Unlike <Directory> and <Location> sections, <Files> sections can be used inside .htaccess files. This allows users to control access to their own files, at a file-by-file level. See “Security tips for HTTP Server” on page 29 and “User profiles and required authorities for HTTP Server” on page 30 for more details.

## <FilesMatch>:

**Module:** core

**Syntax:** <FilesMatch *regex*> ... </FilesMatch>

**Default:** none

**Context:** server config, virtual host, .htaccess, Not in Limit

**Override:** none

**Origin:** Apache

**Example:** <FilesMatch "\.(gif|jpe?g|png)\$">

The <FilesMatch> directive provides for access control by filename, in the same way “<Files>” on page 299 directive does. The <FilesMatch> directive, however, accepts a regular expression. For example:

```
<FilesMatch "\.(gif|jpe?g|png)$">
```

This would match most common Internet graphic formats.

**Note:** The argument to <FilesMatch> does not need to be in quotes unless the regular expression includes a space character.

**Parameter:** *regex*

- *Regex* is a UNIX-style regular expression that is matched against the URL. Subexpressions are grouped within parentheses. Then, parenthetically enclosed regular expressions will be substituted in a subsequent \$*n* statement.

## ForceType:

**Module:** core

**Syntax:** ForceType *media\_type*

**Default:** none

**Context:** directory, .htaccess

**Override:** FileInfo

**Origin:** Apache

**Example:** ForceType image/gif (forces all files in the container to be treated as a GIF file)

The ForceType directive forces all matching files to be served as the content type given by media type when they are placed into an .htaccess file, a <Directory>, or <Location> section.

**Parameter:** *media\_type*

- The *media\_type* parameter is a MIME type/subtype to which all files in the directory will be forced.

**Note:** This overrides any filename extensions that might determine the media type.

## ForensicLog:

**Module:** core

**Syntax:** ForensicLog *path*to *logfile*

**Default:** none

**Context:** server

**Override:** none

**Origin:** Apache

**Example:** ForensicLog logs/forensic\_log

This module provides for forensic logging of client requests. Logging is done before and after processing a request, so the forensic log contains two log lines for each request. The forensic logger is very strict. The

format is fixed. You cannot modify the logging format at runtime. Each request is logged two times. The first time after receiving the headers. The second log entry is written after the request processing at the same time when normal logging occurs. To identify each request, a unique request ID is assigned. This forensic ID can be cross logged in the normal transfer log using the `%{forensic-id}n` format string. If you are using `mod_unique_id` its generated ID will be used.

The first line logs the forensic ID, the request line and all received headers, separated by pipe characters (`|`). A sample line follows:

```
+yQtJf8CoAB4AAFNBIEAAAAA|GET /manual/de/images/down.gif HTTP/1.1|Host:localhost%3
a8080|User-Agent:Mozilla/5.0 (X11; U; Linux i686; en-US; rv%3a1.6) Gecko/20040216
Firefox/0.8|Accept:image/png
```

**Note:** The examples are wrapped for display purposes only.

The plus character at the beginning indicates that this is first log line of this request. The second line just contains a minus character and the id again:

```
-yQtJf8CoAB4AAFNBIEAAAAA
```

The `check_forensic` script takes as its argument the name of the logfile. It looks for the  $\pm$  ID pairs and issues an error if a request was not completed.

### HostNameLookups:

**Module:** core

**Syntax:** `HostNameLookups on | off | double`

**Default:** `HostNameLookups off`

**Context:** server config, virtual host, directory, Not in Limit

**Override:** none

**Origin:** Apache

**Example:** `HostNameLookups on`

The `HostNameLookups` directive enables DNS lookups so the host names can be logged (and passed to CGIs/SSIs in the `REMOTE_HOST` environment variable).

**Parameter:** `on | off | double`

- The *on* value enables DNS lookups so the host names can be logged (and passed to CGIs/SSIs in the `REMOTE_HOST` environment variable).
- The default *off* value saves on the network traffic for those sites that do not truly need the reverse lookup. Heavily loaded sites should leave this directive set to off, since DNS lookups can take a considerable amount of time.
- The value *double* refers to doing double-reverse DNS. That is, after a reverse lookup is performed, a forward lookup is then performed on that command. At least one of the IP addresses in the forward lookup must match the Original address. When `mod_access` is used for controlling access by hostname, regardless of the setting, a double reverse lookup will be performed. This is necessary for security.

**Note:** The result of this double-reverse isn't generally available unless you set `HostnameLookups double`. For example, if you only set `HostnameLookups on` and a request is made to an object that is protected by hostname restrictions, regardless of whether the double-reverse fails or not, CGIs will still be passed to the single-reverse result in `REMOTE_HOST`.

### HotBackup:

**Module:** core

**Syntax:** `HotBackup on | off`

**Default:** HotBackup on  
**Context:** server config  
**Override:** none  
**Origin:** IBM  
**Example:** HotBackup on

The HotBackup directive is used to specify whether or not a hot backup server should be started at the server startup time. With the hot backup server active, if the primary server job abnormally terminates, the hot backup will immediately take over and act as the primary and continue servicing requests. A new hot backup is automatically created, in the background, within one minute. However, if more than five consecutive server failures occur within a ten minute time period, no additional hot backups will be created and the server will fail. The server is allowed to fail in this situation to avoid system degradation, since the hot backup processing can consume system resources.

If the primary server process failure is not due to the network, all user connections remain active during the hot backup take over and the end users do not detect the loss of server; however, some HTTP requests in transient may be lost. If the failure is due to the loss of network, the server must be restarted.

For a full backup recovery, including system and network failures, refer to highly available Web server.

**Parameter:** *on* | *off*

- When set to *on*, if the primary server job abnormally terminates, the hot backup will immediately take over and act as the primary and continue servicing requests.
- With HotBackup *off*, only one multithreaded server child process is started.

**Note:** When a server is configured as highly available (HAModel directive is specified), HotBackup behaves as if it is set to '*off*' and can not be overwritten.

### HTTPSubsystemDesc:

**Module:** core  
**Syntax:** HTTPSubsystemDesc *library* | *subsystem*  
**Default:** HTTPSubsystemDesc QHTTPSVR/QHTTPSVR  
**Context:** server config  
**Override:** none  
**Origin:** IBM  
**Example:** HTTPSubsystemDesc HTTPTEST/HTTP SBS

The HTTPSubsystemDesc directive specifies the user created subsystem that the HTTP server runs in. By default HTTP server runs under QHTTPSVR/QHTTPSVR subsystem.

The subsystem must already exist before using this directive, otherwise HTTP server will fail to start. The subsystem can be automatically started if it's not active when starting HTTP server but will not be ended when stopping the HTTP server

**Note:** To make HTTP server run in subsystem other than QHTTPSVR, at least "HTTPStartJobQueue" on page 303 directive is required to be specified and the desired subsystem is MUST in active status before starting HTTP server. If only HTTPSubsystemDesc directive is specified, only the specified subsystem is started and HTTP server jobs still run under QHTTPSVR. If only "HTTPStartJobQueue" on page 303 is specified but the desired subsystem is not active at that moment, the HTTP server jobs will not be started until the subsystem is started

## HTTPStartJobQueue:

**Module:** core  
**Syntax:** HTTPStartJobQueue *library* | *jobqueue*  
**Default:** HTTPStartJobQueue QHTTPSVR/QZHBHTTP  
**Context:** server config  
**Override:** none  
**Origin:** IBM  
**Example:** HTTPStartJobQueue HTTPTEST/HTTPJOBQ

The HTTPStartJobQueue directive specifies the user created job queue to which the HTTP server jobs will be submitted. The default HTTP server job queue is QHTTPSVR/QZHBHTTP.

The job queue must already exist before using this directive, otherwise HTTP server will fail to start.

**Note:** To make HTTP server run in subsystem other than QHTTPSVR, at least HTTPStartJobQueue directive is required to be specified and the desired subsystem is **MUST** in active status before starting HTTP server. If only “HTTPSubsystemDesc” on page 302 directive is specified, only the specified subsystem is started and HTTP server jobs still run under QHTTPSVR. If only HTTPStartJobQueue is specified but the desired subsystem is not active at that moment, the HTTP server jobs will not be started until the subsystem is started.

## HTTPStartJobDesc:

**Module:** core  
**Syntax:** HTTPStartJobDesc *library* | *jobdescription*  
**Default:** HTTPStartJobDesc QHTTPSVR/QZHBHTTP  
**Context:** server config  
**Override:** none  
**Origin:** IBM  
**Example:** HTTPStartJobDesc HTTPTEST/HTTPJOBQ

The HTTPStartJobDesc directive specifies the user created job description which defines how HTTP server jobs should be run. The default HTTP server job description is QHTTPSVR/QZHBHTTP.

The job description must already exist before using this directive, otherwise HTTP server will fail to start.

**Note:** To make HTTP server run in subsystem other than QHTTPSVR, at least “HTTPStartJobQueue” directive is required to be specified and the desired subsystem is **MUST** in active status before starting HTTP server. If only HTTPSubsystemDesc directive is specified, only the specified subsystem is started and HTTP server jobs still run under QHTTPSVR. If only “HTTPStartJobQueue” is specified but the desired subsystem is not active at that moment, the HTTP server jobs will not be started until the subsystem is started.

## HTTPRoutingData:

**Module:** core  
**Syntax:** HTTPRoutingData *name*  
**Default:** HTTPRoutingData HTTPWWW  
**Context:** server config  
**Override:** none  
**Origin:** IBM  
**Example:** HTTPRoutingData HTTPSVR

The HTTPRoutingData directive specifies the user defined routing data for HTTP server jobs. The default value is HTTPWWW. A maximum of 80 characters can be specified.

The routing entry must be already added to the HTTP subsystem before using this directive, otherwise HTTP server will fail to start.

**Note:** To make HTTP server run in subsystem other than QHTTPSVR, at least "HTTPStartJobQueue" on page 303 directive is required to be specified and the desired subsystem is MUST in active status before starting HTTP server. If only HTTPSubsystemDesc directive is specified, only the specified subsystem is started and HTTP server jobs still run under QHTTPSVR. If only "HTTPStartJobQueue" on page 303 is specified but the desired subsystem is not active at that moment, the HTTP server jobs will not be started until the subsystem is started.

### IdentityCheck:

**Module:** core

**Syntax:** IdentityCheck *on* | *off*

**Default:** IdentityCheck off

**Context:** server config, virtual host, directory, Not in Limit

**Override:** none

**Origin:** Apache

**Example:** IdentityCheck on

The IdentityCheck directive enables compliant logging of the remote user name for each connection, where the client machine runs `identd` or something similar. This information is logged in the access log.

**Parameter:** *on* | *off*

- When set to *on*, the server will attempt to identify the client's user by querying the `identd` daemon of the client host. `Identd` will, when given a socket number, reveal which user created that socket. That is, the username of the client on his home machine. Since the information provided is entirely under the control of the client's machine, this information should not be trusted in any way except for rudimentary usage tracking.
- When set to *off*, the server does not attempt to identify the client's user.

**Note:** This can cause serious latency problems accessing your server since every request requires one of these lookups to be performed. When firewalls are involved each lookup might possibly fail and add 30 seconds of latency to each hit. So in general this is not very useful on public servers accessible from the Internet. This directive controls the `identd` field of the W3C common or extended log format.

A `CustomLog`, `TransferLog` or `FRCACustomLog` must be configured before this directive will take affect. If `IdentityCheck` is configured in a directory or location container, the `CustomLog`, `TransferLog` or `FRCACustomLog` must be configured in the server context where the directory or location container resides for it to take affect. Also for this directive to be used in the `CustomLog`, `TransferLog`, or `FRCACustomLog`, the `LogFormat` for these has to specify "%l" (lower case L) in the format.

See "Module `mod_log_config`" on page 405 for information on log formats.

### <IfDefine>:

**Module:** core

**Syntax:** <IfDefine <parameter-name> ... </IfDefine>

**Default:** none

**Context:** server config

**Override:** none

**Origin:** Apache

**Example:** <IfDefine LDAP>

The <IfDefine *test*> ... </IfDefine> section is used to mark directives that are conditional. The directives within an IfDefine section are only processed if the test is true. If the test is false, everything between the start and end markers is ignored.

The *test* in the <IfDefine> section directive can be one of the two forms:

- *parameter-name*
- *!parameter-name*

In the former case, the directives between the start and end markers are only processed if the parameter named *parameter-name* is defined. The second format reverses the test, and only processes the directives if *parameter-name* is not defined.

**Parameter:** *parameter-name*

- The *parameter-name* parameter is defined as given on the STRTCPSVR command line via *-Dparameter*, at the time the server was started. <IfDefine> sections are nestable, which can be used to implement simple multiple-parameter tests. For example:

```
STRTCPSVR '-DLdap'  
# in the instance configuration  
<IfDefine LDAP>  
    LoadModule ldap_module /QSYS.LIB/QHTTSPVR.LIB/MOD_LDAP.SRVPGM  
</IfDefine>
```

If customer has included "<IfDefine keyword> ... </IfDefine>" in http.conf file, the directives that are in context will be only valid if the command "STRTCPSVR" has included this directive, in this case "STRTCPSVR '-Dkeyword' ", if not, the server will be ignored then.

**<IfModule>:**

**Module:** core

**Syntax:** <IfModule [!]*module-name*> ... </IfModule>

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** none

**Origin:** Apache

**Example:** <IfModule test>

The <IfModule> directive is used to mark directives that are conditional. The directives within an <IfModule> section are only processed if the test is true. If the test is false, everything between the start and end markers is ignored.

The *test* in <IfModule> section directive can be one of two forms:

- *module-name*
- *!module-name*

**Parameter:** *module-name*

- The *module-name* parameter is a module name as given as the file name of the module at the time it was compiled. For example:

```
mod_rewrite.c
```

<IfModule> sections are nestable which can be used to implement simple multiple-module tests.

## **Include:**

**Module:** core

**Syntax:** Include *filename*

**Default:** none

**Context:** server config, virtual host, directory

**Override:** none

**Origin:** Apache

**Example:** Include none/mydirectory/myfile

The Include directive allows inclusion of other configuration files from within the server configuration files. The filename can be either a relative or absolute path.

**Parameter:** *filename*

- The *filename* value identifies other configuration files from within the server configuration files.

**Note:** The filename specified with this directive must be in a file in the Root or QOpenSys file systems. Other file systems are not supported.

## **KeepAlive:**

**Module:** core

**Syntax:** KeepAlive *on* | *off*

**Default:** KeepAlive on

**Context:** server config, virtual host, Not in Limit

**Override:** none

**Origin:** Apache

**Example:** KeepAlive off

The KeepAlive directive enables keep-alive support (also known as persistent connections).

**Parameter:** *on* | *off*

- When set to *on*, the directive enables keep-alive support (also known as persistent connections).
- When set to *off*, keep-alive support (also known as persistent connections) is disabled.

Persistent connections enable a single TCP connection to be used for multiple HTTP requests. Normally, each HTTP request uses a separate connection. Reusing a single connection reduces the connection open/close overhead, thereby improving performance for that client. However with dynamic content, depending on your Web applications, using persistent connections can reserve server resources for each client, thereby reducing the throughput of your server as a whole. Therefore, care should be taken when modifying persistent connection related settings.

Set to *off* to disable persistent connections, *on* to enable. If the KeepAlive directive value is not *off* or zero, *on* is assumed.

See also “KeepAliveTimeout” and “MaxKeepAliveRequests” on page 318.

## **KeepAliveTimeout:**

**Module:** core

**Syntax:** KeepAliveTimeout *seconds*

**Default:** KeepAliveTimeout 300

**Context:** server config, virtual host, Not in Limit

**Override:** none

**Origin:** Apache



**Example:** KeepAliveTimeout 500

The KeepAliveTimeout directive is related to persistent connections and determines the number of seconds HTTP Server waits for a subsequent request before closing the connection. The KeepAlive directive must be set to *on*, enabling persistent connections, for this directive to take effect. It is recommended that this value be set high enough to prevent time outs. Note that this is related to the time between requests and not during requests. Once a request is received, the connection timeout setting (set by the TimeOut directive) applies. The connection time-out applies until request processing is complete and (until the next request is received) the persistent connections related timer setting is applied.

**Parameter:** *seconds*

- The *seconds* value determines the number of seconds HTTP Server waits for a subsequent request before closing the connection.

See also “KeepAlive” on page 306, “MaxKeepAliveRequests” on page 318, and “TimeOut” on page 330.

### <Limit>:

**Module:** core

**Syntax:** <Limit *method method ...* > ... </Limit>

**Default:** none

**Context:** server config, virtual host, directory, .htaccess, Not in Limit

**Override:** none

**Origin:** Modified

**Example:** <Limit GET PUT>

The purpose of the <Limit> directive is to restrict the effect of the access controls to the nominated HTTP methods. For all other methods, the access restrictions that are enclosed in the <Limit> bracket will have no effect. The following example applies the access control only to the methods POST, PUT, and DELETE, leaving all other methods unprotected:

```
<Limit POST PUT DELETE>
    require valid-user
</Limit>
```

Access controls are normally effective for all access methods, and this is the usual desired behavior. In the general case, access control directives should not be placed within a <Limit> section.

**Parameter:** *method*

- *Method* names listed can be one or more of the following: **GET, POST, PUT, DELETE, CONNECT, OPTIONS, PATCH, PROPFIND, PROPPATCH, MKCOL, COPY, MOVE, LOCK** and **UNLOCK**. The method name is case sensitive. If GET is used it will also restrict HEAD requests.

### <LimitExcept>:

**Module:** core

**Syntax:** <LimitExcept *method method ...* > ... </LimitExcept>

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** none

**Origin:** Modified

**Example:** <LimitExcept GET >

<LimitExcept> and </LimitExcept> are used to enclose a group of access control directives which will then apply to any HTTP access method not listed in the arguments; for example, it is the opposite of a "<Limit>" on page 307 section and can be used to control both standard and nonstandard-unrecognized methods. See "<Limit>" on page 307 for more details.

**Parameter:** *method*

- *Method* names listed can be one or more of the following: **GET, POST, PUT, DELETE, CONNECT, OPTIONS, TRACE, PATCH, PROPFIND, PROPPATCH, MKCOL, COPY, MOVE, LOCK** and **UNLOCK**. The method name is case sensitive. If GET is specified, HEAD is also allowed (not restricted).

### **LimitRequestBody:**

**Module:** core

**Syntax:** LimitRequestBody *number*

**Default:** LimitRequestBody 0

**Context:** server config, virtual host, directory, .htaccess, Not in Limit

**Override:** none

**Origin:** Apache

**Example:** LimitRequestBody 100

The LimitRequestBody directive allows the user to set a limit on the allowed size (in bytes) of an HTTP Request message body within the context in which the directive is given (server, per-directory, per-file or per-location). If the client Request exceeds that limit, the server will return an error response instead of servicing the Request. The size of a normal Request message body will vary greatly depending on the nature of the resource and the methods allowed on that resource. CGI scripts typically use the message body for passing form information to the server. Implementations of the PUT method will require a value at least as large as any representation that the server wants to accept for that resource.

This directive gives the server administrator greater control over abnormal client Request behavior, which may be useful for avoiding some forms of denial-of-service attacks.

**Parameter:** *number*

- The *number* parameter is an integer which represents the set limit on the allowed size (in bytes) of an HTTP Request message body within the context in which the directive is given (server, per-directory, per-file or per-location). The default value of '0' (zero) indicated unlimited allowed size.

For example, to limit the size of an uploaded file to 100K use the following:

```
LimitRequestBody 10240
```

### **LimitInternalRecursion:**

**Module:** core

**Syntax:** LimitInternalRecursion *number* [*number*]

**Default:** LimitInternalRecursion 10

**Context:** server, virtual host

**Override:** none

**Origin:** Apache

**Example:** LimitInternalRecursion 5

An internal redirect happens, for example, when using the Action directive, which internally redirects the original request to a CGI script. A subrequest is Apache's mechanism to find out what would happen for some URI if it were requested. For example, mod\_dir uses subrequests to look for the files listed in the DirectoryIndex directive.

LimitInternalRecursion prevents the server from crashing when entering an infinite loop of internal redirects or subrequests. Such loops are usually caused by misconfigurations. The directive stores two different limits, which are evaluated on per-request basis. The first number is the maximum number of internal redirects, that may follow each other. The second number determines, how deep subrequests may be nested. If you specify only one number, it will be assigned to both limits.

### LimitRequestFields:

**Module:** core

**Syntax:** LimitRequestFields *number*

**Default:** LimitRequestFields 100

**Context:** server config, Not in Limit

**Override:** none

**Origin:** Apache

**Example:** LimitRequestFields 800

The LimitRequestFields directive allows the server administrator to modify the limit on the number of Request header fields allowed in an HTTP Request. A server needs this value to be larger than the number of fields that a normal client Request might include. The number of Request header fields used by a client rarely exceeds 20, but this may vary among different client implementations, often depending upon the extent to which a user has configured their browser to support detailed content negotiation. Optional HTTP extensions are often expressed using Request header fields.

This directive gives the server administrator greater control over abnormal client Request behavior, which may be useful for avoiding some forms of denial-of-service attacks. The value should be increased if normal clients see an error response from the server that indicates too many fields were sent in the Request.

**Parameter:** *number*

- The *number* parameter is an integer from 0 (meaning unlimited) to 32767 bytes. The default value is 100.

### LimitRequestFieldsize:

**Module:** core

**Syntax:** LimitRequestFieldsize *number*

**Default:** LimitRequestFieldsize 32766

**Context:** server config, Not in Limit

**Override:** none

**Origin:** Apache

**Example:** LimitRequestFieldsize 8000

The LimitRequestFieldsize directive allows the server administrator to reduce the limit on the allowed size of an HTTP Request header field below the normal input buffer size compiled with the server. A server needs this value to be large enough to hold any one header field from a normal client Request. The size of a normal Request header field will vary greatly among different client implementations, often depending upon the extent to which a user has configured their browser to support detailed content negotiation.

This directive gives the server administrator greater control over abnormal client Request behavior, which may be useful for avoiding some forms of denial-of-service attacks. Under normal conditions, the value should not be changed from the default.

**Parameter:** *number*

- A *number* is an integer from 0 to 32766 (in bytes).

## LimitRequestLine:

**Module:** core

**Syntax:** LimitRequestLine *number*

**Default:** LimitRequestLine 8190

**Context:** server config, Not in Limit

**Override:** none

**Origin:** Apache

**Example:** LimitRequestLine 8000

The LimitRequestLine directive allows the server administrator to reduce the limit on the allowed size of a client's HTTP Request-line below the normal input buffer size compiled with the server. Since the Request-line consists of the HTTP method, URI, and protocol version, the LimitRequestLine directive places a restriction on the length of a Request-URI allowed for a Request on the server. A server needs this value to be large enough to hold any of its resource names, including any information that might be passed in the query part of a GET Request.

This directive gives the server administrator greater control over abnormal client Request behavior, which may be useful for avoiding some forms of denial-of-service attacks. Under normal conditions, the value should not be changed from the default.

**Parameter:** *number*

- A *number* is an integer from 0 to 8190 (in bytes).

## LimitXMLRequestBody:

**Module:** core

**Syntax:** LimitXMLRequestBody *number*

**Default:** LimitXMLRequestBody 1000000

**Context:** server config, virtual host, directory (but not location), .htaccess, Not in Limit

**Override:** none

**Origin:** Apache

**Example:** LimitXMLRequestBody 1000000

The LimitXMLRequestBody directive limits (in bytes) the maximum size of an XML-based request body.

**Parameter:** *number*

- A *number* is an integer from 0 (meaning unlimited) to 2,147,483,647 (2 gigabytes).

## Listen:

**Module:** core

**Syntax:** Listen [*IP address*:] *port number* [*protocol*]

**Default:** Listen 80

**Context:** server config

**Override:** none

**Origin:** Apache

**Example:** Listen 8000

**Example:** Listen 8000 FRCA

**Note:** FRCA support for the Listen directive is not available for V5R1 and earlier releases of HTTP Server.

The Listen directive instructs HTTP Server to listen to more than one IP address or port; by default the server responds to requests on all IP interfaces. It tells the server to accept incoming requests on the

specified IP address or address-and-port combinations. If the first format is used, with an IP address number only, the server listens to the given IP address. If an IP address is given as well as a port, the server will listen on the given port and interface.

**Parameter One:** *IP address*

- The *IP address* parameter specifies a fully qualified IP address.

**Parameter Two:** *port number*

- The *port number* parameter is optional and if specified as word "FRCA", implies the incoming connections on the specified IP address and port are eligible to be monitored and served by FRCA cache support.

**Note:** FRCA does not support SSL. Therefore, do not specify FRCA option for IP addresses and ports that are used for SSL connections.

**Parameter Three:** *[protocol]*

- The *[protocol]* parameter is optional and if specified as word "FRCA", implies the incoming connections on the specified IP address and port are eligible to be monitored and served by FRCA cache support.

**Note:** FRCA does not support SSL. Therefore, do not specify FRCA option for IP addresses and ports that are used for SSL connections.

Multiple Listen directives may be used to specify a number of addresses and ports to listen to. The server will respond to requests from any of the listed addresses and ports.

To make the server accept connections on both port 80 and port 8000, use:

```
Listen 80
Listen 8000
```

To make the server accept connections on two specified interfaces and port numbers, use:

```
Listen 194.170.2.1:80
Listen 194.170.2.5:8000
```

IPv6 addresses must be surrounded in square brackets, as in the following example:

```
Listen [2001:db8::a00:20ff:fea7:ccea]:80
```

To make the FRCA monitor and intercept connections on a specified interface and port numbers, use:

```
Listen 194.170.2.5:8000 FRCA
```

In the above example, since the optional parameter *FRCA* is specified, FRCA will be enabled for the specified IP address and port.

The optional protocol argument is not required for most configurations. If not specified, https is the default for port 443 and http the default for all other ports. The protocol is used to determine which module should handle a request, and to apply protocol specific optimizations with the AcceptFilter directive. You only need to set the protocol if you are running on non-standard ports. For example, running an https site on port 8443:

```
Listen 192.170.2.1:8443 https
```

**Error condition:** Multiple Listen directives for the same ip address and port will result in an Address already in use error message.

**ListenBacklog:**

**Module:** core

**Syntax:** ListenBacklog *backlog*  
**Default:** ListenBacklog 511  
**Context:** server config, Not in Limit  
**Override:** none  
**Origin:** Apache  
**Example:** ListenBacklog 400

The ListenBacklog sets the maximum length of the queue for pending connections. Generally no tuning is needed; however, on some systems it is desirable to increase this when under a TCP SYN flood attack.

**Parameter:** *backlog*

- The *backlog* parameter is an integer value that sets the maximum length of the queue for pending connections.

**<Location>:**

**Module:** core  
**Syntax:** <Location *url*> ... </Location>  
**Default:** none  
**Context:** server config, virtual host, Not in Limit  
**Override:** none  
**Origin:** Apache  
**Example:**

```
Alias /a /b  
<Location /a>
```

The <Location> directive limits the scope of the enclosed directives by URL (the URL is the virtual path used to access a resource), and is similar to the “<Directory>” on page 291 and <Proxy> directives, and starts a subsection which is terminated with a </Location> directive. Everything that is syntactically allowed in <Directory> is also allowed in <Location> (except a sub-“<Files>” on page 299 section). However some directives, most notably the “AllowOverride” on page 289 directive and two of its options (FollowSymLinks and SymLinksIfOwnerMatch) do not belong in <Location>. <Location> sections are processed in the order they appear in the configuration file (as opposed to <Directory> sections which are processed from the least match to the best match). They are processed after the <Directory> and <Proxy> sections, after .htaccess files are read, and after the <Files> sections. See “<Directory>” on page 291, <Proxy>, “<Files>” on page 299, and “AllowOverride” on page 289 directives for more information on <Directory>, <Proxy>, and <Files> containers and access files.

**Parameter :** *url*

- The *url* parameter consists of a URL.  
For all origin (non-proxy) requests, the URL to be matched is of the form /path/. The URL should not include the http://servername prefix. For proxy requests, the matched URL is of the form http://servername/path (you must include the prefix).  
The URL may use wildcards in a wildcard string. '?' matches any single character; '\*' matches any sequence of characters.

**Note:** URLs do not have to line up with the file system. <Location> operates completely outside the file system.

Extended regular expressions can also be used, with the addition of the ~ character. For example:

```
<Location ~ "/(extra|special)/data">
```

This would match URLs that contained the substring `"/extra/data"` or `"/special/data"`. The directive `<LocationMatch>` behaves identical to the regex version of `<Location>`. The `<Location>` functionality is especially useful when combined with the `"SetHandler"` on page 328 directive. For example, to enable origin requests, but allow them only from browsers at QIBM.com, you might use:

```
<Location /Origin>
  SetHandler server-Origin
  order deny,allow
  deny from all
  allow from .QIBM.com
</Location>
```

**Note:** The slash character has special meaning depending on where in a URL it appears. People may be used to its behavior in the file system where multiple adjacent slashes are frequently collapsed to a single slash (for example, `/home///QIBM` is the same as `/home/QIBM`). For `<Location>` this is not necessarily true. The `<LocationMatch>` directive and the regex version of `<Location>` require you to explicitly specify multiple slashes if that is your intention. For example, `<LocationMatch ^/ABC>` would match the request URL `/ABC` but not the request URL `//ABC`. The (non-regex) `<Location>` directive behaves similarly when used for proxy requests. But when (non-regex) `<Location>` is used for non-proxy requests it will implicitly match multiple slashes with a single slash. For example, if you specify `<Location /ABC/def>` and the request is to `/ABC//def`, the request will match the location.

#### **<LocationMatch>:**

**Module:** core

**Syntax:** `<LocationMatch regex> ... </LocationMatch>`

**Default:** none

**Context:** server config, virtual host, Not in Limit

**Override:** none

**Origin:** Apache

**Example:** `<LocationMatch "/(extra|special)/data">`

The `<LocationMatch>` directive provides for access control by URL. This directive works in an identical manner to the `"<Location>"` on page 312 directive. However, it takes a regular expression as an argument instead of a simple string.

**Parameter:** *regex*

- The *regex* parameter is a UNIX-style regular expression that is matched against the URL.

For example:

```
<LocationMatch "/(extra|special)/data">
```

This would match URLs that contained the substring `"/extra/data"` or `"/special/data"`.

**Note:** The argument to `LocationMatch` does not need to be in quotes unless the regular expression includes a space character.

#### **LogCycle:**

**Module:** core

**Syntax:** `LogCycle Off | Hourly | Daily | Weekly | Monthly`

**Default:** `LogCycle Daily`

**Context:** server config, Not in Limit

**Override:** none

**Origin:** IBM

**Example:** `LogCycle Monthly`

The LogCycle directive controls the server's log cycle. This refers to how often the server will close all log files and open new files with a new date/time stamp.

**Parameter:** *Off* | *Hourly* | *Daily* | *Weekly* | *Monthly*

- If *Off* is specified, one continuous log file is generated. Log files are not rolled over.
- If *Hourly* is specified, log files are closed and a new one created at the end of each hour.
- If *Daily* is specified, log files are closed and a new one created at midnight each day.
- If *Weekly* is specified, log files are closed and a new one created at midnight each Sunday morning. Weekly may not work correctly if the system is not using the Gregorian calendar (this would be similar to the help behind system value QDAYOFWEEK).
- If *Monthly* is specified, log files are closed and a new one created at midnight on the first day of the month.

**Note:** Daily and monthly log rollovers will always occur at midnight. Hourly rollovers will occur at the top of the hour. At the end of a log cycle, HTTP Server will roll over all logs. That means that it will flush all entries to the log file, close the current logs, and create a log file with a timestamp for the next log cycle.

If LogCycle is active and the path defined on an ErrorLog, CustomLog, TransferLog, or FRCACustomLog directive ends without a (/) character, then the path is considered to be the complete log file name. In that case, the server will add an extension to the given file with the format QCYMMDDHH, where these variables have the following values:

- Q is a default value that indicates to the server that this is a log file.
- C is the century indicator (0 for pre-2000, 1 for post-2000)
- YY is the year indicator
- MM is the month indicator
- DD is the day indicator
- HH is the hour indicator (00 = 00:00 (midnight), 23=23:00)

**Note:** HH will not be generated for file system QDLS.

For example, a path of "/logs/errorlog" results in a file such as "/logs/errorlog.Q100030300".

If LogCycle is active and the path defined on an ErrorLog, CustomLog, TransferLog, or FRCACustomLog directive ends with a (/) character, then the path is considered to be the directory that will contain the log file. In that case, the server will create log files named in the QCYMMDDHH format. For example, a path of "/logs/errorlog/" created on March 3, 2001 results in a file such as "/logs/errorlog/Q101030300".

If LogCycle is active and the path defined on an ErrorLog, CustomLog, TransferLog, or FRCACustomLog directive is in the QSYS file system, the name must end with the file component of the IFS path. For example:

```
# Config file directives
LogCycle Daily
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog /QSYS.LIB/MYLIB.LIB/MYLOGS.FILE common
```

The resulting daily log rollovers will be of the form /QSYS.LIB/MYLIB.LIB/MYLOGS.FILE/Qcyymmddhh.MBR.

If LogCycle is not active, no special naming is used. The name of the log file given on an ErrorLog, CustomLog, TransferLog, or FRCACustomLog directive is used as given for the name of the log file.



If *LogCycle Weekly* is specified, rollover will occur when QDAYOFWEEK is equal to \*SUN. However, if the system is not using the Gregorian calendar, this value may not be set correctly and the logs may not get rolled over as expected.

*LogCycle Hourly* is not valid if the log file is in the QDLS file system as that file system only supports 8 character file names and 3 character extensions.

### **LogLength:**

**Module:** core

**Syntax:** LogLength *number-of-bytes*

**Default:** LogLength 0

**Context:** server config, Not in Limit

**Override:** none

**Origin:** IBM

**Example:** LogLength 1000000

The LogLength directive limits the size of any defined log file. To prevent problems due to unbounded growth of log files, this directive can be used to set an maximum file-size for log files. If the file exceeds this size, no more information will be written to it until logs and alertable message HTP8433 will be sent to QSYSOPR. The server will automatically restart logging requests when the logs are rolled over to the next log cycle. This directive can be specified multiple times in the configuration file.

**Parameter:** *number-of-bytes*

- The *number-of-bytes* parameter is an integer value that sets the maximum size limit of the log file. When any defined log file ( those defined with CustomLog, TransferLog, FRCACustomLog, or ErrorLog directives) exceeds this value, no more information will be logged until log rollover occurs. An alertable message TCP7201 will be sent to QSYSOPR. A value of 0 means there is no limit. If 'LogCycle Off' is specified and a non-zero value is specified for LogLength, when the LogLength size is reached no more logging will be done (even on starts and restarts of the server instance) since the same file will be used every time.

### **Security notes:**

- Security could be compromised if the directory where log files are stored is writable by anyone other than the user that starts the server.
- If a program is used, then it will be run under the user who started httpd. Be sure that the program is secure.

### **LogLevel:**

**Module:** core

**Syntax:** LogLevel *level*

**Default:** LogLevel warn

**Context:** server config, virtual host, Not in Limit

**Override:** none

**Origin:** Apache

**Example:** LogLevel debug

The LogLevel directive adjusts the verbosity of the messages recorded in the error logs. See the "ErrorLog" on page 296 directive for more information.

**Parameter:** *level*

The following levels are available, in order of decreasing significance:

- If *emerg*, system is unusable messages ("Child cannot open lock file. Exiting.").

- If *alert*, action must be taken immediately messages ("getpwuid: couldn't determine user name from uid.").
- If *crit*, critical conditions messages ("Socket: Failed to get socket, exiting child.").
- If *error*, error conditions messages ("Premature end of script headers.").
- If *warn*, warning conditions messages ("Child process 1234 did not exit, sending another SIGHUP."). If *notice*, normal but significant conditions messages ("httpd: caught SIGBUS, attempting to dump core in...").
- If *info*, informational messages ("Server seems busy, (you may need to increase StartServers or Min/MaxSpareServers)...").
- If *debug*, debug-level messages ("Opening config file...").

When a particular level is specified, messages from all other levels of higher significance will be reported as well. For example, when LogLevel *info* is specified, then messages with log levels of *notice* and *warn* will also be posted. Using a level of at least *crit* is recommended.

### LogMaint:

**Module:** core

**Syntax:** LogMaint *path\_to\_file* *expire* *size\_limit*

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** IBM

**Example:** LogMaint logs/access\_log 10 2000000

**Note:** If this directive is not present, log maintenance is not performed. If the directive is present, all parameters are required. Values of 0 for *expire* and *size\_limit* have a special meaning of *no limit*. If a LogMaint directive with values of 0 for both *expire* and *size\_limits* specified, no log maintenance will be done on the specified file.

The LogMaint directive allows you perform log maintenance on the specified file and its derivatives. When log maintenance is performed on a file, it is purged from the system. Derivatives consist of either the *path\_to\_file* name provided, concatenated with the extension ".Qcyymmdd", or "Qcyymmdd" if the provided *path\_to\_file* value was a directory. LogCycle must be active in order to enable derivatives.

A separate LogMaint directive is required in the server configuration for each CustomLog or ErrorLog that requires log maintenance. The recommended way to configure maintenance is to match the path configured on the LogMaint directive with the path specified on the associated CustomLog or ErrorLog directive.

#### Parameter One: *path\_to\_file*

- The *path\_to\_file* value specifies the IFS-style path (for example, /QSYS.LIB/MYHTTP.LIB/MYLOGS.FILE) of the log file to be included in log maintenance. Refer to the "LogCycle" on page 313 directive for more information on log file names and extensions.

#### Parameter Two: *expire*

- The *expire* value specifies an integer value indicating the number of days before a log file expires. Files older than this value are to be removed. A value of 0 means the log file will never expire. The age of the error log file is determined by the file creation date (as reported by the operating system). The file name suffix, such as errorlog.Q100082213, is not used to determine the age of the file. Files that are currently open and active in the server instance will not be removed.

#### Parameter Three: *size\_limit*

- The *size\_limit* value specifies an integer value indicating the maximum aggregate size of log files with the name *path\_to\_file*. When the combined size of the log files exceeds this value in bytes, files are deleted starting with the oldest file. Eligible files are deleted until the collective size is less than or equal to the value specified on this directive. A value of 0 means there is no size limit. Note that it is possible for the aggregate size of log files to exceed the total *size\_limit*. This is possible due to the fact that the size of any open log files are not included in the *size\_limit* total. Users should take this into account when they are calculating a value for size limit, and when setting a maximum value for the LogLength directive.

If both *expire* and *size\_limit* are configured to non-zero values, the expired files are purged first. If the *size\_limits* still exceeded after expired files are purged, the server continues purging files (oldest files first) until the collective log size is equal to or less than the *size\_limit*.

**Note:** If invalid values are used for *expire* or *size\_limit*, an error message will be placed into the job log and the HTTP Server will not start.

The following example of log maintenance will be performed on the logs/access\_log file and its derivatives (see below for details). The files will expire after 10 days. In addition, if the total limit exceeds 2,000,000 bytes, log maintenance will be performed.

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log common
LogMaint logs/access_log 10 2000000
```

The following example of log maintenance will be performed on the /QSYS.LIB/MYHTTP.LIB/MYLOGS.FILE/Q\* files. The files will expire after 25 days and there is no total limit on the size of the files.

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog /QSYS.LIB/MYHTTP.LIB/MYLOGS.FILE common
LogMaint /QSYS.LIB/MYHTTP.LIB/MYLOGS.FILE 25 0
```

Only one occurrence of this directive can exist per server or virtual host container. If the directive occurs more than once, the last one specified in the server or virtual host is used.

**Note:** If LogCycle is configured to Off, then log maintenance is not performed.

### LogMaintHour:

**Module:** core

**Syntax:** LogMaintHour *variable*

**Default:** LogMaintHour 0

**Context:** server config, virtual host

**Override:** none

**Origin:** IBM

**Example:** LogMaintHour 3

The LogMaintHour directive may be used to control which hour of the day log maintenance occurs. The default is for log maintenance to occur at midnight. Log maintenance always occurs at the beginning of the hour. By using this directive, which hour of the day maintenance will occur can be controlled, to do maintenance in the early morning or in the evening after the normal work day is done.

### LogTime:

**Module:** core

**Syntax:** LogTime *LocalTime* | GMT

**Default:** LogTime LocalTime

**Context:** server config, virtual host, Not in Limit

**Override:** none  
**Origin:** IBM  
**Example:** LogTime GMT

The LogTime directive specifies whether your log should record entrees using local time or Greenwich Mean Time (GMT). This directive affects timestamps for log entries only.

**Parameter:** *LocalTime* | *GMT*

- *LocalTime* indicates the local time for log entry timestamps.
- *GMT* indicates the Greenwich Mean Time for log entry timestamp.

### **MaxKeepAliveRequests:**

**Module:** core  
**Syntax:** MaxKeepAliveRequests *number*  
**Default:** MaxKeepAliveRequests 100  
**Context:** server config, virtual host, Not in Limit  
**Override:** none  
**Origin:** Apache  
**Example:** MaxKeepAliveRequests 50

The MaxKeepAliveRequests directive limits the number of requests allowed per connection when “KeepAlive” on page 306 is on. If it is set to 0, unlimited requests will be allowed.

**Parameter:** *number*

- The *number* parameter specifies an integer value that limits the number of requests allowed per connection when KeepAlive is on.

See “KeepAlive” on page 306, and “KeepAliveTimeout” on page 306.

### **NameVirtualHost:**

**Module:** core  
**Syntax:** NameVirtualHost *address[:port]*  
**Default:** none  
**Context:** server config  
**Override:** none  
**Origin:** Apache  
**Example:** NameVirtualHost 10.1.1.1

The NameVirtualHost directive is a required directive if you want to configure name-based virtual hosts.

**Parameter:** *address*

- The *address* parameter consists of an IP address or hostname. Although address can be a hostname, it is recommended that you always use an IP address. For example:

```
NameVirtualHost 10.22.33.44
```

With the NameVirtualHost directive you specify the address to which your name-based virtual host names resolve. If you have multiple name-based hosts on multiple addresses, repeat the directive for each address.

**Note:** The “main server” and any `_default_` servers will never be served for a request to a NameVirtualHost IP Address (unless for some reason you specify NameVirtualHost but then do not define any VirtualHosts for that address).

Optionally you can specify a port number on which the name-based virtual hosts should be used, for example:

```
NameVirtualHost 10.22.33.44:8080
```

**Options:**

**Module:** core

**Syntax:** Options [+|-]*option* [[+|-]*option* ...]

**Default:** Options all

**Context:** server config, virtual host, directory (but not location), .htaccess

**Override:** none

**Origin:** Apache

**Example:** Options +Indexes +FollowSymLinks

The Options directive controls which server features are available in a particular directory.

**Parameter :** *option*

- The option parameter can be set to one or more of the following:

Option	Description
None	None of the extra features are enabled.
All	All options except for MultiViews. This is the default setting.
ExecCGI	Execution of CGI scripts is permitted.
Follow SymLinks	The server will follow symbolic links in this directory. <b>Note:</b> Even though the server follow the SymLink, it does not change the pathname used to match against <Directory> sections. This option gets ignored if set inside <Location> sections.
Includes	Server-side includes are permitted.
IncludesNOEXEC	Server-side includes are permitted, but the #exec command and #include of CGI scripts are disabled.
Indexes	If a URL which maps to a directory is requested and there is no DirectoryIndex (for example, index.html) in that directory, then the server will return a formatted listing of the directory.
MultiViews	Content negotiated MultiViews are allowed. See “Content negotiation for HTTP Server” on page 17 for more information.
SymLinksIfOwnerMatch	The server will only follow symbolic links for which the target file or directory is owned by the same user id as the link. <b>Note:</b> This option is ignored if set inside a <Location> sections.

Normally, if multiple *Options* could apply to a directory, then the most specific one is taken complete; the options are not merged. However if all the options on the Options directive are preceded by a + or - symbol, the options are merged. Any options preceded by a + are added to the options currently in force; any options preceded by a - are removed from the options currently in force.

For example, without any + and - symbols:

```

<Directory /web/docs>
  Options Indexes FollowSymLinks
</Directory>
<Directory /web/docs/spec>
  Options Includes
</Directory>

```

Then only Includes will be set for the /web/docs/spec directory. However if the second Options directive uses the + and - symbols:

```

<Directory /web/docs>
  Options Indexes FollowSymLinks
</Directory>
<Directory /web/docs/spec>
  Options +Includes -Indexes
</Directory>

```

Then the options FollowSymLinks and Includes are set for the /web/docs/spec directory.

**Note:** Using -IncludesNOEXEC or -Includes disables server-side includes completely regardless of the previous setting. The default in the absence of any other settings is All.

The *option* +IncludesNOEXEC can be used instead of +Includes. If the previous is specified, then the SSI Exec tag is not processed during SSI processing.

### ProfileToken:

**Module:** core

**Syntax:** ProfileToken *on* | *off*

**Default:** ProfileToken off

**Context:** directory

**Override:** AuthConfig

**Origin:** IBM

**Example:** ProfileToken on

The ProfileToken directive creates a 32-byte value called the ProfileToken. This token is used the same way as a userid/password combination to identify/authenticate a user, and prevents passing these values in the clear. The ProfileToken value can be used on any of the IBM i security APIs that accept a ProfileToken as input.

**Parameter:** *on* | *off*

- If *on* is specified, and basic authentication is performed successfully, the userid/password is passed in by the user (only an IBM i user) to generate a ProfileToken. A ProfileToken is not generated if this parameter is set to off, or if basic authentication was not successful.

The ProfileToken is accessible in a CGI program via the HTTP\_AS\_AUTH\_PROFILETKN environment variable. The HTTP\_AS\_AUTH\_PRFILETKN environment variable is not set if a ProfileToken is not generated.

The ProfileToken is accessible in HTTP Server modules via the headers section ( *r->headers\_in* field), which is an internal representation of the HTTP request structure. The profile token is stored as the AS\_Auth\_ProfileTkn header in the headers section. HTTP Server modules can then retrieve this ProfileToken and either use it, or pass it on to another application. The AS\_Auth\_ProfileTkn header is not created if a ProfileToken is not generated.

### ReceiveBufferSize:

**Module:** core

**Syntax:** ReceiveBufferSize *bytes*

**Default:** ReceiveBufferSize 0

**Context:** server config  
**Override:** none  
**Origin:** Apache

The `ReceiveBufferSize` directive can be used to control the TCP receive buffer size for the server. The server will set the TCP receive buffer size to the number of bytes specified.

**Parameter:** *bytes*

- The *bytes* value is an integer that must be set to 0 or a value that is greater or equal to 512 (in bytes). If 0 is specified, the server will use the default TCP receive buffer size that is configured for the IBM i server.

**Require:**

**Module:** core  
**Syntax:** `require entity-name entity entity...`  
**Default:** none  
**Context:** directory, .htaccess  
**Override:** AuthConfig  
**Origin:** Apache  
**Example:** `require group admin`

This directive selects which authenticated users can access a directory.

**Parameter:** *entity-name entity entity...*

- If *require user userid userid*, then only the named users can access the directory.
- If *require group groupname groupname*, then only users in the named groups can access the directory.
- If *require valid-user*, then all valid users can access the directory.

`Require` must be accompanied by `AuthName` and `AuthType` directives, and directives such as `PasswdFile` and `GroupFile` (to define users and groups) in order to work correctly. For example:

```
AuthType Basic
AuthName "Restricted Directory"
PasswdFile web/users
GroupFile /web/groups
require group admin
```

Access controls which are applied in this way are effective for all methods. This is what is normally desired. If you want to apply access controls only to specific methods, while leaving other methods unprotected, then place the `require` statement into a `<Limit>` section.

Access controls can be used in a named protection setup. To implement a named protection setup, place all of the access control directives in a file. Use the `Include` directive to include the file in your `<Directory>`, `<File>`, or `<Location>` context. This allows users that want to use the same type of protection setup within multiple contexts to add an `include` statement inside of each context.

**Note:** The *require valid-user* directive parameter should NOT be configured in the same context as any *require user* or *require group* directive parameters. The `require` directives are processed in order (from top to bottom) as they appear in the configuration file. Since *require valid-user* allows access to any authenticated user, the *require valid-user* directive parameter effectively overrides the presence of any *require user* or *require group* directives.

## RuleCaseSense:

**Module:** core

**Syntax:** RuleCaseSense *on* | *off*

**Default:** RuleCaseSense *off*

**Context:** server config

**Override:** none

**Origin:** IBM

**Example:** RuleCaseSense *on*

The RuleCaseSense directive is used to control how requested URLs are handled.

**Parameter:** *on* | *off*

- When *on* is specified, the URLs are treated as case sensitive. This means that an exact match with case is required.
- When *off* is specified, the URLs are treated as case insensitive. Paths on directives and requested URLs are internally converted to upper case before they are compared.

The default value for the case sensitivity is *off*. Rules that map the same value in different cases to different things, for example ABC to XYZ, will not work correctly when RuleCaseSense is *off*. This type of mapping is not recommended.

When using protection for any URL, it is recommended that you set this directive to *off*. This ensures that all variations in case for your URLs are protected. If you do not protect any of your site, then this directive can be either *on* or *off*. Setting it to *Off* allows your users to specify any case on their URLs and enables them to see your site.

If you use the QOpenSys file system, you will have to be careful to match the case of your file system in your directives. You will also need to be aware that case differences matter when serving files from the case sensitive file system. You should only turn this directive on when absolutely necessary.

RuleCaseSense affects the processing of the incoming URL in the "URL Fixup" and "URL translation" server phases. These phases manipulate the incoming URL and do not necessarily relate to other directives. RuleCaseSense affects the following directives: Alias, AliasMatch, RewriteBase, RewriteCond, RewriteMap, RewriteRule, ScriptAlias, and ScriptAliasMatch

## Satisfy:

**Module:** core

**Syntax:** Satisfy *any* | *all*

**Default:** Satisfy *all*

**Context:** directory, .htaccess

**Override:** AuthConfig

**Origin:** Modified

**Example:** Satisfy *any*

The Satisfy directive establishes access policy if both allow and require are used. The parameter can be either 'all' or 'any'. This directive is only useful if access to a particular area is being restricted by both username/password and client host address.

**Parameter:** *any* | *all*

- In this case, the default behavior *all* requires that the client passes the address access restriction and enters a valid username and password.



- With the *any* option, the client will be granted access if they either pass the host restriction or enter a valid username and password. This can be used to password restrict an area, but to let clients from particular addresses in without prompting for a password.

The Require directive has to indicate Satisfy is not required every time AuthType is used, but if "Satisfy Any" is used, then you must also use Allow, Require, AuthType AuthName and PasswdFile in order for the Satisfy to work correctly. For example:

```
Order allow,deny
Allow from All
Satisfy Any
AuthType Basic
AuthName "Realm can go here"
PasswdFile %%SYSTEM%%
Require valid-user
```

**Note:** If you are using SSL Authentication the satisfy directive should be set to any. The all option allows for SSL Authentication, and also authentication with userid and passwords. You do not want to use the Require directive if SSLClientAuth equals zero (0). In this case, the Satisfy directive should not be used with "Allow from All" and "SSLClientAuth 0".

See also mod\_access.

### SendBufferSize:

**Module:** core  
**Syntax:** SendBufferSize *bytes*  
**Default:** SendBufferSize 0  
**Context:** server config  
**Override:** none  
**Origin:** Apache  
**Example:** SendBufferSize 4000

The SendBufferSize directive tells the server to set the TCP buffer size to the number of specified bytes. The TCP send buffer size provides a limit on the number of outgoing bytes that are buffered by TCP. Once this limits reached, attempts to send additional bytes may result in the application blocking until the number of outgoing bytes buffered drops below this limit. The number of outgoing buffered bytes is decremented when the remote system acknowledges the sent data.

**Parameter:** *bytes*

- The *bytes* value is an integer that must be set to 0 or a value that is greater or equal to 512 (in bytes). If 0 is specified, the server will use the default TCP send buffer size that is configured for the IBM i server.

### SendFileMinSize:

**Module:** core  
**Syntax:** SendFileMinSize *bytes*  
**Default:** SendBufferSize 16000  
**Context:** server  
**Override:** none  
**Origin:** Apache  
**Example:** SendBufferSize 150

This directive specifies the minimum size of a file that is allowed to be sent via sendfile. SendFileMinSize is an IBM i specific directive. This directive may also limit the caching of local files when using the

CacheLocalFD. A file that is Cached using CacheLocalFD must be served using sendfile. Because of this, when using CacheLocalFD, a file is only cached when its size is greater than SendFileMinSize.

**Note:** Files larger than SendFileMinSize will not be cached dynamically.

### ServerAdmin:

**Module:** core

**Syntax:** ServerAdmin *email-address*

**Default:** none

**Context:** server config, Not in Limit

**Override:** none

**Origin:** Apache

**Example:** ServerAdmin www-admin@myserver.com

The ServerAdmin directive specifies the e-mail address to be used in trailing footer lines for hard coded error messages returned to clients. The specified value is used in hypertext link references generated by the server when "email" is specified for the "ServerSignature" on page 326 directive.

**Parameter:** *email-address*

- The *email-address* parameter specifies a valid email address.

For example,

```
ServerAdmin www-admin@server.ibm.com
```

**Note:** This setting is not used if ServerSignature is not set to "email", or for errors handled by custom error messaging (see "ErrorDocument" on page 294 for more details on custom error messaging).

### ServerAlias:

**Module:** core

**Syntax:** ServerAlias *host1* [*host2* ...]

**Default:** none

**Context:** virtual host

**Override:** none

**Origin:** Apache

**Example:** ServerAlias ibm.com \*.ibm.com

The ServerAlias directive allows servers to be accessible by more than one name. For example, HTTP Server might want to be accessible as `ibm.org`, or `ftp.ibm.org`, assuming the IP addresses pointed to the same server. In fact, one might want it so that all addresses at `ibm.org` were picked up by the server. This is possible with the ServerAlias directive placed inside the "<VirtualHost>" on page 332 section.

**Parameter:** *host*

- The *host* parameter specifies a hostname. Note that you can use '\*' and '?' as wildcard characters.

For example,

```
<VirtualHost 10.22.33.55>
  ServerAdmin webmaster@host.QIBM.com
  DocumentRoot /usr/web/host.QIBM.com
  ServerName host.QIBM.com
  ServerAlias ibm.com *.ibm.org
  ErrorLog logs/host.QIBM.com-error_log
  TransferLog logs/host.QIBM.com-access_log
</VirtualHost>
```

You may need `ServerAlias` if you are serving local users who do not always include the domain name. For example, if local users are familiar with typing "www" or "www.physics" then you will need to add `ServerAlias www www.physics`. It isn't possible for the server to know what domain the client uses for their name resolution because the client doesn't provide that information in the request.

The `ServerAlias` directive sets the alternate names for a host, for use with name-based virtual hosts.

If multiple occurrences of this directive are configured in a container, only the last occurrence is processed. All other occurrences are ignored.

#### **ServerName:**

**Module:** core

**Syntax:** `ServerName fully-qualified-domain-name [:port]`

**Default:** none

**Context:** server config, virtual host, Not in Limit

**Override:** none

**Origin:** Apache

**Example:** `ServerName www.example.com`

The `ServerName` directive sets the server hostname. This setting is used when creating redirection URLs. If it is not specified, the server attempts to deduce the server name from its own IP address; however, this may not work reliably or may not return the preferred hostname.

**Parameter:** *fully-qualified-domain-name*

- The *fully-qualified-domain-name* parameter sets the server hostname.

For example,

```
ServerName simple.example.com:80
<VirtualHost 10.1.2.3>
  ServerAdmin webmaster@host.QIBM.com
  DocumentRoot /usr/web/host.QIBM.com
  ServerName host.QIBM.com
  ErrorLog logs/host.QIBM.com-error_log
  TransferLog logs/host.QIBM.com-access_log
</VirtualHost>
```

This would be used if the canonical (main) name of the actual machine were `simple.example.com`. If you are using name-based virtual hosts, the `ServerName` inside a "`<VirtualHost>`" on page 332 section specifies what hostname must appear in the request's `Host:` header to match this virtual host.

This directive allows a port to be added to the server name. This allows an administrator to assign the canonical port at the same time that the canonical name is assigned. If no port is specified, HTTP Server implies port 80 for `http://` and port 443 for `https://` requests. This setting also specifies the server name used when trailing footer lines are added to hard coded error messages (see "`ServerSignature`" on page 326).

**Note:** TCP/IP must be properly configured to recognize all possible server host names.

See also "`UseCanonicalName`" on page 331, "`NameVirtualHost`" on page 318 and "`ServerAlias`" on page 324.

#### **ServerPath:**

**Module:** core

**Syntax:** `ServerPath pathname`

**Default:** none

**Context:** virtual host, Not in Limit  
**Override:** none  
**Origin:** Apache  
**Example:** ServerPath /sub1/

The ServerPath directive sets the legacy URL pathname for a host, for use with name-based virtual hosts.

**Parameter:** *pathname*

- The *pathname* parameter sets the legacy URL pathname for a host, for use with name-based virtual hosts.

For example, an HTTP server exists with two name-based virtual hosts. In order to match the correct virtual host a client must send the correct `Host:` header. Old HTTP/1.0 clients do not send such a header and the server has no clue what virtual host the client tried to reach (and serves the request from the primary virtual host). To provide as much backward compatibility as possible, create a primary virtual host that returns a single page containing links with an URL prefix to the name-based virtual hosts.

A request to the URL `http://www.sub1.domain.tld/sub1/` is always served from the sub1-virtual host. A request to the URL `http://www.sub1.domain.tld/` is only served from the sub1-virtual host if the client sent a correct `Host:` header. If no `Host:` header is sent, the client gets the information page from the primary host. Note that there is one exception: a request to `http://www.sub2.domain.tld/sub1/` is also served from the sub1-virtual host if the client did not send a `Host:` header. The RewriteRule directives are used to make sure that a client who sent a correct `Host:` header can use both URL variants (for example, with or without the URL prefix).

### ServerRoot:

**Module:** core  
**Syntax:** ServerRoot *directory-path*  
**Default:** none  
**Context:** server config  
**Override:** none  
**Origin:** Apache  
**Example:** ServerRoot /www/webserver

The ServerRoot directive sets the directory in which the server lives. Typically it will contain the subdirectories `conf/` and `logs/`. Relative paths for other configuration files are taken as relative to this directory.

The *directory-path* parameter must specify a path in either the root (`/`) or QOpenSys file system.

**Parameter:** *directory-path*

- The *directory-path* parameter sets the directory in which the server lives.

### ServerSignature:

**Module:** core  
**Syntax:** ServerSignature *on | off | email*  
**Default:** ServerSignature off  
**Context:** server config, virtual host, directory, .htaccess, Not in Limit  
**Override:** none  
**Origin:** Apache  
**Example:** ServerSignature on

The `ServerSignature` directive specifies if trailing footer lines are to be generated for hard coded error messages returned to clients. When requests pass through a chain of servers, this feature is useful to identify which server generated the error message. The default value is *off*.

**Parameter:** *on* | *off* | *email*

- If *on* is specified, trailing footer lines containing the server name and version information are added to hard coded error messages.
- If *off* is specified (the default), trailing footer lines are suppressed and only hard coded error messages are returned.
- If *email* is specified, trailing footer lines are added and look identical to those generated when *on* is specified, however the server name is also a hypertext link that references the server administrator's e-mail address.

The value used for server name is that specified by the "`ServerName`" on page 325 directive of the serving virtual host or server. The value used for version information is that specified by the "`ServerTokens`" directive. The value used for server administrator's e-mail address is that specified by the "`ServerAdmin`" on page 324 directive.

For example, the value used for server name is that specified by the `ServerName` directive of the serving virtual host or server. The value used for version information is that specified by the `ServerTokens` directive. The value used for server administrator's e-mail address is that specified by the `ServerAdmin` directive.

For example,

```
ServerAdmin www-admin@myserver.com
ServerSignature email
```

**Note:** This setting is not used for errors handled by custom error messaging (see "`ErrorDocument`" on page 294 for more details on custom error messaging).

### **ServerTokens:**

**Module:** core

**Syntax:** `ServerTokens` *Major* | *Minor* | *Minimal* | *OS* | *Full* | *Prod*

**Default:** `ServerTokens` *Prod*

**Context:** server config

**Override:** none

**Origin:** Apache

**Example:** `ServerToken` *Full*

The `ServerTokens` directive specifies which form of the `Server:` header value is included in response headers sent to clients. The value may consist of a minimal description of the server, a description with a generic OS-type included, a description that includes information about compiled-in modules, or a simple product description.

**Parameter:** *Major* | *Minor* | *Minimal* | *OS* | *Full* | *Prod*

- If *Major* is specified, the server sends: "Server : Apache/2"
- If *Minor* is specified, the server sends: "Server : Apache/2.2"
- If *Minimal* is specified, the server sends: "Server: Apache"
- If *OS* is specified, the server sends: "Server: Apache (i5)"
- If *Full* is specified, the server sends: "Server: Apache (i5) MymMod/1.2"
- If *Prod* is specified, the server sends: "Server: Apache"

This setting also specifies the version information used when trailing footer lines are added to hard coded error messages (see “ServerSignature” on page 326).

**Note:** This setting applies to the entire server, and cannot be enabled or disabled on a virtualhost-by-virtualhost basis.

### ServerUserID:

**Module:** core

**Syntax:** ServerUserID *user\_profile*

**Default:** ServerUserID QTMHHTTP

**Context:** All

**Override:** AuthConfig

**Origin:** IBM

**Example:** ServerUserID webmaster

The ServerUserID directive specifies the user profile that the HTTP Server will run under. This directive tells what user profile to use when starting the worker threads under the child process.

**Parameter:** *user\_profile*

- The *user\_profile* parameter must be a valid user profile. This profile must be authorized to all the directories, files, and other server resources accessed by the Web server unless the server is configured to swap to another profile for specific requests or directories.

This directive is now valid in all contexts. If *userid* is set and authentication is performed in a context, and the *UserID* value is set to `%%SERVER%%`, then *ServerUserID* will be used for that context. The *ServerUserID* is inherited through all contexts unlike *UserID*. If authentication is performed and the *UserID* directive is set to something other than `%%SERVER%%`, then *ServerUserID* is overridden by the *UserID*. If authentication is not performed at all, then *ServerUserID* is used from the correct context. This allows for specific and unique user id security models for separate virtual hosts, locations, directories, files or `.htaccess`, allowing for more security control (specific access versus "global") over the resources served by the HTTP Server.

**Note:** To start the server you must have authority to the specified profile.

See also “*UserID*” on page 234.

### SetHandler:

**Module:** core

**Syntax:** SetHandler *handler-name*

**Default:** none

**Context:** directory, `.htaccess`

**Override:** none

**Origin:** Apache

**Example:** SetHandler `imap-file`

The SetHandler directive forces all matching files to be parsed through the handler given by *handler-name*. This happens when it is placed into an `.htaccess` file or a “<Directory>” on page 291 or “<Location>” on page 312 section. For example, if you had a directory you wanted to be parsed entirely as `imagemap` rule files, regardless of extension, you might put the following into an `.htaccess` file in that directory: See “Handler for HTTP Server” on page 200 for more information

```
SetHandler imap-file
```

**Parameter:** *handler-name*

- The *handler-name* parameter is the name of the handler that will parse files in this directory.

**Note:** The core directives ForceType and SetHandler are used to associate all the files in a given container (<Location>, <Directory>, or <Files>) with a particular MIME-type or handler. These settings override any filename extension mappings defined in mod\_mime.

### SetInputFilter:

**Module:** core  
**Syntax:** SetInputFilter *filter* [*filter* ...]  
**Default:** none  
**Context:** directory, .htaccess  
**Override:** none  
**Origin:** Apache  
**Example:** SetInputFilter gzip

The SetInputFilter directive sets the filters that process client requests when they are received by the server. Parameter One: filter

#### Parameter: *filter*

- The *filter* parameter sets the filters that process client requests when they are received by the server.

For example,

```
<Directory /www/data/>  
  SetInputFilter gzip  
</Directory>
```

If more than one filter is specified, they must be separated by semicolons in the order in which they should process the content.

The order of the arguments determines the order in which the filters process the content. The first filter in the list processes content first, followed by the second in the list, and so on until all filters in the list have processed the content.

See the Apache HTTP Server Version 2.0 Filters  documentation for more information regarding filters.

### SetOutputFilter:

**Module:** core  
**Syntax:** SetOutputFilter *filter* [*filter* ...]  
**Default:** none  
**Context:** directory, .htaccess  
**Override:** none  
**Origin:** Apache  
**Example:** SetOutputFilter INCLUDES

The SetOutputFilter directive sets the filters that process responses from the server before they are sent to the client.

#### Parameter: *filter*

- The *filter* parameter sets the filters that process responses from the server before they are sent to the client.

For example, the following configuration will process all files in the /www/data/ directory for server-side includes:

```
<Directory /www/data/>  
  SetOutputFilter INCLUDES  
</Directory>
```

If more than one filter is specified, they must be separated by semicolons in the order in which they should process the content.

The order of the arguments determines the order in which the filters process the content. The first filter in the list processes content first, followed by the second in the list, and so on until all filters in the list have processed the content.

See the Apache HTTP Server Version 2.0 Filters  documentation for more information regarding filters.

### ThreadsPerChild:

**Module:** core

**Syntax:** ThreadsPerChild *number*

**Default:** Global HTTP Server setting for maximum number of servers.

**Context:** server config

**Override:** none

**Origin:** Apache

**Example:** ThreadsPerChild 20

Use this directive to specify the maximum number of threads per server child process. If the directive is not specified, the global HTTP Server setting for maximum number of servers is used. The shipped value is 40. You can view and change global HTTP Server settings using the Change HTTP Attributes (CHGHTTPPA) command.

**Parameter:** *number*

- The *number* value is an integer value that specifies the maximum number of threads per server child process.

### TimeOut:

**Module:** core

**Syntax:** TimeOut *number*

**Default:** TimeOut 300

**Context:** server config, Not in Limit

**Override:** none

**Origin:** Apache

**Example:** TimeOut 500

The TimeOut directive defines the amount of time (in seconds) HTTP Server will wait for:

1. The amount of time between receipt of TCP packets on a request.
2. The amount of time between ACKs on transmissions of TCP packets in responses.

**Parameter:** *number*

- The *number* value is an integer value that specifies defines the amount of time (in seconds) HTTP Server will wait.



## | **TraceEnable:**

| **Module:** core  
| **Syntax:** TraceEnable *on* | *off* | *extended*  
| **Default:** TraceEnable off  
| **Context:** server config  
| **Override:** none  
| **Origin:** Apache  
| **Example:** TraceEnable on  
|

| This directive overrides the behavior of TRACE for both the core server and mod\_proxy. When "on" is specified for the TraceEnable directive, TRACE requests are permitted per RFC 2616, which disallows any request body to accompany the request. Setting the TraceEnable directive to "off" causes the core server and mod\_proxy to return a 405 (Method not allowed) error response to the client.

| For testing and diagnostic purposes only, request bodies may be allowed by specifying "extended" for the TraceEnable directive. The core will restrict the request body to 64 KB (plus 8 KB for chunk headers if Transfer-Encoding: chunked is used). The core will reflect the full headers and all chunk headers with the response body. As a proxy server, the request body is not restricted to 64 KB.

| **Parameter:** *on* | *off* | *extended*

- | • A value of *on* permits TRACE requests.
- | • A value of *off* causes TRACE requests to return a 405 (Method not allowed) error response to the client.
- | • A value of *extended* permits TRACE requests that are not compliant for testing and diagnostic purposes.

## **UseCanonicalName:**

**Module:** core  
**Syntax:** UseCanonicalName *on* | *off* | *DNS*  
**Default:** UseCanonicalName on  
**Context:** server config, virtual host, directory, Not in Limit  
**Override:** Options  
**Origin:** Apache  
**Example:** UseCanonicalName off

In many situations HTTP Server has to construct a self-referential URL. That is, a URL that refers back to the same server.

**Parameter:** *on* | *off* | *DNS*

- When set to *on*, HTTP Server will use the ServerName directive to construct a canonical name for the server. This name is used in all self-referential URLs, and for the values of SERVER\_NAME and SERVER\_PORT environment variables in CGIs.
- When set to *off*, HTTP Server will form self-referential URLs using the hostname and port supplied by the client if any are supplied (otherwise it will use the canonical name). These values are the same that are used to implement name based virtual hosts, and are available with the same clients. The CGI variables SERVER\_NAME and SERVER\_PORT will be constructed from the client supplied values as well.

An example where this may be useful is on an intranet server where you have users connecting to the machine using short names such as www. You'll notice that if the users type a shortname, and a URL which is a directory, such as http://www/splat, without the trailing slash then HTTP Server will redirect them to http://www.domain.com/splat/. If you have

authentication enabled, this will cause the user to have to reauthenticate twice (once for `www` and once again for `www.domain.com`). But if `UseCanonicalName` is set *off*, then HTTP Server will redirect to `http://www/splat/`.

- The `DNS` setting is intended for use with mass IP-based virtual hosting to support clients that do not provide a `Host:` header. With this option HTTP Server does a reverse DNS lookup on the server IP address that the client connected to in order to work out self-referential URLs.

**Important:** If CGIs make assumptions about the values of `SERVER_NAME` they may be broken by this option. The client is essentially free to give whatever value they want as a hostname. But if the CGI is only using `SERVER_NAME` to construct self-referential URLs then it should be fine.

See also “`ServerName`” on page 325.

### **UseShutdown:**

**Module:** core

**Syntax:** `UseShutdown On | Off`

**Default:** `UseShutdown Off`

**Context:** server config

**Override:** none

**Origin:** Apache

**Example:** `UseShutdown On`

This directive instructs the HTTP Server to use shutdown on the socket connections.

### **<VirtualHost>:**

**Module:** core

**Syntax:** `<VirtualHost addr[:port] [addr[:port]]...> ... </VirtualHost>`

**Default:** none

**Context:** server config, Not in Limit

**Override:** none

**Origin:** Apache

**Example 1: Using an IPv4 address:**

```
<VirtualHost 10.1.2.3>
ServerAdmin webmaster@host.foo.com
DocumentRoot /www/docs/host.foo.com
ServerName host.foo.com
ErrorLog logs/host.foo.com-error_log
TransferLog logs/host.foo.com-access_log
</VirtualHost>
```

**Example 2: Using an IPv6 address:**

```
<VirtualHost [2001:db8::a00:20ff:fea7:ccea]>
ServerAdmin webmaster@host.example.com
DocumentRoot /www/docs/host.example.com
ServerName host.example.com
ErrorLog logs/host.example.com-error_log
TransferLog logs/host.example.com-access_log
</VirtualHost>
```

The term Virtual Host refers to the practice of running more than one web site (such as `www.company1.com` and `www.company2.com`) on a single machine. Virtual hosts can be “IP-based”, meaning that you have a different IP address for every web site, or “name-based”, meaning that you have multiple names running on each IP address. The fact that they are running on the same physical server is not apparent to the end user.

**Parameter One:** *address*

- The *address* parameter specifies a fully qualified IP address or hostname.

**Parameter Two:** *port*

- The *port* parameter specifies a port number. This parameter is optional. If a port is not specified the server port will be used.

<VirtualHost> and </VirtualHost> are used to enclose directives that apply only to a particular virtual host. Any directive that is allowed in a virtual host context may be used. When the server receives a document request on a particular virtual host, it uses the configuration directives enclosed in the <VirtualHost> section. The *address* parameter may be one of the following:

- The IP address of the virtual host.
- A fully qualified domain name for the IP address of the virtual host.
- The character \*, which is used only in combination with NameVirtualHost \* to match all IP addresses.
- The string *\_default\_*, which is used only with IP virtual hosting to catch unmatched IP addresses.

Each Virtual Host must correspond to a different IP address, different port number or a different host name for the server. In the former case, the server machine must be configured to accept IP packets for multiple addresses.

IPv6 addresses must be specified in square brackets because the optional port number could not be determined otherwise.

**Note:** The use of <VirtualHost> does not affect what addresses the server listens on. You may need to ensure that HTTP Server is listening on the correct addresses using the Listen directive.

When using IP-based virtual hosting, the special name *\_default\_* can be specified in which case this virtual host will match any IP address that is not explicitly listed in another virtual host. In the absence of any *\_default\_* virtual host the "main" server config, consisting of all those definitions outside any VirtualHost section, is used when no IP-match occurs.

**Note:** Any IP address that matches a NameVirtualHost directive will use neither the "main" server config nor the *\_default\_* virtual host. See the name-based virtual hosting documentation for further details.

You can specify a *:port* to change the port that is matched. If unspecified then it defaults to the same port as the most recent Listen statement of the main server. You may also specify *:\** to match all ports on that address, which is recommended when used with *\_default\_*.

### **Name-based vs IP-based Virtual Hosts**

IP-based virtual hosts use the IP address of the connection to determine the correct virtual host to serve. Therefore you need to have a separate IP address for each host. With name-based virtual hosting, the server relies on the client to report the hostname as part of the HTTP headers. Using this technique, many different hosts can share the same IP address. Name-based virtual hosting is usually simpler, since you need only configure your DNS server to map each hostname to the correct IP address and then configure the Apache HTTP Server to recognize the different hostnames. Name-based virtual hosting also eases the demand for scarce IP addresses. Therefore you should use name-based virtual hosting unless there is a specific reason to choose IP-based virtual hosting. Some reasons why you might consider using IP-based virtual hosting:

- Some ancient clients are not compatible with name-based virtual hosting. For name-based virtual hosting to work, the client must send the HTTP Host header. This is required by HTTP/1.1, and is implemented by all modern HTTP/1.0 browsers as an extension. If you need to support obsolete clients and still use name-based virtual hosting, a possible technique is discussed at the end of this document.

- Name-based virtual hosting cannot be used with SSL secure servers because of the nature of the SSL protocol.
- Some operating systems and network equipment implement bandwidth management techniques that cannot differentiate between hosts unless they are on separate IP addresses.

## Using Name-based Virtual Hosts

To use name-based virtual hosting, you must designate the IP address (and possibly port) on the server that will be accepting requests for the hosts. This is configured using the `NameVirtualHost` directive. In the normal case where any and all IP addresses on the server should be used, you can use `*` as the argument to `NameVirtualHost`. If you're planning to use multiple ports (e.g. running SSL) you should add a `Port` to the argument, such as `*:80`. Note that mentioning an IP address in a `NameVirtualHost` directive does not automatically make the server listen to that IP address. In addition, any IP address specified here must be associated with a network interface on the server.

The next step is to create a `<VirtualHost>` each different host that you would like to serve. The argument to the `<VirtualHost>` directive should be the same as the argument to the `NameVirtualHost` directive (ie, an IP address, or `*` for all addresses). Inside each `<VirtualHost>` block, you will need at minimum a `ServerName` directive to designate which host is served and a `DocumentRoot` directive to show where in the filesystem the content for that host lives.

## Main host goes away

If you are adding virtual hosts to an existing web server, you must also create a `<VirtualHost>` block for the existing host. The `ServerName` and `DocumentRoot` included in this virtual host should be the same as the global `ServerName` and `DocumentRoot`. List this virtual host first in the configuration file so that it will act as the default host. For example, suppose that you are serving the domain `www.domain.tld` and you wish to add the virtual host `www.otherdomain.tld`, which points at the same IP address. Then you simply add the following to `httpd.conf`:

```
NameVirtualHost *:80
<VirtualHost *:80>
ServerName www.domain.tld
ServerAlias domain.tld *.domain.tld
DocumentRoot /www/domain
</VirtualHost>

<VirtualHost *:80>
ServerName www.otherdomain.tld
DocumentRoot /www/otherdomain
</VirtualHost>
```

You can alternatively specify an explicit IP address in place of the `*` in both the `NameVirtualHost` and `<VirtualHost>` directives. For example, you might want to do this in order to run some name-based virtual hosts on one IP address, and either IP-based, or another set of name-based virtual hosts on another address.

Many servers want to be accessible by more than one name. This is possible with the `ServerAlias` directive, placed inside the `<VirtualHost>` section. For example in the first `<VirtualHost>` block above, the `ServerAlias` directive indicates that the listed names are other names which people can use to see that same web site:

```
ServerAlias domain.tld *.domain.tld
```

Requests for all hosts in the `domain.tld` domain will be served by the `www.domain.tld` virtual host. The wildcard characters `*` and `?` can be used to match names. Of course, you can't just make up names and place them in `ServerName` or `ServerAlias`. You must first have your DNS server properly configured to map those names to an IP address associated with your server. Finally, you can fine-tune the configuration of the virtual hosts by placing other directives inside the `<VirtualHost>` containers. Most

directives can be placed in these containers and will then change the configuration only of the relevant virtual host. To find out if a particular directive is allowed, check the Context of the directive. Configuration directives set in the main server context (outside any <VirtualHost> container) will be used only if they are not overridden by the virtual host settings. Now when a request arrives, the server will first check if it is using an IP address that matches the NameVirtualHost. If it is, then it will look at each <VirtualHost> section with a matching IP address and try to find one where the ServerName or ServerAlias matches the requested hostname. If it finds one, then it uses the configuration for that server. If no matching virtual host is found, then the first listed virtual host that matches the IP address will be used.

As a consequence, the first listed virtual host is the default virtual host. The DocumentRoot from the main server will never be used when an IP address matches the NameVirtualHost directive. If you would like to have a special configuration for requests that do not match any particular virtual host, simply put that configuration in a <VirtualHost> container and list it first in the configuration file.

## Module mod\_dav

Module mod\_dav supports directives for the IBM HTTP Server for i Web server.

### Summary

This module provides class 1 and class 2 WebDAV (Web-based Distributed Authoring and Versioning) functionality for HTTP Server. This extension to the HTTP protocol allows creating, moving, copying, and deleting resources and collections on a remote web server.

In order for WebDAV to function, you have to have your LoadModules, Dav provider, and either DavLockDB or DavQsysLockDB (depending on your provider) in your configuration file. If any of these elements are missing, your server will not start.

To use DAV at all, your configuration file must include:

```
LoadModule dav_module /QSYS.LIB/QHTTPSVR.LIB/QZSRDAV.SRVPGM
```

To use DAV for root, QOpenSys, or other UNIX-like filesystems, in addition to the above, your configuration file must include:

```
LoadModule dav_fs_module /QSYS.LIB/QHTTPSVR.LIB/QZSRDAVF.SRVPGM
```

To use DAV in QSYS, your configuration file must include:

```
LoadModule dav_qsys_module /QSYS.LIB/QHTTPSVR.LIB/QZSRDAVQS.SRVPGM
```

**Note:** You'll need two LoadModules to use DAV. If you want to DAV-enable both IFS and QSYS, you'll need three LoadModules.

### Directives

- "Dav"
- "DavDepthInfinity" on page 337
- "DavLockDB" on page 337
- "DavMinTimeout" on page 338
- "DavQsysLockDB" on page 338

### Dav:

**Module:** mod\_dav

**Syntax:** Dav *on* | *off* | [*provider name*]

**Default:** Dav *off*

**Context:** directory

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule dav\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRDAV.SRVPGM

**Example:** Dav on

The Dav directive enables the WebDAV HTTP methods for the given container. You may want to add a <Limit> clause inside the location directive to limit access to Dav-enabled locations.

**Parameter:** *on* | *off* | [*provider name*]

- When *on* is specified, WebDAV HTTP methods are enabled for the given container, using the default provider "filesystem".
- When *off* is specified, WebDAV HTTP methods are disabled for the given container.
- The optional *provider name* parameter is used to specify the Dav provider for a directory or location. There are no Server restrictions on the number or types of characters in the provider name. The provider name used on the Dav directive is case sensitive.

The values on and off are not case sensitive.

#### Example 1:

```
DavLockDB /tmp/DavLock
LoadModule dav_module /qsys.lib/qhttpsvr.lib/qzsrдав.srvpgm
LoadModule dav_fs_module /qsys.lib/qhttpsvr.lib/qzsrдавf.srvpgm
<Location /foo>
  Dav on
</Location>
```

#### Example 2:

```
DavQsysLockDB mylib/DavLock
LoadModule dav_module /qsys.lib/qhttpsvr.lib/qzsrдав.srvpgm
LoadModule dav_qsys_module /qsys.lib/qhttpsvr.lib/qzsrдавqs.srvpgm

<Directory /qsys.lib/webserver.lib*>
  Dav qsys
</Directory>
```

If you specify "Dav on" in a directory, you will get the default provider "filesystem".

The Dav directive does not override like other directory-scoped directives. You cannot turn Dav on in one directory, and then turn it off in a sub-directory. You also cannot change providers in a sub-directory. You will receive runtime errors if this happens. The following examples are invalid and will cause the HTTP Server to generate a runtime error:

```
<Directory />
  AllowOverride None
  Order Deny,Allow
  Deny From all
  Dav filesystem
  </Files>
  Dav off
  </Files>
</Directory>
```

Another invalid example:

```
<Directory /www/parentDirectory>
  Dav filesystem
</Directory>
<Directory /www/parentDirectory/childDirectory>
  Dav off
</Directory>
```

**Note:** If you want to Dav-enable file systems other than root or QOpenSys, you will have to specify your provider's name on the directive to get the desired behavior. As the server is shipped, the only valid provider names are "filesystem" and "qsys". Filesystem supports root, QOpenSys (and other UNIX-like file systems); qsys supports QSYS objects.

### DavDepthInfinity:

**Module:** mod\_dav

**Syntax:** DavDepthInfinity *on* | *off*

**Default:** DavDepthInfinity off

**Context:** server config, virtual host, directory

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule dav\_module /QSYS.LIB/QHTTSPVR.LIB/QZSRDAV.SRVPGM

**Example:** DavDepthInfinity on

The DavDepthInfinity directive allows the processing of PROPFIND requests containing the header 'Depth: Infinity'. Because this type of request could constitute a denial-of-service attack, by default it is not allowed.

**Parameter:** *on* | *off*

- When *on* is specified, processing of PROPFIND requests containing the header 'Depth: Infinity' is allowed.
- When *off* is specified, processing of PROPFIND requests containing the header 'Depth: Infinity' is not allowed.

### DavLockDB:

**Module:** mod\_dav

**Syntax:** DavLockDB *filename*

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule dav\_fs\_module /QSYS.LIB/QHTTSPVR.LIB/QZSRDAV.SRVPGM

**Example:** DavLockDB /tmp/DavLock

The DavLockDB directive specifies the full path to the lock database, excluding an extension. The default (file system) implementation of mod\_dav uses a SDBM database to track user locks.

**Parameter:** *filename*

- The *filename* parameter specifies the full path to the lock database, excluding an extension.

This directive is required if you are using Dav with the default (filesystem) provider. For example,  
DavLockDB /tmp/DavLock

## DavMinTimeout:

**Module:** mod\_dav

**Syntax:** DavMinTimeout *seconds*

**Default:** DavMinTimeout 0

**Context:** server config, virtual host, directory

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule dav\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRDAV.SRVPGM

**Example:** DavMinTimeout 600

The DavMinTimeout directive specifies, in seconds, the minimum lock timeout to return to a client. Microsoft Web Folders defaults to a timeout of 120 seconds; the DavMinTimeout can override this to a higher value (like 600 seconds) to reduce the chance of the client losing the lock due to network latency.

When a client requests a DAV resource lock, it can also specify a time when the lock will be automatically removed by the server. This value is only a request, and the server can ignore it or inform the client of an arbitrary value. The maximum value for minutes is 166; the maximum value for seconds is 9999.

**Parameter:** *seconds*

- The *seconds* parameter is any integer value from 0 to 9999.

## DavQsysLockDB:

**Module:** mod\_dav

**Syntax:** DAVQsysLockDB *library/filename*

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** Modified

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule dav\_qsys\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRDAVQS.SRVPGM

**Example:** DAVQsysLockDB mylib/LockDB

The DAVQsysLockDB directive specifies the library qualified database file that the QSYS repository manager uses to track user locks of QSYS resources. The library must exist. The names of the library and file must follow the QSYS file system naming rules.

**Parameter:** *library/filename*


- The *library/filename* parameter specifies the library qualified database file that the QSYS repository manager uses to track user locks of QSYS resources.

## Module mod\_deflate

Module mod\_deflate supports directives for the IBM HTTP Server for i Web server.

### Summary

Module mod\_deflate specifies compression and decompression functions using filters, MIME types, environment variables, and HTTP responses. Compressed output is transferred to requesting client browsers at a higher rate of speed than output that is not compressed. Compression and decompression is implemented by the DEFLATE filter, located in module mod\_deflate. See Apache HTTP Server Version

2.2 Documentation  for additional information and examples on configuring the Apache server to use compression.



## Directives

- “DeflateBufferSize”
- “DeflateCompressionLevel”
- “DeflateFilterNote”
- “DeflateMemLevel” on page 340
- “DeflateWindowSize” on page 340

### DeflateBufferSize:

**Module:** mod\_deflate

**Syntax:** DeflateBufferSize *value*

**Default:** DeflateBufferSize 8096

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: deflate\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** DeflateBufferSize 8096

The DeflateBufferSize directive specifies the size of the fragments that zlib should compress at one time.

**Parameter:** *value*

- The *value* parameter specifies the size, in bytes, of the fragments that zlib should compress at one time.

### DeflateCompressionLevel:

**Module:** mod\_deflate

**Syntax:** DeflateCompressionLevel *value*

**Default:** DeflateCompressionLevel 6

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: deflate\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** DeflateCompressionLevel 5

The DeflateCompressionLevel directive specifies what level of compression should be used.

**Parameter:** *value*

- The *value* parameter value specifies the level of compression. The higher the value, the greater the compression.

**Note:** Higher compression levels require additional CPU time.

### DeflateFilterNote:

**Module:** mod\_deflate

**Syntax:** DeflateFilterNote [*type*] *notename*

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: deflate\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** DeflateFilterNote ratio

**Example:** DeflateFilterNote Ratio ratio

**Example:** DeflateFilterNote Input input

**Example:** DeflateFilterNote input input

The DeflateFilterNote directive specifies that a note about compression ratios should be attached to the request. The note is used for statistical purposes by adding a value to your access log.

**Parameter One:** *type*

- The *type* parameter value specifies what type of data is added to the note for logging. The parameter value is not case-sensitive. Possible values include:

**Input** Store the byte count of the filter's input stream in the note.

**Output**

Store the byte count of the filter's output stream in the note.

**Ratio** Store the compression ratio (output/input \* 100) in the note. This is the default, if the type argument is omitted.

**Parameter Two:** *notename*

- The *notename* parameter value specifies the note name entered in the log. The *notename* value is not required to match the *type* value. Blank characters are not valid.

**Example: accurate logging**

```
DeflateFilterNote Input instream
DeflateFilterNote Output outstream
DeflateFilterNote Ratio ratio
```

```
LogFormat "%r" %{outstream}n/%{instream}n (%{ratio}n%)" deflate
CustomLog logs/deflate_log deflate
```

**DeflateMemLevel:**

**Module:** mod\_deflate

**Syntax:** DeflateMemLevel *value*

**Default:** DeflateMemLevel 9

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: deflate\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** DeflateMemLevel 8

The DeflateMemLevel directive specifies how much memory should be used for zlib for compression.

**Parameter:** *value*

- The *value* parameter value specifies how much memory should be used for zlib compression. Each value is equal to 16K. For example, a value of 1 equates to 16K, while a value of 8 equates to 128K.

**DeflateWindowSize:**

**Module:** mod\_deflate

**Syntax:** DeflateWindowSize *value*

**Default:** DeflateWindowSize 15

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: deflate\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** DeflateWindowSize 14

The DeflateWindowSize directive specifies the zlib compression window size.

**Parameter:** *value*

- The *value* parameter value specifies the level of compression window size. The higher the value, the greater the compression window size.

**Note:** Higher compression levels require additional CPU time.

## Module mod\_dir

Module mod\_dir supports directives for the IBM HTTP Server for i Web server.

### Summary

The module mod\_dir provides "trailing slash" redirects and serving directory index files. The index of a directory can come from one of two sources:

- A file written by the user, typically called index.html. The name of this file is set by the DirectoryIndex directive. This directive is controlled by module mod\_dir.
- A list generated by the server through mod\_auto\_index. See mod\_auto\_index for more information.

The two functions are separated so you can completely remove (or replace) automatic index generation.

By default, a trailing slash ( '/') redirect is issued when the server receives a request for a URL `http://servername/QIBM/dirname` where `dirname` is a directory. Directories require a trailing slash, so mod\_dir issues a redirect to `http://servername/QIBM/dirname/`.

The AlwaysDirectoryIndex directive controls how the server will respond to directory requests.

### Directives

- "AlwaysDirectoryIndex"
- "DirectoryIndex" on page 342
- "DirectorySlash" on page 342

### AlwaysDirectoryIndex:

**Module:** mod\_dir

**Syntax:** AlwaysDirectoryIndex *on* | *off*

**Default:** Always DirectoryIndex on

**Context:** server config, virtual host, directory, .htaccess

**Override:** Indexes

**Origin:** IBM

**Example:** AlwaysDirectoryIndex off

The AlwaysDirectoryIndex directive specifies if you want the server to always handle directory requests by first searching the directory for an index (Welcome) file.

**Parameter:** *on* | *off*

- The *on* parameter sets the server to always search the directory for an index file, regardless of whether a trailing slash (/) exists in the URL. The `DirectoryIndex` directive specifies the names of the files that the server recognizes as index (Welcome) files.
- The *off* parameter sets the server to first check the last character of requests that refer to IBM i files, QDLS folders, or integrated file system directories for the slash (/) character. If the directory request ends with a slash, the server searches the directory for an index file. If the directory request does not end with a slash, the server attempts to return a directory listing (rather than doing a "trailing slash" redirect).

If the server does not find an index file, or `AlwaysWelcome` is set to `off` and the directory request does not end in a slash, "Options" on page 319 controls whether or not the server responds to the request with a directory listing.

### **DirectoryIndex:**

**Module:** `mod_dir`

**Syntax:** `DirectoryIndex local-url [[local-URL ...]`

**Default:** `DirectoryIndex index.html`

**Context:** server config, virtual host, directory, `.htaccess`

**Override:** Indexes

**Origin:** Apache

**Example:** `DirectoryIndex bob.html index.html`

The `DirectoryIndex` directive sets the list of resources to look for, when the client requests an index of the directory by specifying a / at the end of the a directory name. Local-URL is the (%-encoded) URL of a document on the server relative to the requested directory; it is usually the name of a file in the directory. Several URLs may be given, in which case the server will return the first one that it finds. If none of the resources exist and the `Indexes` option is set, the server will generate its own listing of the directory.

**Parameter:** *local-url*

- The *local-url* parameter is the (%-encoded) URL of a document on the server relative to the requested directory; it is usually the name of a file in the directory. For example:

```
DirectoryIndex index.html
```

A request for `http://myserver/docs/` would return `http://myserver/docs/index.html` if it exists, or it would list the directory if it did not exist.

The documents do not need to be relative to the directory. For example:

```
DirectoryIndex index.html index.txt /cgi-bin/index.pl
```

This would cause the CGI script `/cgi-bin/index.pl` to be run if neither `index.html` or `index.txt` existed in a directory. This same idea will also work for `QSYS.LIB` files. For example, if the directory index is stored in `/QSYS.LIB/MYLIB.LIB/MYFILE.FILE/INDEX.MBR`, you would need to specify **DirectoryIndex Index.mbr**.

This directive may be configured multiple times in a container. The directives are processed from the first to the last occurrence.

### **DirectorySlash:**

**Module:** `mod_dir`

**Syntax:** `DirectorySlash on | off`

**Default:** `DirectorySlash on`

**Context:** Server, Virtual Host, Directory, `.htaccess`

**Override:** Indexes

**Origin:** Apache

The `DirectorySlash` directive determines, whether `mod_dir` should fixup URLs pointing to a directory or not. Typically if a user requests a resource without a trailing slash, which points to a directory, `mod_dir` redirects him to the same resource, but with trailing slash for the following reasons:

- The user is finally requesting the canonical URL of the resource
- The directive `mod_autoindex` works correctly. Since `mod_autoindex` doesn't emit the path in the link, it would point to the wrong path.
- The `DirectoryIndex` directive will be evaluated only for directories requested with trailing slash.
- The relative URL references inside HTML pages will work correctly.

If you don't want this effect and the reasons above don't apply to you, you can turn off the redirect with the following:

```
# see security warning below!
<Location /some/path>
DirectorySlash Off
SetHandler some-handler
</Location>
```

**Security Warning:** Turning off the trailing slash redirect may result in an information disclosure. Consider a situation where `mod_autoindex` is active (Options `+Indexes`) and `DirectoryIndex` is set to a valid resource (say, `index.html`) and there's no other special handler defined for that URL. In this case a request with a trailing slash would show the `index.html` file. But a request without trailing slash would list the directory contents.

## Module `mod_disk_cache`

Module `mod_disk_cache` supports directives for the IBM HTTP Server for i server.

### Two Phase Disk Cache Maintenance

The server may take each iteration of the disk cache maintenance process through one or two phases, depending on how much maintenance is needed. In the first phase, the server will examine the file system directories for the disk cache function and discard data that no longer complies with the current server configuration settings. It will also discard unused or unmodified data according to the criteria set by `CacheGcClean` or `CacheGcUnused` directives. File names and expiration times for the remaining data will be collected and the total amount of space allocated for them will be tallied. If the tally is above the maximum disk storage limit (set by `CacheSize`), the server will go into phase two. If the tally is at or below the maximum disk storage limit, the server will stop the current iteration of the maintenance process. If the server takes the current iteration into the second phase, information collected in the first phase for the remaining data is sorted according to cache expiry time. The server will then discard remaining data, by order of expiration (soonest to latest), until the amount of allocated space is at or below the maximum disk storage limit.

The following steps summarize the disk cache maintenance process:

#### Phase One:

1. Data files are examined, one by one, starting at the directory root specified by `CacheRoot`.
2. Data files not complying with settings specified for `CacheDirLevels`, `CacheDirLength`, `CacheMinFileSize`, and `CacheMaxFileSize` are discarded.
3. Unused or unmodified data matching the criteria set by `CacheGcClean` and `CacheGcUnused` directives is discarded.
4. File names and expiration times for remaining data is collected.

5. The total amount of space allocated for remaining data is determined. Phase two is entered if this total is greater than that specified by CacheSize. If not, phase two is skipped and maintenance completes (until the next iteration).

**Phase Two:**

1. Information collected in phase one for remaining data is sorted according to cache expiry times.
2. Data is discarded, by order of expiration (soonest to latest), until the total amount of allocated space is at or below that specified by CacheSize.

**Note:** The server stops collecting information for remaining data when it reaches the maximum amount of memory allowed for disk cache maintenance (set by CacheGcMemUsage). If the server reaches this limit in phase one, it may not have recorded enough information for phase two to bring the total amount of space allocated for the cache down to the limit specified by the CacheSize directive in one iteration of the disk cache maintenance process. In this case, a warning message is written to the server log and the server completes maintenance and waits for the next disk cache maintenance iteration.

**Directives**

- “CacheDirLength”
- “CacheDirLevels” on page 345
- “CacheGcClean” on page 346
- “CacheGcDaily” on page 347
- “CacheGcInterval” on page 349
- “CacheGcMemUsage” on page 350
- “CacheGcUnused” on page 350
- “CacheRoot” on page 352
- “CacheSize” on page 353

**CacheDirLength:**

**Module:** mod\_disk\_cache

**Syntax:** CacheDirLength *length*

**Default:** CacheDirLength 2

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule disk\_cache\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** CacheDirLength 4

The CacheDirLength directive specifies the number of characters in subdirectory names used by the disk cache function to store data.

**Parameter:** *length*

- The *length* parameter specifies the number of characters in subdirectory names used by the disk cache function. The specified value multiplied by the value specified for the CacheDirLevels directive must be less than or equal to 20.

If the values specified for CacheDirLevels and CacheDirLength are changed once they have been used to cache data, the server will discard all existing cache data when it runs disk cache maintenance since the file paths used to store data no longer adhere to the new values. See the CacheGcDaily or CacheGcInterval directives for more details on disk cache maintenance.

- This directive is used only if CacheRoot is set.

**Note:** HTTP Server does not support inheritance for the CacheDirLength directive.

### CacheDirLevels:

**Module:** mod\_disk\_cache

**Syntax:** CacheDirLevels *levels*

**Default:** CacheDirLevels 3

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule disk\_cache\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** CacheDirLevels 3

The CacheDirLevels directive specifies the number of directory levels used by the disk cache function to store data.

#### **Parameter:** *levels*

- The *length* parameter specifies the number of directory levels used by the disk cache function. The specified value multiplied by the value specified for the CacheDirLength directive must be less than or equal to 20.

A hash algorithm is used to generate unique and seemingly random character strings from hash keys (or URLs) provided for data stored in cache. These character strings are used to build unique file system path names. Data is stored in the file system using these path names, relative to the directory root specified by the CacheRoot directive. This setting specifies how many directory levels are used, while the CacheDirLength directives specifies the length of each subdirectory name, with remaining characters simply used for file names. The server uses the hash algorithm and directory levels to improve the performance of the server when working with a potentially large number of data files.

#### **Example 1**

```
CacheRoot /QIBM/UserData/HTTP/CacheRoot/MyCache
CacheDirLevels 3
CacheDirLength 1
```

The above example indicates that a hash key such as ftp://ibm.com/document.html may be used to build a directory path such as /x/3/\_/9sj4t2svBA where x, 3, and \_ are three subdirectory names (CacheDirLevels 3) each having a length of one character (CacheDirLength 1). The remaining characters, 9sj4t2svBA, are used for file names.

#### **Example 2**

```
CacheRoot /QIBM/UserData/HTTP/CacheRoot/MyCache
CacheDirLevels 5
CacheDirLength 2
```

The above example indicates that the same hash key described for example one (ftp://ibm.com/document.html) may be used to build a directory path such as /x3/\_9/sj/4t/2s/vBA where x3, \_9, sj, 4t, and 2s are five subdirectory names (CacheDirLevels 5) each having a length of two characters (CacheDirLength 2). The remaining characters, vBA, are used for file names.

Directory paths generated in this process are relative to the directory root defined by the CacheRoot directive. Therefore, for example one (above), two files, one named 9sj4t2svBA.data and the other named 9sj4t2svBA.header will be created to store data using the hash key ftp://ibm.com/document.html. Both files will reside within the /QIBM/UserData/HTTP/CacheRoot/MyCache/x/3/\_ directory. For example two

(above), the two files will be named vBA.data and vBA.header and will reside within the /QIBM/UserData/HTTP/CacheRoot/MyCache/x3/\_9/sj/4t/2s directory using the same hash key.

### Directory length and level limits:

Since the hash algorithm generates an exponential number of directories using this schema, a limit must be set upon the values that CacheDirLevels and CacheDirLength may have. The limits described as such:

`CacheDirLevels * CacheDirLength <= 20`

The maximum number of directory levels multiplied by the maximum length of each subdirectory must be less than or equal to 20. If not, the server will fail to activate at startup.

If the values specified for CacheDirLevels and CacheDirLength are changed once they have been used to cache data, the server will discard all existing cache data when it runs disk cache maintenance since the file paths used to store data no longer adhere to the new values. See the CacheGcDaily or CacheGcInterval directives for more details on disk cache maintenance.

- This directive is used only if CacheRoot is set.

**Note:** HTTP Server does not support inheritance for the CacheDirLevels directive.

### CacheGcClean:

**Module:** mod\_disk\_cache

**Syntax:** CacheGcClean *hash-key-criteria period*

**Default:** CacheGcClean \*2592000 (seconds, or 30 days)

**Context:** server config, virtual host

**Override:** none

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: `LoadModule disk_cache_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM`

**Example:** CacheGcClean `http://www.ibm.com /* 1296000`

The CacheGcClean directive specifies a complete URL or URL match expression and a maximum period value used to identify and remove data from cache that has not been updated (or written to cache) within the number of specified seconds. Multiple CacheGcClean directives are allowed. If disk cache maintenance is disabled, this setting has no affect and the cache may grow without bound, unless managed by some application or process other than the server.

This directive is similar to the CacheGcUnused directive, however the former distinguishes when data was last written (or saved) to cache, not when it was last served from cache.

#### Parameter One: *hash-key-criteria*

- The *hash-key-criteria* parameter accepts a complete URL or URL match expression used to identify cached data by hash key. Complete URLs do not contain asterisks (\*) or question marks (?) and must match hash keys URLs completely (see example two). URL match expressions contain one or more asterisks (\*) or question marks (?) used as wildcards to match multiple hash keys. For example: `http://ibm.com/*`, `*://ibm.com/*`, or `ftp://server?.ibm.com/*` (see example one).

#### Parameter Two: *period*

- The *period* parameter specifies the maximum amount of time (in seconds) that matched data may remain cached.

Cached data for the disk caching function is identified by comparing hash keys with the value specified for the hash-key-criteria parameter. Matched data that has not been updated (or written to cache) within the number of seconds specified by the corresponding period parameter is discarded



by the server during phase one of the disk cache maintenance process. Matched data that has been updated within the number of specified seconds is not affected. Unmatched data is not affected. See “Two Phase Disk Cache Maintenance” on page 343 for details concerning the disk cache maintenance process.

#### Example 1: URL match expressions

```
CacheRoot serverCache
CacheGcClean *://ibm.com/* 2592000
CacheGcClean ftp://server?.ibm.com/* 1209600
```

For this example, the first CacheGcClean directive ensures cached data with hash keys (or URLs) that match the expression `*://ibm.com/*` and has not been updated within the past 2592000 seconds (or 30 days) is discarded during phase one of the cache maintenance process. The second CacheGcClean directive ensures cached data with hash keys (or URLs) that match the expression `ftp://server?.ibm.com/*` and has not been updated within the past 1209600 seconds (or 2 weeks) is discarded.

Example one uses CacheGcClean directives with URL match expressions to manage data stored in cached using the disk cache function (CacheRoot serverCache). For the expression `*://ibm.com/*`, the first wildcard (\*) is used to match one or more characters in hash keys preceding the characters `//ibm.com/`. The second wildcard (\*) is used to match one or more characters succeeding the characters `//ibm.com/`. Hash keys that match this expression, for example, include `http://ibm.com/public/welcome.html` and `ftp://ibm.com/patch.zip`. For the expression `ftp://server?.ibm.com/*`, the first wildcard (?) is used to match any single character between `ftp://server` and `.ibm.com/`. The second wildcard (\*) is used to match one or more characters succeeding the characters `.ibm.com/`. Hash keys that match this expression, for example, include `ftp://server1.ibm.com/whitepaper.pdf` and `ftp://server5.ibm.com/downloads/driver.exe`.

#### Example Two: Complete URL

```
CacheRoot serverCache
CacheGcClean ftp://server5.ibm.com/downloads/application.zip 432000
```

For this example, the CacheGcClean directive uses a complete URL to ensure cached data with the hash key `ftp://server5.ibm.com/downloads/application.zip` is discarded during phase one of the disk cache maintenance process if it has not been updated within the past 432000 seconds (or 5 days). No other data will be matched since complete URLs identify a single hash key.

The server detects updates to cached data for the disk caching function by comparing the "Data change date/time" values of data file attributes. These are commonly referred to as last-modified times. When data is updated within cache, the corresponding last-modified times record the date and time that the last update was made.

- This directive is negated when `off` is specified for CacheGcDaily and CacheGcInterval is not specified.
- This directive is used only if CacheRoot is set.
- Disk cache maintenance may occur at regular time periods for CacheGcInterval and at a particular time of day for CacheGcDaily if both are set.

**Note:** HTTP Server does not support inheritance for the CacheGcClean directive.

#### CacheGcDaily:

**Module:** mod\_disk\_cache  
**Syntax:** CacheGcDaily *time-of-day* | *off*  
**Default:** CacheGcDaily 03:00  
**Context:** server config, virtual host  
**Override:** none  
**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule disk\_cache\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** CacheGcDaily 23

The CacheGcDaily directive specifies whether the server is to perform disk cache maintenance, at a particular time, when the disk cache function is enabled. If the disk cache function is disabled (the default), this setting has no effect and the server does not perform disk cache maintenance. The default value is 3:00 (3:00 am local system time).

**Parameter:** *time-of-day* | *off*

- The *time-of-day* parameter accepts a value in the HH:MM:SS format (24 hour clock) where HH is an hour value (0 to 23), MM is a minute value (0 to 59), and SS is a second value (0 to 59). A minute (MM) or second (SS) value is not required. If a minute value is not specified, maintenance will commence at the beginning of the hour specified by the hour value (see example two). Likewise, if a second value is not specified, maintenance will commence at the specified number of minutes past the hour (see example one).
- If *off* is specified, maintenance will not be performed based on a particular time of day (see example three).

If *off* is not specified, the server will perform cache maintenance every day, starting at the specified local system time (if disk caching is enabled, see examples one and two). If *off* is specified, the server will not perform disk cache maintenance at a specific time of day, however it may perform disk cache maintenance at regular time intervals, if a maintenance period is set using the CacheGcInterval directive. If *off* is specified, and a maintenance period is not specified using CacheGcInterval, the server will never perform disk cache maintenance (see example three).

#### Example 1

```
CacheRoot dataCache
CacheGcDaily 15:55
```

#### Example 2

```
CacheRoot dataCache
CacheGcDaily 9
```

#### Example 3

```
CacheRoot dataCache
CacheGcDaily off
```

For example one, the server will perform cache maintenance every day at 15:55 (or 3:55 pm local system time). For example two, the server will perform cache maintenance every day at 9:00 (or 9:00 am local system time). For example three, the server will not perform disk cache maintenance since CacheGcDaily is set to off, and CacheGcInterval is not specified.

See “Two Phase Disk Cache Maintenance” on page 343 for details concerning the disk cache maintenance process.

- Disk cache maintenance may occur at time intervals for CacheGcInterval and at a particular time of day for CacheGcDaily if both are set.
- This directive is used only if CacheRoot is set.

**Note:** HTTP Server does not support inheritance for the CacheGcDaily directive. For the configuration shown below, garbage collection is performed at 1:30 AM and again at 2:30 AM.

**Example:**

```
CacheRoot dataCache
CacheGcDaily 01:30:00
<VirtualHost ...>
  CacheGcDaily 02:30 00
</Virtual Host>
```

### CacheGcInterval:

**Module:** mod\_disk\_cache

**Syntax:** CacheGcInterval *period*

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule disk\_cache\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** CacheGcInterval 8100

The CacheGcInterval directive specifies whether the server is to perform disk cache maintenance, at regular time intervals, when the disk cache function is enabled. Maintenance for this setting will commence at the time the server is started, and repeat every number of specified seconds, until the server is ended. If the disk cache function is disabled (the default), this setting has no affect and the server does not perform disk cache maintenance.

**Parameter:** *period*

- The *period* parameter specifies a period for cache maintenance cycles, in seconds. The value may include a decimal to indicate fractional hours. For example, use CacheGcInterval 5400 to perform cache maintenance every 5400 seconds (every 90 minutes).

If this directive is not used (not specified), the server will not perform disk cache maintenance at regular time intervals, however it may at a particular time of day, if such a time is specified using the CacheGcDaily directive. If this directive is not used (not specified), and CacheGcDaily is set to *off*, the server will never perform disk cache maintenance (see example two).

#### Example 1

```
CacheRoot dataCache
CacheGcInterval 9900
```

#### Example 2

```
CacheRoot dataCache
CacheGcDaily offexample
```

For example one, the server will perform disk cache maintenance every 9900 seconds (every 2 hours and 45 minutes), starting from the time the server is started. For example two, the server will not perform disk cache maintenance since CacheGcDaily is set to off, and CacheGcInterval is not specified.

See “Two Phase Disk Cache Maintenance” on page 343 for details concerning the disk cache maintenance process.

- Disk cache maintenance may start at regular time intervals for CacheGcInterval and at a particular time of day for CacheGcDaily if both are set.
- This directive is used only if CacheRoot is set.

**Note:** HTTP Server does not support inheritance for the CacheGcInterval directive.

## CacheGcMemUsage:

**Module:** mod\_disk\_cache

**Syntax:** CacheGcMemUsage *size*

**Default:** CacheGcMemUsage 5000000

**Context:** server config, virtual host

**Override:** none

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule disk\_cache\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** CacheGcMemUsage 3000000

The CacheGcMemUsage directive specifies the maximum amount of system memory, in bytes, the server is to use to collect information for phase two of the disk cache maintenance process. See Two Phase Disk Cache Maintenance for details concerning the disk cache maintenance process.

**Parameter:** *size*

- The *size* parameter specifies, in bytes, the amount of main store memory that the server may use for phase two of the disk cache maintenance process.

When the amount of system memory consumed for phase two of the disk cache maintenance process reaches the value specified for the size parameter, the server stops collecting information for remaining data in cache but continues to do the other tasks for phase one until finished. If the server takes disk cache maintenance into phase two, only the information collected in phase one is used. This will not include information for all remaining cached data if the size parameter is not large enough.

### Example

```
CacheRoot dataCache
CacheGcDaily 5:00
CacheGcMemUsage 200000
```

For this example, the server will perform disk cache maintenance every day at 5:00 (CacheGcDaily 5:00). During phase one maintenance, the server records file names and expiration times for data remaining cached, until it consumes 200000 bytes of memory (CacheGcMemUsage 200000). After this limits reached, the server continues to perform the other phase one tasks. After all phase one tasks are complete, the server performs phase two maintenance (if needed) using whatever information it was able to collect in phase one.

- This directive is negated when *off* is specified for CacheGcDaily and CacheGcInterval is not specified.
- Cache maintenance may occur at time intervals for CacheGcInterval and at a particular time of day for CacheGcDaily if both are set.
- This directive is used only if CacheRoot is set, and cache maintenance is enabled.

**Note:** HTTP Server does not support inheritance for the CacheGcMemUsage directive.

## CacheGcUnused:

**Module:** mod\_disk\_cache

**Syntax:** CacheGcUnused *hash-key-criteria period*

**Default:** CacheGcUnused \* 1209600 (seconds, or 2 weeks)

**Context:** server config, virtual host

**Override:** none

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule disk\_cache\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** CacheGcUnused http://www.ibm.com/\* 432000

The CacheGcUnused directive specifies a complete URL or URL match expression and a maximum period value used to identify and remove data from cache that has not been used (or served from cache) within the number of specified seconds. Multiple CacheGcUnused directives are allowed. If disk cache maintenance is disabled (see “CacheGcDaily” on page 347 or “CacheGcInterval” on page 349), this setting has no effect and the cache may grow without bound, unless managed by some application or process other than the server itself.

This directive is similar to the “CacheGcClean” on page 346 directive, however the latter does not distinguish when data was last served from cache, but rather when it was last written (or saved) to cache.

**Parameter One:** *hash-key-criteria*

- The *hash-key-criteria* parameter accepts a complete URL or URL match expression used to identify cache data by hash key. Complete URLs do not contain asterisks (\*) or question marks (?) and must match hash keys completely (see example two). URL match expressions contain one or more asterisks (\*) or question marks (?) as wildcards to match multiple hash keys. For example, `http://*` or `ftp://server?.ibm.com/*` (see example one).

**Parameter Two:** *period*

- The *period* parameter specifies the maximum amount of time (in seconds) that matched data may remain cached.

Cached data for the disk caching function is identified for this setting by comparing hash keys with the value specified for the hash-key-criteria parameter. Matched data that has not been used (or served from cache) within the number of seconds specified by the corresponding period parameter are discarded by the server during phase one of the disk cache maintenance process. Matched data that has been used within the number of specified seconds is not affected. Unmatched documents are not affected. See “Two Phase Disk Cache Maintenance” on page 343 for details concerning the disk cache maintenance process.

**Example 1: URL match expressions**

```
CacheRoot serverCache
CacheGcUnused http://* 25929000
CacheGcUnused ftp://server?.ibm.com/* 1209600
```

For this example, the first CacheGcUnused directive ensures that cached data with hash keys (or URLs) that match the expression `http://*` and has not been updated within the past 25929000 seconds (or 30 days) are discarded during phase one of the disk cache maintenance process. The second CacheGcClean directive ensures that cached data with hash keys (or URLs) that match the expression `ftp://server?.ibm.com/*` and has not been updated within the past 1209600 seconds (or 2 weeks) is discarded.

Example one uses CacheGcUnused directives with URL match expressions to manage data stored in cache using the disk caching function (CacheRoot serverCache). For the expression `http://*`, the wildcard (\*) is used to match one or more characters in hash keys preceding the characters `http://`. This expression matches all hash keys starting with the characters `http://`. For the expression `ftp://server?.ibm.com/*`, the first wildcard (?) is used to match any single character in hash keys between `ftp://server` and `.ibm.com/`. The second wildcard (\*) is used to match one or more characters in hash keys succeeding the characters `.ibm.com/`. Hash keys that match this expression, for example, include `ftp://server1.ibm.com/whitepaper.pdf` and `ftp://server5.ibm.com/downloads/driver.exe`.

**Example 2: Complete URL**

```
ProxyRequests on
CacheRoot serverCache
CacheGcUnused ftp://server5.ibm.com/downloads/application.zip 432000
```

For this example, the CacheGcUnused directive uses a complete URL to ensure cached data with the hash key ftp://ftpserver.ibm.com/downloads/application.zip is discarded during phase one of the disk cache maintenance process if it has not been requested within the past 432000 seconds (or 5 days). No other data will be matched since complete URLs identify a single hash key.

The server detects requests for cached data for the disk caching function by comparing the "Last access date/time" values of data file attributes. These are commonly referred to as last-accessed times. When data is served from cache, the corresponding last-accessed times record the date and time that the last request was served.

- This directive is negated when off is specified for CacheGcDaily and CacheGcInterval is not specified.
- Cache maintenance may occur at regular time periods for CacheGcInterval and at a particular time of day for CacheGcDaily if both are set.
- This directive is used only if CacheRoot is set.

**Note:** HTTP Server does not support inheritance for the CacheGcUnused directive.

### CacheRoot:

**Module:** mod\_disk\_cache

**Syntax:** CacheRoot *directory*

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule disk\_cache\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** CacheRoot webProxyCache

The CacheRoot directive enables the disk cache function and specifies the name of the file system directory root. Setting this directive also enables disk cache maintenance for the CacheGcDaily directive, by default, and the CacheGcInterval directive. See the "CacheGcDaily" on page 347 or "CacheGcInterval" on page 349 directives for more details on disk cache maintenance.

#### Parameter: *directory*

- The *directory* parameter accepts a file system path name to specify the file system directory root for the disk cache function (see directory root limits below).

The disk cache function provides underlying cache support for a local proxy cache and user written modules, using local file system space (disk space). The server must have \*RWX data authorities and \*ALL object authorities to the specified directory.

A hash algorithm is used to generate unique and seemingly random file system path names based on hash keys (or URLs) provided for data stored in cache (see also CacheDirLength and CacheDirLevels). Data is stored in the local file system using these path names, relative to the specified directory root. The following limits are placed on the directory root:

#### Directory root limits:

- If the directory parameter specifies an absolute path it must start with /QIBM/UserData/HTTP/CacheRoot, otherwise the proxy will fail to activate at startup.
- If the directory parameter does not specify an absolute path (does not start with a '/'), it will be assumed to be relative to the following: /QIBM/UserData/HTTP/CacheRoot

The directory will be created if it does not exist prior to server startup. Only the last directory in the path will be created. All other directories in the path must previously exist.

For example, if "CacheRoot abc/def" is configured, the server will create directory "/QIBM/UserData/HTTP/CacheRoot/ABC/def".

**Example 1: Absolute Path**

```
CacheRoot /QIBM/UserData/HTTP/CacheRoot/proxyCache
ProxyRequests on
```

**Example 2: Relative Path**

```
CacheRoot proxyCache
CacheEnable %%PROXY%%
ProxyRequests on
```

**Example 3: Relative Path (with disk cache function unavailable for proxy data)**

```
CacheRoot cache
CacheEnable disk /
ProxyRequests on
```

**Example 4: Bad Path**

```
CacheRoot /MyServerCache
```

For example one, CacheRoot enables the disk cache function (CacheRoot /QIBM/UserData/HTTP/CacheRoot/proxyCache), ProxyRequests specifies that the proxy function is enabled to handle forward proxy requests (ProxyRequests on). With these directive settings, HTTP proxy response data is cached and maintained within the /QIBM/UserData/HTTP/CacheRoot/proxyCache directory using disk cache function support. See the ProxyRequests directive for more information on handling proxy requests and caching HTTP proxy response data.

For example two, the disk cache function is enabled (CacheRoot proxyCache), the proxy function is enabled (ProxyRequests on), and the local proxy cache is enabled. With these directive settings, HTTP proxy response data is cached and maintained within the proxyCache directory, relative to the /QIBM/UserData/HTTP/CacheRoot/ directory. This directory is the same one described in example one, simply specified as a relative path name rather than an absolute path name. Either specification is acceptable.

For example three, the disk cache function is enabled (CacheRoot cache), and the proxy function is enabled (ProxyRequests on), however the local proxy cache is disabled. With these directive settings, the disk cache function is not used to cache data for the proxy function, but may be used to cache data for user written modules.

For example four, the directory specified for CacheRoot is not valid since an absolute path within /QIBM/UserData/HTTP/CacheRoot/ is not specified. With this configuration the server will generate an error message(s) at startup and fail to activate.

- This directive is required when ProxyNoConnect is set to on.

**Note:** HTTP Server does not support inheritance for the CacheRoot directive.

**CacheSize:**

**Module:** mod\_disk\_cache

**Syntax:** CacheSize *size*

**Default:** CacheSize 5000000

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule cache_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule disk_cache_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

**Example:** CacheSize 8550

The CacheSize directive specifies the maximum amount of system storage space allocated for the disk cache function (in kilobytes). Although actual usage may exceed this setting, the server will discard data when it runs disk cache maintenance until the total allocated cache space is at or below this setting. If disk cache maintenance is disabled, this setting has no affect and the cache may grow without bound, unless managed by some application or process other than the server itself. See “CacheGcDaily” on page 347 or “CacheGcInterval” on page 349 for more details on the disk cache maintenance process.

**Parameter:** *size*

- The *size* parameter specifies the maximum number of kilobytes allocated for the disk cache function. Depending on the expected server traffic volume, and values set for CacheGcInterval or CacheGcDaily, use a size value that is at least twenty to forty percent lower than the available space.

The disk cache function uses the local file system to store data. Therefore, space allocated for this cache is used to maintain directory structures and file attributes as well as to store cache data. It also includes unused space within file system storage blocks allocated to files and directories. Therefore, the total amount of system storage allocated for the cache will always be greater than the total amount of actual cache data. This setting sets a limit for the total amount of allocated space, not a limit for the total amount of actual cache data.

- This directive is negated when off is specified for CacheGcDaily and CacheGcInterval is not specified.
- This directive is used only if CacheRoot is set.

**Note:** HTTP Server does not support inheritance for the CacheSize directive.

## Module mod\_env

Module mod\_env supports directives for the IBM HTTP Server for i Web server.

### Summary

This module allows the HTTP Server CGI and SSI environment to inherit environment variables.

### Directives

- “PassEnv”
- “SetEnv” on page 355
- “UnsetEnv” on page 355

### PassEnv:

**Module:** mod\_env

**Syntax:** PassEnv *variable* [*variable* ...]

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** none

**Origin:** Apache

**Example:** PassEnv LD\_LIBRARY\_PATH



The PassEnv directive specifies one or more environment variables to pass to the CGI scripts. The variables originate from the server's own environment. See "Environment variables set by HTTP Server" on page 493 for more information.

**Parameter:** *variable*

- The *variable* parameter is any valid environment variable.

**SetEnv:**

**Module:** mod\_env

**Syntax:** SetEnv *variable* [*value*]

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** FileInfo

**Origin:** Apache

**Examples:**

- SetEnv SPECIAL\_PATH /QIBM/bin
- SetEnv QIBM\_CGI\_LIBRARY\_LIST "MIME;CGIURL;CGILIBL"

The SetEnv directive allows you to set an environment variable that is passed on to CGI scripts. See "Environment variables set by HTTP Server" on page 493 for more information.

**Parameter One:** *variable*

- The *variable* parameter is any valid EBCDIC characters except the equal sign (=), the null-terminator (X'00') and blank (X'40'). The name must be enclosed in quotation marks if it contains any non-alphanumeric character.

**Parameter Two:** [*value*]

- The [*value*] parameter is optional and can include partial URLs. The case is preserved when lowercase characters are specified. Valid values include all EBCDIC characters. The value must be enclosed in quotation marks if it contains any non-alphanumeric character or blanks. Lowercase characters for the library names will not work if this directive is used to change the library list. When changing the library list values, the libraries need to be separated by a semicolon.

**UnsetEnv:**

**Module:** mod\_env

**Syntax:** UnsetEnv *variable* [*variable* ...]

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** none

**Origin:** Apache

**Example:** UnsetEnv LD\_LIBRARY\_PATH

The UnsetEnv directive removes one or more environment variables from those passed on to CGI scripts. See "Environment variables set by HTTP Server" on page 493 for more information.

**Parameter:** *variable*

- The *variable* parameter is any valid environment variable.

## Module mod\_example

Module mod\_example supports directives for the IBM HTTP Server for i Web server.

## Summary

The module `mod_example` provides a simple example to demonstrate the use of the Apache APIs.

### Directive

- "Example"

### Example:

**Module:** `mod_example`

**Syntax:** `Example`

**Default:** none

**Context:** server config, virtual host, directory, `.htaccess`

**Override:** Options

**Origin:** Apache

**Usage Considerations:** A `LoadModule` is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule example_module /QSYS.LIB/QHTTPSVR.LIB/QZSREXAMPL.SRVPGM
```

**Example:** `Example`

This directive sets a demonstration flag. The example module's content handler displays the flag. There are no arguments. If you browse a URL to which the example content-handler applies, the routines within the module and how and in what order they were called to service the document request are displayed.

## Module `mod_expires`

Module `mod_expires` supports directives for the IBM HTTP Server for i Web server.

### Summary

This module controls the setting of the Expires HTTP header in server responses. The expiration date can be set relative to either the time that the source file was last modified, or relative to the time that the client accessed the server.

The Expires HTTP header is an instruction to the client regarding the document's validity and persistence. If cached, the document may be retrieved from the cache rather than from the source until the allocated time has passed. After this occurs, the cache copy is considered "expired" and a new copy must be obtained from the source.

### Alternate Interval Syntax

The `ExpiresDefault` and `ExpiresByType` directives can also be defined in a more readable syntax of the form:

```
ExpiresDefault "<base> [plus] {<num> <type>}"  
ExpiresByType type|encoding "<base> [plus] {<num> <type>}"
```

The `<base>` argument is one of the following:

- access
- now (equivalent to 'access')
- modification

The `[plus]` keyword is optional. The `<num>` argument should be an integer value [acceptable to `atoi()`], and `<type>` is one of the following:

- years
- months

- weeks
- days
- hours
- minutes
- seconds

For example, any of the following directives can be used to make documents expire 1 month after being accessed, by default:

```
ExpiresDefault "access plus 1 month"
ExpiresDefault "access plus 4 weeks"
ExpiresDefault "access plus 30 days"
```

**Note:** Time is stored in seconds. The value month is actually calculated as 60\*60\*24\*30 seconds. Keep in mind that one month is equal 30 days, and 4 weeks is only equal to 28 days. If you specify 52 weeks, it is calculated as 362 days instead of 365 days.

The expiry time can be fine-tuned by adding several <num> and <type> arguments. For example:

```
ExpiresByType text/html "access plus 1 month 15 days 2 hours"
ExpiresByType image/gif "modification plus 5 hours 3 minutes"
```

**Note:** If you use a modification date based setting, the Expires header is added only to content that comes from a file on a disk, because there is no modification time for content that does not come from a file on a disk.

## Directives

- “ExpiresActive”
- “ExpiresByType” on page 358
- “ExpiresDefault” on page 358

## ExpiresActive:

**Module:** mod\_expires

**Syntax:** ExpiresActive *on* | *off*

**Default:** ExpiresActive off

**Context:** server config, virtual host, directory, .htaccess

**Override:** Indexes

**Origin:** Apache

**Example:** ExpiresActive on

The ExpiresActive directive enables or disables the generation of the Expires header for the document realm in question. If this directive is found in an .htaccess file it only applies to documents generated from that directory.

**Parameter:** *on* | *off*

- If set to *on*, an Expires header will be added to served documents according to criteria set by the ExpiresByType and ExpiresDefault directives.
- If set to *off*, an Expires header will not be generated for any document in the realm (unless overridden at a lower level, such as an .htaccess file overriding a server config file).

**Note:** This directive does not guarantee that an Expires header will be generated. If the criteria is not met, no header will be sent, and the effect will be as though this directive was never specified.

## ExpiresByType:

**Module:** mod\_expires

**Syntax:** ExpiresByType *MIME-type code seconds* | "*<base> [plus] <num> <type>*"

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** Indexes

**Origin:** Apache

**Example:** ExpiresByType image/gif A2592000

**Example:** ExpiresByType text/html "access plus 30 days"

The ExpiresByType directive defines the value of the Expires header generated for documents of the specified type ( for example, text/html). The second argument sets the number of seconds that will be added to a base time to construct the expiration date.

The base time is either the last modification time of the file, or the time of the client's access to the document. Whether access time or modification time should be used is specified by the code field. *M* means that the file's last modification time should be used as the base time, and *A* means the client's access time should be used.

The difference in effect between the *A* and the *M* is minimal. If *M* is used, all current copies of the document in all caches will expire at the same time. This could be useful for something like a weekly notice that is always found at the same URL. If *A* is used, the date of expiration is different for each client. This could be useful for image files that do not change very often, particularly for a set of related documents that all refer to the same images ( for example, the images will be accessed repeatedly within a relatively short time span).

**Parameter One:** *MIME-type*

- The document type for which an Expires header should be generated.

**Parameter Two:** *code*

- The *code* parameter specifies one of two possible choices. Specify *A* if the expiration time should be calculated from the time the resource was accessed. Specify *M* if the expiration time should be calculated from the last modified date of the resource.

**Parameter Three:** *seconds*

- The *seconds* parameter is a number of seconds until the resource expires.

Here is an example to specify the expiration time calculation. For examples using alternate syntax, see the beginning of this topic.

```
#enable expirations
ExpiresActive on
#expire GIF images after a month in the clients cache
ExpiresByType image/gif A2592000
#HTML documents are good for a week from the time they were changed
ExpiresByType text/html M604800
```

**Note:** This directive only has effect if ExpiresActive On has been specified. It overrides, for the specified MIME type only, any expiration date set by the ExpiresDefault directive. If you use a modification date based setting, the Expires header will not be added to content that does not come from a file on disk. This is due to the fact that there is no modification time for such content.

## ExpiresDefault:

**Module:** mod\_expires

**Syntax:** ExpiresDefault *code seconds* | "*<base> [plus] <num> <type>*"

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** Indexes

**Origin:** Apache

**Example:** ExpiresDefault A2592000

**Example:** ExpiresDefault "access plus 1 month"

The ExpiresDefault directive sets the default algorithm for calculating the expiration time for all documents in the affected realm. It can be overridden on a type-by-type basis by the ExpiresByType directive. See the description of the ExpiresByType directive for details about the syntax of the argument, and the alternate syntax description as well.

**Parameter One:** *code*

- The code parameter specifies has two arguments. Specify *A* if the expiration time should be calculated from the time the resource was accessed. Specify *M* if the expiration time should be calculated from the last modified date of the resource.

**Parameter Two:** *seconds*

- The *seconds* parameter is a number of seconds until the resource expires.

**Note:**

- If you use a modification date based setting, the Expires header will not be added to content that does not come from a file on disk. This is due to the fact that there is no modification time for such content.
- You can also specify the expiration time calculation using the alternate interval syntax. For examples using alternate syntax, see the beginning of this topic.

## Module mod\_ha

Module mod\_ha supports directives for the IBM HTTP Server for i Web server.

### Summary

The module mod\_ha contains directives that define support for the highly available HTTP Server function.

### Directives

- "HACGI"
- "HAModel" on page 360
- "LmExitProgram" on page 361
- "LmIntervalTime" on page 361
- "LmMaxReactivation" on page 362
- "LmResponseTime" on page 362
- "LmUrlCheck" on page 363
- "LmUrlCheckBackup" on page 364

### HACGI:

**Module:** mod\_ha

**Syntax:** HACGI *on* | *off*

**Default:** HACGI *off*

**Context:** server config, virtual host, directory, .htaccess

**Override:** none

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ha\_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM

**Example:** HACGI on

The HACGI directive specifies if CGI programs in a directory can be highly available. The CGI programs in the specified directory must use the highly available HTTP Server APIs.

**Parameter:** *on* | *off*

- The *on* parameter value specifies CGI programs in a directory can be highly available.
- The *off* parameter value specifies CGI programs in a directory are not high available.

## HAModel:

**Module:** mod\_ha

**Syntax:** HAModel *model*

**Default:** none

**Context:** server config

**Override:** none

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ha\_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM

**Example:** HAModel PrimaryBackupWithIpTakeover

**Example:** HAModel PrimaryBackupWithDispatcher

**Example:** HAModel PurePeer

The HAModel directive establishes which highly available model is to be used (PrimaryBackupWithIpTakeover, PrimaryBackupWithDispatcher, or PurePeer).

**Parameter:** *model*

- The *PrimaryBackupWithIpTakeover* parameter value specifies that the highly available Web server runs on the primary and all backup nodes. The backup node or nodes are in a idle state, ready to become the primary Web server should the primary Web server fail (failover), or a switchover takes place.
- The *PrimaryBackupWithDispatcher* parameter value specifies that the highly available Web server runs on the primary and all backup nodes. The backup nodes are in an idle state and all client requests are served by the primary node. A network dispatcher (for example the IBM WebSphere Edge Server) sends client requests to the Web server.
- The *PurePeer* parameter value specifies that all highly available nodes are in an active state and serve client requests. A network dispatcher (for example the IBM WebSphere Edge Server) evenly distributes requests to different cluster nodes. This guarantees distribution of resources in case of heavy load. Linear scalability is not guaranteed beyond a small number of nodes. After some number of nodes are added, scalability can disappear, and the cluster performance can deteriorate.

See “Highly available HTTP Server” on page 43 for more information regarding highly available Web server models.

### Example

```
LoadModule ha_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
HAModel PrimaryBackupWithIPTakeover
LmUrlCheck http://hostname/web/docs/spec/wscheck.html
LmIntervalTime 100
LmMaxReactivation 5
LmResponseTime 300
```

**Note:** When a server is configured as highly available (HAModel directive is specified), “HotBackup” on page 301 behaves as if it is set to 'off' and can not be overwritten.

### **LmExitProgram:**

**Module:** mod\_ha

**Syntax:** LmExitProgram *libraryname programname [userprofile]*

**Default:** none

**Context:** server config

**Override:** none

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ha\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** LmExitProgram httpstest exitpgm joeuser

The LmExitProgram directive is used to specify a user-defined program in the QSYS file system that is started by the Liveness Monitor whenever it initiates a change in the HA model of a server instance from the primary model or to the primary model. When the server instance is going to become the primary HA server instance, then this program is called with a parameter of '1'. When the current HA primary server instance is no longer going to be the primary instance, then this program is called with a parameter of '0'. For example a program can be created which will start or end a job, depending on the role of the server.

#### **Parameter One:** *libraryname*

- The *libraryname* parameter value specifies the name of the library to be used. The parameter value can be up to 10 characters and must follow the rules for IBM i library names.

#### **Parameter Two:** *programname*

- The *programname* parameter value specifies the name of the program to be used. The parameter value can be up to 10 characters and must follow the IBM i rules for program names in a library.

#### **Parameter Three:** *userprofile*

- The *userprofile* parameter value is optional and specifies which user profile the named program should run under. If the *userprofile* parameter is not specified, user profile QTMHHTTP is used.

### **LmIntervalTime:**

**Module:** mod\_ha

**Syntax:** LmIntervalTime *interval*

**Default:** LmIntervalTime 15

**Context:** server config

**Override:** none

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ha\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** LmIntervalTime 30

The LmIntervalTime directive is used by the Liveness Monitor to specify how often (in seconds, between performing Web server Liveness checks (HEAD or GET)) a liveness check should be performed on the server. The LmResponseTime and LmIntervalTime directives are independent. One sends out checks (LmIntervalTime), while the other tests for responses (LmResponseTime). The LmResponseTime value should always be larger than the LmIntervalTime value. It is recommended that LmResponseTime be at least 3 times larger than LmIntervalTime.

**Parameter:** *integer*

- The *interval* parameter value specifies how often (in seconds, between performing Web server Liveness checks (HEAD or GET)) a liveness check should be performed on the server. Valid values include integers between 0 and 4,294,967,295.

**Example**

```
LoadModule ha_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
HAMode1 PrimaryBackupWithIPTakeover
LmUrlCheck http://hostname/web/docs/spec/wscheck.html
LmIntervalTime 100
LmMaxReactivation 5
LmResponseTime 300
```

**LmMaxReactivation:**

**Module:** mod\_ha

**Syntax:** LmMaxReactivation *integer*

**Default:** LmMaxReactivation 3

**Context:** server config

**Override:** none

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ha\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** LmMaxReactivation 5

The LmMaxReactivation directive specifies how many times the Liveness Monitor should attempt to reactivate the Web server after a detected failure.

**Parameter:** *integer*

- The *integer* parameter value specifies how many times the Liveness Monitor should attempt to reactivate the Web server after a detected failure. Valid values include integers between 0 and 2,147,483,647

**Example**

```
LoadModule ha_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
HAMode1 PrimaryBackupWithIPTakeover
LmUrlCheck http://hostname/web/docs/spec/wscheck.html
LmIntervalTime 100
LmMaxReactivation 5
LmResponseTime 300
```

**LmResponseTime:**

**Module:** mod\_ha

**Syntax:** LmResponseTime *interval*

**Default:** LmResponseTime 120

**Context:** server config

**Override:** none

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ha\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** LmResponseTime 60

The LmResponseTime directive specifies how long the Liveness Monitor should wait for a response from the Web server before taking appropriate action (based on the other Liveness Monitor directive settings). The LmResponseTime and LmIntervalTime directives are independent. One sends out checks



(LmIntervalTime), while the other tests for responses (LmResponseTime). The LmResponseTime value should always be larger than the LmIntervalTime value. It is recommended that LmResponseTime be at least 3 times larger than LmIntervalTime.

**Parameter:** *interval*

- The *interval* parameter value specifies how long the Liveness Monitor should wait for a response from the Web server before taking appropriate action (based on the other Liveness Monitor directive settings).

**Example**

```
LoadModule ha_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
HAModel PrimaryBackupWithIPTakeover
LmUrlCheck http://hostname/web/docs/spec/wscheck.html
LmIntervalTime 100
LmMaxReactivation 5
LmResponseTime 300
```

**LmUrlCheck:**

**Module:** mod\_ha

**Syntax:** LmUrlCheck *URL*

**Default:** LmUrlCheck http://

**Context:** server config

**Override:** none

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ha\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** LmUrlCheck http://194.170.2.5:8000/web/docs/spec/wscheck.html

The LmUrlCheck directive specifies a fully qualified URL that is used by the Liveness Monitor to perform liveness checks on HTTP Server. Specifying a domain name is not valid for this directive. Only one IP address can be specified in a highly available HTTP Server configuration.

**Note:** This is a required directive for highly available and must exist in the global server configuration context and not in a container. See “HAModel” on page 360 and “Highly available HTTP Server” on page 43 for additional details.

**Parameter:** *URL*

- The *URL* parameter value specifies a fully qualified URL that is used by the Liveness Monitor to perform liveness checks on the server. Only one IP address can be specified in a highly available server configuration. Specifying a domain name is not valid for this parameter. The IP Address must be the same address as specified with the Listen directive. The default port number is 80

**Example**

```
LoadModule ha_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
HAModel PrimaryBackupWithIPTakeover
LmUrlCheck http://194.170.2.5:8000/web/docs/spec/wscheck.html
LmIntervalTime 20
LmMaxReactivation 3
LmResponseTime 60
```

Specify *https* when the HTTP Server instance is configured to receive client requests using only secure sockets. The IP address must be the same as the IP address that was specified in the virtual host container for the SSL application. The default port number for SSL is 443.

### Example

```
LoadModule ha_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
HAModel PrimaryBackupWithIPTakeover
LmUrlCheck https://194.170.2.5:8008/web/docs/spec/wscheck.html
LmIntervalTime 20
LmMaxReactivation 3
LmResponseTime 60
```

### LmUrlCheckBackup:

**Module:** mod\_ha

**Syntax:** LmUrlCheckBackup *URL*

**Default:** none

**Context:** server config

**Override:** none

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ha\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** LmUrlCheckBackup http://194.170.2.5:8008/web/docs/spec/wscheck.html

The LmUrlCheckBackup directive specifies a fully qualified URL that is used by the Liveness Monitor to perform liveness checks on the HA backup server instance. If this directive is not configured, then the URL passed is the URL parameter value specified on the LmUrlCheck directive.

For example, if the server is configured to run Payment Manager, only one instance of Payment Manager can be active in the cluster at any given time. If the URL on the LmUrlCheck directive specifies a URL for the Payment Manager, then this same URL will not work for the HA backup server instance, so the LmUrlCheckBackup directive needs to be configured to use a non-Payment Manager URL.

**Note:** This directive is ignored when HAModel *PurePeer* is configured.

### Parameter: *URL*

- The *URL* parameter value specifies a fully qualified URL that is used by the Liveness Monitor to perform liveness checks on the HTTP Server when the server is currently the HA backup server. The use of a domain name is not a valid parameter with this directive. Only one IP address can be specified in a High Availability server configuration (i.e. Listen 194.170.2.5:xxxxxx <virtual host 194.170.2.5:yyyyyy>). The IP Address must be the same address as specified with the Listen directive. The default port number is 80.

### Example

```
LoadModule ha_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
HAModel Primary/BackupWithIPTakeover
LmUrlCheck http://194.170.2.5:8008/web/docs/spec/wscheck.html
LmUrlCheckBackup https://194.170.2.5:8008/web/docs/spec/wscheckbackup.html
LmIntervalTime 20
LmMaxReactivation 3
LmResponseTime 60
```

Specify *https* when the HTTP Server instance is configured to receive client requests using only secure sockets. The IP address must be the same as the IP address that was specified in the virtual host container for the SSL application. The default port number for SSL is 443.

### Example

```
LoadModule ha_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
HAModel Primary/BackupWithIPTakeover
LmUrlCheck http://194.170.2.5:8008/web/docs/spec/wscheck.html
LmUrlCheckBackup https://194.170.2.5:8008/web/docs/spec/wscheckbackup.html
LmIntervalTime 20
LmMaxReactivation 3
LmResponseTime 60
```

**Note:** This directive is ignored when HAModel *PurePeer* is configured.

## Module mod\_headers

Module mod\_headers supports directives for the IBM HTTP Server for i Web server.

### Summary

The headers module allows for the customization of HTTP request and response headers. The module allows headers to be merged, replaced or removed.

### Directives

- “Header”
- “RequestHeader” on page 367

### Header:

**Module:** mod\_headers

**Syntax:** Header [condition] set | append | add *header value* [env=(!)variable]

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** FileInfo

**Origin:** Apache

**Example:** Header append Author "John P. Doe"

**Example:** Header unset Author

**Example:** Header echo ^TS\*

**Example:** Header echo Host

The Header directive can replace, merge or remove HTTP response headers. The action performed by this module is determined by the action parameter. This parameter is followed by a header name, which can include the final colon, but it is not required. Case is also ignored. For add, append, and set, a value is given as the next parameter. If this value contains spaces, it should be surrounded by double quotes. For "unset", no value should be given.

**Note:**The [condition] parameter was introduced with Apache 2.0.52.

### Order of Processing

The Header directive can occur almost anywhere within the server configuration. It is valid in the main server config and virtual host contexts, inside <Directory>, <Location>, and <Files> contexts and within .htaccess files.

The Header directives are processed in the following order:

1. main server
2. virtual host
3. <Directory> sections and .htaccess
4. <Location>
5. <Files>

Order is important. These two headers have a different effect if reversed. For example:

```
Header append Author "John P. Doe"
Header unset Author
```

This way the Author header is not set. If reversed, the Author header is set to "John P. Doe". The Header directives are processed just before the response is sent by its handler. This means that some headers, that are added just before the response is sent, cannot be changed or overridden. This includes headers such as Date and Server.

**Parameter One:** *condition*

- The condition parameter is an optional parameter which can be one of the following values:

Condition	Description
onsuccess	The Header directive will only effect responses with a status code of 2xx.
always	The Header directive will effect all responses, including 2xx.

**Parameter Two:** *action*

- The action parameter can be one of the following values:

Action	Description
set	The response header is set, replacing any previous header with this name.
append	The response header is appended to any existing header of the same name. When a new value is merged onto an existing header it is separated from the existing header with a comma. This is the HTTP standard way of giving a header multiple values.
add	The response header is added to the existing set of headers, even if this header already exists. This can result in two (or more) headers having the same name. This can lead to unforeseen consequence, and in general, "append" should be used instead.
unset	The response header of this name is removed, if it exists. If there are multiple headers of the same name, all will be removed.
echo	Request headers with this name are echoed back in the response headers. <i>Header</i> may be a regular expression. Echo without any parameters echoes back all the request headers in the response.

**Parameter Three:** *header*

- The HTTP Response *header* parameter to be set, appended, or unset with this directive. There is no validity checking of the header, which allows the use of experimental headers.

**Parameter Four:** *value*

- The *value* parameter may be a character string, a string containing format specifiers or a combination of both and specifies the value of the header to be set. It is only valid for set, add and append. There is no validity checking of the value specified, which allows the use of experimental headers values. All characters and escaped characters, such as '\n', are allowed in the value string. If *value* contains spaces, it should be surrounded by double quotes. The following format specifiers are supported in value:

**%t** The time the request was received in Universal Coordinated Time since the epoch (Jan. 1, 1970) measured in microseconds. The value is preceded by "t=".

**%D** The time from when the request was received to the time the headers are sent on the wire. This is a measure of the duration of the request. The value is preceded by "D=".

**%{ENVVAR}e**

The contents of the environment variable ENVVAR.

Other format strings, such as %s, will receive an error and the server will not start.

**Parameter Five:** *env=[!]environment-variable*

- Optional: The envvar parameter specified in the env=[!]envvar clause can be any character or numeric value. When the Header directive is used with the add, append, or set argument, a

fourth argument may be used to specify conditions under which the action will be taken. If the environment variable specified in the `env=...` argument exists (or if the environment variable does not exist and `env=!...` is specified) then the action specified by the Header directive will take effect. Otherwise, the directive will have no effect on the request. See the environment variable directives for details on naming environment variables.

The Header directives are processed just before the response is sent to the network. This means that it is possible to set or override most headers, except for those headers added by the header filter.

### **RequestHeader:**

**Module:** mod\_headers

**Syntax:** RequestHeader *action header value*

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** none

**Origin:** Apache

**Example:** RequestHeader set Accept-Encoding "gzip"

**Example:** RequestHeader unset Referer

The RequestHeader directive can replace, merge or remove HTTP request headers. The header is modified just before the content handler is run, allowing incoming headers to be modified. The action it performs is determined by the first argument.

This argument is followed by a header name, which can include the final colon, but it is not required. Case is ignored. For add, append, and set, a value is given as the third argument. If this value contains spaces, it should be surrounded by double quotes. For unset, no value should be given.

### **Order of Processing**

The RequestHeader (and Header) directives can occur almost anywhere within the server configuration. It is valid in the main server config and virtual host sections, inside <Directory>, <Location>, and <Files> sections, and within .htaccess files.

The RequestHeader directives are processed in the following order:

1. main server
2. virtual host
3. <Directory> sections and .htaccess
4. <Location>
5. <Files>

Order is important. These two headers have a different effect if reversed:

```
RequestHeader append MirrorID "mirror 12"  
RequestHeader unset MirrorID
```

This way round, the MirrorID header is not set. If reversed, the MirrorID header is set to "mirror 12".

The RequestHeader directive is processed just before the request is run by its handler in the fixup phase. This should allow headers generated by the browser, or by Apache input filters to be overridden or modified. For example,

```
RequestHeader append MirrorID "mirror 12"
```

### **Parameter One:** *action*

- The action parameter can be one of the following values:

Action	Description
set	The request header is set, replacing any previous header with this name.
append	The request header is appended to any existing header of the same name. When a new value is merged into an existing header, it is separated from the existing header with a comma. This is the HTTP standard way of giving a header multiple values.
add	The request header is added to the existing set of headers, even if this header already exists. This can result in two (or more) headers having the same name. This can lead to unforeseen consequences, and in general <b>append</b> should be used instead.
unset	The request header of this name is removed, if it exists. If there are multiple headers of the same name, all will be removed.

**Parameter Two:** *header*

- The HTTP Response *header* parameter to be set, appended, or unset with this directive. There is no validity checking of the header, which allows the use of experimental headers.

**Parameter Three:** *value*

- The *value* parameter can have any valid character string as a value and specifies the value of the header to be set. It is only valid for set, add and append. There is no validity checking of the value specified, which allows the use of experimental headers values. All characters and escaped characters are allowed in this string. If this value contains spaces, it should be surrounded by double quotes.

**Module mod\_ibm\_linc**

Module mod\_ibm\_linc supports directives for the IBM HTTP Server for i Web server.

**Summary**

The module mod\_ibm\_linc supports the LDAPInclude directive that allows HTTP Server to access a Lightweight Directory Access Protocol (LDAP) directory and to query the directory in a database fashion to obtain HTTP configuration information. The LDAPInclude directive requires a file that contains the directives necessary to contact the LDAP server. This can be the same filename that is used on the LdapConfigFile directive. See the mod\_ibm\_ldap module for details on the directives necessary to contact an LDAP server.

If the LDAPInclude directive is placed in the server configuration file, the following directive must be specified prior to its use:

```
LoadModule ibm_ldap_include /QSYS.LIB/QHTTPSVR.LIB/QZSRVLDAP.SRVPGM
```

**Directive**

- "LDAPInclude"

**LDAPInclude:**

**Module:** mod\_ibm\_linc

**Syntax:** LDAPInclude *filename filter attribute*

**Default:** none

**Context:** server config, virtual host, directory

**Override:** none

**Origin:** IBM

**Example:** LDAPInclude /QIBM/UserData/HTTP/HTTP/LDAP/ldap.prop (cn=web) binProperty

LDAPInclude directive is used to retrieve HTTP configuration information that is stored in an LDAP directory. The LDAP server is contacted using information from the configuration file provided, and an LDAP search is performed using the filter. Once information is returned from the LDAP search, the values of the attributes are then used as part of the HTTP configuration file.

The same filename that is used on an LDAPConfigFile directive may also be used for the LDAPInclude directive.

**Parameter One:** *filename*

- The *filename* parameter is the name of the file that contains LDAP directives required to connect to an LDAP server.

**Parameter Two:** *filter*

- The *filter* parameter is the search string that is passed from HTTP Server to the LDAP server to return an LDAP entry.

**Parameter Three:** *attribute*

- The *attribute* parameter is the name of the LDAP attribute whose value is some arbitrary part of HTTP Server configuration file.

## Module `mod_ibm_ldap`

This module supports directives that allow IBM HTTP Server for i Web servers to access an Lightweight Directory Access Protocol (LDAP) directory and to query the directory in a database fashion to obtain authentication information.

These directives provide the server with information regarding the LDAP Servers in which HTTP Server configuration (see `mod_ibm_linc`) and authentication information may be stored. You can put these directives in a file and then include that file in your server configuration file using the `LdapConfigFile` directive. If these directives are placed in the configuration file, the following directive must be specified prior to their use:

```
LoadModule ibm_ldap_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVLDAP.SRVPGM
```

### Directives

- “`ldap.AppId`” on page 370
- “`ldap.application.authType`” on page 370
- “`ldap.application.DN`” on page 371
- “`ldap.application.password.stashFile`” on page 371
- “`ldap.cache.timeout`” on page 371
- “`ldap.group.memberAttributes`” on page 372
- “`ldap.group.name.filter`” on page 372
- “`ldap.group.url`” on page 373
- “`ldap.idleConnection.timeout`” on page 373
- “`ldap.NTDomain`” on page 374
- “`ldap.ObjectClass`” on page 374
- “`ldap.realm`” on page 375
- “`ldap.search.timeout`” on page 375
- “`ldap.transport`” on page 375
- “`ldap.url`” on page 376
- “`ldap.user.authType`” on page 377
- “`ldap.user.name.fieldSep`” on page 377
- “`ldap.user.name.filter`” on page 377
- “`ldap.version`” on page 378

- “ldap.waitToRetryConnection.interval” on page 378
- “LDAPConfigFile” on page 378
- “LDAPRequire” on page 379

### ldap.AppId:

**Module:** mod\_ibm\_ldap

**Syntax:** ldap.AppId *application\_ID*

**Default:** none

**Context:** directory, .htaccess

**Override:** AuthCfg

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ldap\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVLDAP.SRVPGM

**Example:** ldap.AppId QIBM\_HTTP\_SERVER\_SRVINST1

The ldap.AppId directive is used to enable SSL connections to the LDAP server. An Application ID that has been obtained and associated with a certificate through Digital Certificate Manager (DCM ) is supplied with this directive. The application ID is then used when making an SSL connection to the LDAP server to validate that the server can make a secure connection. The Application ID provided may be the same Application ID that is used elsewhere in HTTP Server.

The ldap.AppId directive is required if ldap.transport is SSL.

**Parameter:** *application\_ID*

- The *application\_ID* parameter is an application ID obtained from DCM for this HTTP Server instance.

### ldap.application.authType:

**Module:** mod\_ibm\_ldap

**Syntax:** ldap.application.authType *authtype*

**Default:** ldap.application.authType Basic

**Context:** directory, .htaccess

**Override:** AuthCfg

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ldap\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVLDAP.SRVPGM

**Example:** ldap.application.authType None

The ldap.application.authtype directive is used to specify the method used to authenticate HTTP Server application to the LDAP server. The possible values are None and Basic.

For Basic authentication, the ldap.application.DN and the ldap.application.password.stashFile directives are required to identify HTTP Server.

**Parameter:** *authtype*

- The *authtype* parameter specifies the method used to authenticate HTTP Server application to the LDAP server. Valid values are *Basic*, or *None*.
  1. If *None* is selected, HTTP Server connects using anonymous access, if permitted by the LDAP server.
  2. If *Basic* authentication is chosen, HTTP Server is required to identify itself to the LDAP server by using a Distinguished Name and password.



## ldap.application.DN:

**Module:** mod\_ibm\_ldap

**Syntax:** ldap.application.DN *Distinguished\_Name*

**Default:** none

**Context:** directory, .htaccess

**Override:** AuthCfg

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ldap\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVLDAP.SRVPGM

**Example:** ldap.application.DN cn=Administrator

The ldap.application.DN directive specifies the Distinguished Name (DN) HTTP Server uses to authenticate to the LDAP server.

When using ldap.application.authType Basic, the directive ldap.application.password.stashFile should be used with ldap.application.DN. Unless the LDAP server allows anonymous access, the connection between HTTP Server and the LDAP server will not be made without a valid password.

**Parameter:** *Distinguished\_Name*

- The *Distinguished\_Name* parameter is a character string representing the Distinguished Name used by HTTP Server to authenticate to the LDAP server.

## ldap.application.password.stashFile:

**Module:** mod\_ibm\_ldap

**Syntax:** ldap.application.password.stashFile *filename*

**Default:** none

**Context:** directory, .htaccess

**Override:** AuthCfg

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ldap\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVLDAP.SRVPGM

**Example:** ldap.application.password.stashFile /QIBM/UserData/HTTPPA/LDAP/websrv1/lcfg1.stash

The ldap.application.password.stashFile directive specifies the file that contains the encoded password used by HTTP Server to authenticate to the LDAP server when ldap.application.authType is Basic. The configuration tools create, encode, and name the filename.

**Parameter:** *filename*

- The *filename* parameter is the name of a file containing the encoded password used to authenticate HTTP Server to the LDAP server.

## ldap.cache.timeout:

**Module:** mod\_ibm\_ldap

**Syntax:** ldap.cache.timeout *seconds*

**Default:** ldap.cache.timeout 600 (10 minutes)

**Context:** directory, .htaccess

**Override:** AuthCfg

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ldap\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVLDAP.SRVPGM

**Example:** ldap.cache.timeout 300

The `ldap.cache.timeout` directive specifies the maximum length of time (in seconds) that these cached results may be used. After `ldap.cache.timeout` seconds, the cache elements are discarded, and subsequent requests cause a search of the LDAP server. Results of a search of an LDAP server are cached in local HTTP Server storage to save the time of executing another LDAP search in a short period of time.

**Parameter:** *seconds*

- The *seconds* parameter is the length of time, in seconds, for the server to retain the results of successful LDAP searches.

### **ldap.group.memberAttributes:**

**Module:** `mod_ibm_ldap`

**Syntax:** `ldap.group.memberAttributes "attributes"`

**Default:** `ldap.group.memberAttributes "member uniquemember"`

**Context:** `directory, .htaccess`

**Override:** `AuthCfg`

**Origin:** IBM

**Usage Considerations:** A `LoadModule` is required in the configuration file prior to using the directive. The statement should be as follows: `LoadModule ibm_ldap_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVLDAP.SRVPGM`

**Example:** `ldap.group.memberAttributes "member"`

The `ldap.group.memberAttributes` directive specifies the attribute names that are used to extract members from a group entry in an LDAP directory. The values of these attributes must be the distinguished names of the members of the group.

This directive is used in conjunction with the `ldap.group.name.filter` and the `LDAPRequire` directives to allow users in specific groups access to a resource.

**Parameter One:** *attributes*

- The *attributes* parameter is the group attribute names used to extract users from an LDAP group entry. Beginning in IBM i 5.4, if the *attributes* parameter is the operational attribute `ibm-allMembers`, then group membership is checked for all forms of groups: static, dynamic, nested, and hybrid. Otherwise, group membership is checked only for a static group.

If multiple occurrences of this directive are configured in a container, only the last occurrence is processed. All other occurrences are ignored.

### **ldap.group.name.filter:**

**Module:** `mod_ibm_ldap`

**Syntax:** `ldap.group.name.filter filter`

**Default:** `ldap.group.name.filter (&(cn=%v)(!(objectclass=groupofnames)(objectclass=groupofuniquenames)))`

**Context:** `directory, .htaccess`

**Override:** `AuthCfg`

**Origin:** IBM

**Usage Considerations:** A `LoadModule` is required in the configuration file prior to using the directive. The statement should be as follows: `LoadModule ibm_ldap_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVLDAP.SRVPGM`

**Example:** `ldap.group.name.filter (&(cn=%v)(objectclass=groupofnames))`

The `ldap.group.name.filter` directive specifies the filter that is used to convert, via an LDAP search request, a group name to a unique DN. The unique DN for the group is then used to allow individual users who are members of the group to access their source. The default value is `"(&(cn=%v)(!(objectclass=groupofnames)(objectclass=groupofuniquenames)))"`, where `%v` is a substitution variable for the group name.

This directive is used in conjunction with the `ldap.group.memberAttributes` and the `LDAPRequire` directives to allow users in specific groups access to a resource.

**Parameter:** *filter*

- The *filter* parameter is a valid LDAP search filter that will return a unique DN for a given group name.

**ldap.group.url:**

**Module:** `mod_ibm_ldap`

**Syntax:** `ldap.group.url ldap://hostname:port/BaseDN`

**Default:** none

**Context:** `directory`, `.htaccess`

**Override:** `AuthCfg`

**Origin:** IBM

**Usage Considerations:** A `LoadModule` is required in the configuration file prior to using the directive. The statement should be as follows: `LoadModule ibm_ldap_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVLDAP.SRVPGM`

**Example:** `ldap.group.url ldap://www-5.ibm.com/o=deltawing,c=au`

The `ldap.group.url` directive tells HTTP Server the location of the LDAP server that is being used for authentication of users in groups. `hostname` is the hostname of the LDAP server. The DNS name or the IP address is used to identify the host where the LDAP server resides. The port is optional. If not specified, port 389 will be assumed if using TCP/IP connections, and 636 will be used for SSL connections to the LDAP server. The `BaseDN` provides the starting point for searches of the LDAP directory.

If the `ldap.group.url` is not present in the configuration file, the `ldap.url` value is used. If the same host, port and `BaseDN` are the same for group searches, as they are for user searches, you do not need to specify `ldap.group.url`.

**Parameter One:** *hostname*

- The *hostname* parameter is the DNS name or IP address of the host where the LDAP server is located.

**Parameter Two:** *port*

- The *port* parameter is the port on which the LDAP server listens. It is optional. If not present, and the transport is TCP, the well-known LDAP port 389 is assumed. If the transport is SSL, the well-known LDAP SSL port 636 will be assumed.

**Parameter Three:** *BaseDN*

- The *BaseDN* parameter is the starting point for searches of the LDAP directory for group information.

**Note:** The `ldap.group.url` value is case sensitive. For example, the following value is not valid:

`ldap.group.url ldap://www-5.ibm.com/o=deltawing,c=au`. However, the following value is valid:

`ldap.group.url ldap://www-5.ibm.com/o=deltawing,c=au`.

**ldap.idleConnection.timeout:**

**Module:** `mod_ibm_ldap`

**Syntax:** `ldap.idleConnection.timeout seconds`

**Default:** `ldap.idleConnection.timeout 600` (10 minutes)

**Context:** `directory`, `.htaccess`

**Override:** `AuthCfg`

**Origin:** IBM

**Usage Considerations:** `LoadModule` is required in the configuration file prior to using the directive. The statement should be as follows: `LoadModule ibm_ldap_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVLDAP.SRVPGM`

**Example:** ldap.idleConnection.timeout 900

The ldap.idleConnection.timeout directive is used to determine the time that idle connections to the LDAP server are kept open. This improves performance by saving the path length necessary to open connections if there are several requests of the LDAP server in a short period of time.

**Parameter:** *seconds*

- The seconds parameter is the length of time, in seconds, that an idle connection should remain open.

### **ldap.NTDomain:**

**Module:** mod\_ibm\_ldap

**Syntax:** ldap.NTDomain *domainname*

**Default:** none

**Context:** directory, .htaccess

**Override:** AuthCfg

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ldap\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVLDAP.SRVPGM

**Example:** ldap.NTDomain "cn=myexchServer"

Since Microsoft Windows NT authenticates differently than the other industry LDAP servers, this directive was added to configure the Microsoft Windows NT domain name. This directive should only be used when a Microsoft Exchange Server is being used and the authentication requires that ldap.NTDomain be specified. This directive should not be used in other cases.

Use of this directive allows an HTTP Server to access a Microsoft Exchange Server version 5.0 or 5.5 by means of Lightweight Directory Access Protocol (LDAP). It may be necessary to use this directive if this product is used to perform LDAP authentication of HTTP requests.

Directive ldap.NTDomain can be specified two different ways. The format may be dependent on the Microsoft Exchange Server.

If the Exchange Server requires the account to look like "cn=NTAccount, cn=NTDomain", use the format:

```
ldap.NTDomain "cn=exchServer"
```

If the Exchange Server requires the account in the form ("dc=NTDomain, cn=NTAccount"), use the format:

```
ldap.NTDomain "dc=exchServer"
```

When this directive is present, HTTP Server appends or precedes the information in the ldap.NTDomain directive to the DN used when authenticating a user to the LDAP server.

### **ldap.ObjectClass:**

**Module:** mod\_ibm\_ldap

**Syntax:** ldap.ObjectClass *objectclass*

**Default:** ldap.ObjectClass eProperty

**Context:** directory, .htaccess

**Override:** AuthCfg

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule IBM\_ldap\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVLDAP.SRVPGM

**Example:** ldap.ObjectClass applicationProcess

The ldap.ObjectClass directive is used to publish configuration information to the LDAP server. The object class is used as an entry to the LDAP server and describes the content and purpose of an object in the LDAP directory tree. The configuration information may then be retrieved using the LDAPInclude directive.

**Parameter:** *objectclass*

- The *objectclass* parameter is the name of the object class to be used as the entry in the LDAP directory. The object class used should have a binary file attribute value.

#### **ldap.realm:**

**Module:** mod\_ibm\_ldap

**Syntax:** ldap.realm "*label*"

**Default:** none

**Context:** directory, .htaccess

**Override:** AuthCfg

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ldap\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVLDAP.SRVPGM

**Example:** ldap.realm "HTTP Auth Server"

The ldap.realm directive is used to identify the LDAP configuration in error log messages. If a server uses different LDAP servers or different LDAP base DN's for different directories, ldap.realm will identify this particular LDAP configuration.

**Parameter:** *label*

- The *label* parameter can be a character string describing this LDAP configuration.

#### **ldap.search.timeout:**

**Module:** mod\_ibm\_ldap

**Syntax:** ldap.search.timeout *seconds*

**Default:** ldap.search.timeout 10

**Context:** directory, .htaccess

**Override:** AuthCfg

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ldap\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVLDAP.SRVPGM

**Example:** ldap.search.timeout 30

The ldap.search.timeout directive supplies the maximum amount of time (in seconds) to wait for an LDAP search request to complete. This prevents HTTP Server from waiting on a request to a slow LDAP server.

**Parameter:** *seconds*

- The *seconds* parameter is the length of time, in seconds, for the server to wait for an LDAP search request to complete.

#### **ldap.transport:**

**Module:** mod\_ibm\_ldap

**Syntax:** ldap.transport *transport*

**Default:** ldap.transport TCP

**Context:** directory, .htaccess

**Override:** AuthCfg

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: `LoadModule ibm_ldap_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVLDAP.SRVPGM`

**Example:** `ldap.transport SSL`

The `ldap.transport` directive is used to specify the transport used to communicate with the LDAP server. The LDAP server can communicate over either TCP/IP or SSL connections.

If `ldap.transport` is set to SSL, then the `ldap.AppId` directive must be set, or HTTP Server will be unable to make the connection to the LDAP server.

**Parameter:** *transport*

- The *transport* parameter specifies the transport to be used for communication with the LDAP server. Valid values are 'TCP' or 'SSL'.

### **ldap.url:**

**Module:** mod\_ibm\_ldap

**Syntax:** `ldap.url ldap://hostname:port/baseDN`

**Default:** none

**Context:** directory, .htaccess

**Override:** AuthCfg

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: `LoadModule ibm_ldap_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVLDAP.SRVPGM`

**Example:** `ldap.url ldap://www-6.ibm.com:1636/ou=Payroll,o=Company,c=US`

The `ldap.url` directive tells HTTP Server the location of the LDAP server that is being used for authentication or configuration. Hostname is the hostname of the LDAP server. The DNS name or the IP address is used to identify the host where the LDAP server resides. The port is optional. If not specified, port 389 will be assumed if using TCP/IP connections, and 636 will be used for SSL connections to the LDAP server. The BaseDN provides the starting point for searches of the LDAP directory.

This directive is required when using LDAP for authentication or configuration.

The `ldap.url` directive will be used for all searches, unless a different value is provided with the `ldap.group.url` directive. If an `ldap.group.url` directive is present, its value is used to search for groups.

**Parameter One:** *hostname*

- The *hostname* parameter is the DNS name or IP address of the host where the LDAP server is located.

**Parameter Two:** *port*

- The *port* parameter is the port on which the LDAP server listens. It is optional. If not present, and the transport is TCP, the well-known LDAP port 389 is assumed. If the transport is SSL, the well-known LDAP SSL port 636 will be assumed.

**Parameter Three:** *baseDN*

- The *baseDN* parameter is the starting point for searches of the LDAP directory.

**Note:** The `ldap.url` value is case sensitive. For example, the following value is not valid: `ldap.url Ldap://www-5.ibm.com/o=deltawing,c=au`. However, the following value is valid: `ldap.url ldap://www-5.ibm.com/o=deltawing,c=au`.

### **ldap.user.authType:**

**Module:** mod\_ibm\_ldap

**Syntax:** ldap.user.authType *authtype*

**Default:** ldap.user.authType Basic

**Context:** directory, .htaccess

**Override:** AuthCfg

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ldap\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVLDAP.SRVPGM

**Example:** ldap.user.authType Basic

The ldap.user.authtype directive is used to specify the method used to authenticate the user requesting an HTTP resource to the LDAP server. Basic is the only possible value. During basic authentication, the user is prompted to enter a username and password.

**Parameter:** *authtype*

- The *authtype* parameter specifies the method used to authenticate the user requesting an HTTP resource to the LDAP server. 'Basic' is the only valid value.

### **ldap.user.name.fieldSep:**

**Module:** mod\_ibm\_ldap

**Syntax:** ldap.user.name.fieldSep "*separators*"

**Default:** ldap.user.name.fieldSep " \t,"

**Context:** directory, .htaccess

**Override:** AuthCfg

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ldap\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVLDAP.SRVPGM

**Example:** ldap.user.name.fieldSep " \t,/"

The ldap.user.name.fieldSep directive specifies the characters that are considered valid field separator characters when parsing the user name into fields. The fields are then put into a filter and used on an LDAP search request. For example, if '/' is the only valid field separator, and the user entered "Joe Smith/Acme", then the first field is set to "Joe Smith" and the second field is set to "Acme".

**Parameter:** *separators*

- The *separators* parameter is the valid separator characters used to delimit fields.

If multiple occurrences of this directive are configured in a container, only the last occurrence is processed. All other occurrences are ignored.

### **ldap.user.name.filter:**

**Module:** mod\_ibm\_ldap

**Syntax:** ldap.user.name.filter *filter*

**Default:** ldap.user.name.filter(&(objectclass=person)(!(cn=%v1 %v2)(uid=%v1)))

**Context:** directory, .htaccess

**Override:** AuthCfg

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ldap\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVLDAP.SRVPGM

**Example:** ldap.user.name.filter (&(objectclass=person)(uid=%v1))

The `ldap.user.name.filter` directive specifies the filter that is used to convert, via an LDAP search request, a user name to a unique DN. The DN is then used to authenticate the user making the HTTP request. The default value is "`(&(objectclass=person)(l(cn=%v1 %v2)(uid=%v1))`", where `%v1` and `%v2` are substitution variables for the words the user entered at the browser.

This directive is used when `ldap.user.authType` is Basic.

**Parameter:** *filter*

- The *filter* parameter is a valid LDAP search filter that will return a unique DN for a given user name.

### **ldap.version:**

**Module:** `mod_ibm_ldap`

**Syntax:** `ldap.version version`

**Default:** `ldap.version 3`

**Context:** `directory, .htaccess`

**Override:** `AuthCfg`

**Origin:** IBM

**Usage Considerations:** A `LoadModule` is required in the configuration file prior to using the directive. The statement should be as follows: `LoadModule ibm_ldap_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVLDAP.SRVPGM`

**Example:** `ldap.version 2`

The `ldap.version` directive is used to specify the version of LDAP to use to communicate with the LDAP server. The default version used by HTTP Server is version 3. If your LDAP server is not at version 3, use this directive to set it to 2.

**Parameter:** *version*

- The *version* parameter specifies the version of the LDAP to be used. Valid versions are '2' or '3'.

### **ldap.waitForRetryConnection.interval:**

**Module:** `mod_ibm_ldap`

**Syntax:** `ldap.waitForRetryConnection.interval seconds`

**Default:** `ldap.waitForRetryConnection.interval 30`

**Context:** `directory, .htaccess`

**Override:** `AuthCfg`

**Origin:** IBM

**Usage Considerations:** A `LoadModule` is required in the configuration file prior to using the directive. The statement should be as follows: `LoadModule ibm_ldap_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVLDAP.SRVPGM`

**Example:** `ldap.waitForRetryConnection.interval 60`

If an LDAP server is down, HTTP Server may have degraded performance because it will be continually trying to connect. The `ldap.waitForRetryConnection.interval` directive gives the length of time (in seconds) to wait between failed attempts to connect to the LDAP server.

**Parameter:** *seconds*

- The *seconds* parameter is the length of time, in seconds, for the server to wait between attempts to connect to the LDAP server.

### **LDAPConfigFile:**

**Module:** `mod_ibm_ldap`

**Syntax:** `LDAPConfigFile filename`

**Default:** none



**Context:** directory, .htaccess

**Override:** AuthCfg

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ldap\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVLDAP.SRVPGM

**Example:** LDAPConfigFile /QIBM/UserData/HTTP/ldap/ldapSvr1.conf

The LDAPConfigFile directive provides a filename that contains the LDAP directives necessary to access an LDAP server. It allows the LDAP directives to be grouped into a file so they may easily be referenced in any container in HTTP Server configuration file by using the LDAPConfigFile directive. An example file can be found in /QIBM/ProdData/HTTP/conf/ldap.conf

All LDAP directives except LDAPRequire may be put into the file.

**Parameter:** *filename*

- The *filename* parameter is the filename that contains other LDAP directives.

### LDAPRequire:

**Module:** mod\_ibm\_ldap

**Syntax:** LDAPRequire *type* [*groupname* | *filter*]

**Default:** none

**Context:** directory, .htaccess

**Override:** AuthCfg

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ldap\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVLDAP.SRVPGM

**Example:** LDAPRequire filter (&(objectclass=person)(ou=Payroll)(cn=\*))

The LDAPRequire directive is used to restrict access to a resource controlled by LDAP authentication to members of a group. It can either use groups defined in LDAP by using the "group" parameter, or it can use an LDAP filter to assemble a group of users with a similar quality.

The LDAPRequire directive may not be put into an LDAP configuration file, it must be in the server configuration file. For LDAP, this can be used instead of the GroupFile directive. For more information, see the "GroupFile" on page 232 directive.

**Parameter One:** *type*

- Valid values for the *type* parameter include 'group' or 'filter'.
- Group should be used for LDAP group entries.
- Filter should be used when grouping users by other qualities.

**Parameter Two:** *groupname* | *filter*

- The *groupname* parameter is the name of a group as defined in the LDAP directory.
- The *filter* parameter is a valid filter that may be used to determine if a user meets qualifications to be authenticated.

### Module mod\_ibm\_si

Module mod\_ibm\_si supports directives for the IBM HTTP Server for i Web server.

#### Summary

This module enables the association of WebSphere Application Server instances to an HTTP Server Web server, which allows users to start and stop application servers by starting and stopping the associated Web server.

## Directives

- “AppServer”
- “WASInstance”

### AppServer:

**Module:** mod\_ibm\_si

**Syntax:** AppServer *serverName startOption endOption*

**Default:** none

**Context:** server config

**Override:** none

**Origin:** IBM

**Usage Considerations:** The server must be restarted prior to using the directive. A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule mod\_ibm\_si /QSYS.LIB/QHTTPSVR.LIB/QZISI.SRVPGM.

#### Examples:

- AppServer \*ALL start end
- AppServer server1 start end
- AppServer server1 nostart noend

The AppServer directive instructs the load module how to handle the WebSphere application servers associated with the WebSphere application server profile when the HTTP server is started and ended. More than one directive is allowed, where each directive is used to specify an application server contained in the WebSphere Application Server profile. If multiple AppServer directives is specified that contain the same server name, the last one is the directive that is used.

**Note:** This directive is only effective if directive “WASInstance” is specified in the HTTP configuration file.

#### Parameter: *serverName*

The *serverName* parameter value specifies the WebSphere application server name for which the AppServer directive applies. A special value of \*ALL indicates all servers associated with the WebSphere profile specified on the WASInstance directive.

#### Parameter: *startOption*

The *startOption* parameter value specifies whether or not to start the WebSphere application server(s) for the profile specified on the WASInstance directive when the associated Web server is started. Valid values include:

- **start** - WebSphere application server(s) are started when the associated Web server is started.
- **nostart** - WebSphere application server(s) are not started when the associated Web server is started.

#### Parameter: *endOption*

The *endOption* parameter value specifies whether or not to end the WebSphere application server(s) for the profile specified on the WASInstance directive when the associated Web server is ended. Valid values include:

- **end** - WebSphere application server(s) are ended when the associated Web server is ended.
- **noend** - WebSphere application server(s) are not ended when the associated Web server is ended.

### WASInstance:

**Module:** mod\_ibm\_si

**Syntax:** WASInstance *profilePath productID productOption*

**Default:** none

**Context:** server config

**Override:** none

**Origin:** IBM

**Usage Considerations:** The server must be restarted prior to using the directive. A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: `LoadModule mod_ibm_si /QSYS.LIB/QHTTPSVR.LIB/QZISI.SRVPGM.`

**Example:** `WASInstance /QIBM/UserData/WebSphere/AppServer/V61/Base/profiles/default 5733W61 2`

The `WASInstance` directive specifies the WebSphere application profile that is associated with the HTTP server. The `WASInstance` directive must be specified in order to specify the “AppServer” on page 380 directive.

**Parameter:** *profilePath*

The *profilePath* parameter is the path to a WebSphere application server profile.

**Parameter:** *productID*

The *ProductID* parameter is a WebSphere Application Server licensed program identifier that is associated with the specified *profilePath*.

**Parameter:** *productOption*

The *productOption* parameter is a number associated with the part of the *productID* that contains the *profilePath*.

## Module `mod_ibm_ssl`

Module `mod_ibm_ssl` supports directives for the IBM HTTP Server for i Web server.

### Configuration details

The module `mod_ibm_ssl` directives provide the server with information on the extent of the SSL authentication required for access to the server by the client. When configuring the server for SSL, it is best to use virtual hosts if the server is to be both SSL and non-SSL. The default behavior for SSL is `SSLDisable`, which causes the server to not do any SSL processing for each server or virtual host which does not specify `SSLEnable`. If SSL processing is required, then a SSL Virtual Host should be set up to handle this. The SSL port should be specified on the `<Virtual Host>` directive, with the `SSLEnable` and `SSLAppName` located inside the virtual host container. Each resource for which SSL processing is desired should be located inside the SSL virtual host container. This prevents the resource from being accessed through a non-SSL port and served when SSL is not used. If the resource is located outside the SSL virtual host container, and is located in the main server, it is still possible to access the resource through SSL. Any SSL directives are handled if the resource is requested on a SSL port, but the SSL directives, with the exception of the `SSLRequireSSL` directive, are ignored if the resource is requested on a non-SSL port. Unless the resource is configured to handle both SSL authentication and non-SSL authentication, the results in this case may not be what is desired. If a resource must be accessed only through a SSL port the `SSLRequireSSL` directive can be placed in the resource container, and any request for that resource that is received from a non-SSL port is rejected.

When configuring a resource for SSL authentication, the behavior of other directives affects how the SSL directives behave. The primary concerns are when `SSLAuthType` is configured. There are other directives that need to be set in order for SSL to behave as expected. If `SSLAuthType Cert` is specified, this tells the server to check for a certificate, and authenticate the user based on the information in that certificate. This should be the only authentication necessary for this resource. In order to ensure this, `AuthType SSL` and `Satisfy Any` needs to be configured in the resource container. This results in the desired behavior.

When configuring a resource for `SSLAuthType CertOrBasic`, this tells the server to check for a certificate and authenticate the user based on the information in that certificate. If this authentication fails, then the server authenticates the user based on any other type of authentication that is configured for that resource. In most cases, this is Basic authentication, which requests a user ID and password from the client, and the user is authenticated based on this information received from the client, but may also be

LDAP authentication if indicated in the configuration of that resource. In order for the SSLAuthType CertOrBasic to function properly, Satisfy Any, AuthType Basic, and Require needs to be configured in the resource container.

If there are CGI programs that will be using SSL, the environment variable HTTPS\_PORT must be set in the configuration file. The SetEnv HTTPS\_PORT port-number directive is used for this.

### Directives

- “SSLAppName”
- “SSLAuthType” on page 383
- “SSLCacheDisable” on page 384
- “SSLCacheEnable” on page 384
- “SSLCipherBan” on page 385
- “SSLCipherRequire” on page 386
- “SSLCipherSpec” on page 387
- “SSLClientAuth” on page 388
- “SSLClientAuthGroup” on page 388
- “SSLClientAuthRequire” on page 390
- “SSLClientCertDisable” on page 391
- “SSLClientCertEnable” on page 391
- “SSLDenySSL” on page 392
- “SSLDisable” on page 392
- “SSLEnable” on page 392
- “SSLEngine” on page 393
- “SSLProxyAppName” on page 393
- “SSLProxyCipherSpec” on page 394
- “SSLProxyEngine” on page 395
- “SSLProxyVerify” on page 395
- “SSLProxyVersion” on page 396
- “SSLRequireSSL” on page 397
- “SSLUpgrade” on page 397
- “SSLVersion” on page 398
- “SSLV2Timeout” on page 398
- “SSLV3Timeout” on page 399

### SSLAppName:

**Module:** mod\_ibm\_ssl

**Syntax:** SSLAppName *server\_application\_name*

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** IBM

**Usage Considerations:** LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ssl\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVSSL.SRVPGM

**Example:** SSLAppName QIBM\_HTTP\_SERVER\_APACHE

The SSLAppName directive is used for the following reasons:

- unique label to identify the server as an application that intends to use SSL

- to keep track of the registered name used by the server
- to identify the server when association of a server certificate with a secure application is done in the Digital Certificate Manager (DCM)
- to identify the server to the SSL API's so that the SSL API's can use the certificate that is associated with the server

This registration of the secure application and the creation of the SSLAppName is done automatically when the system administrator enables SSL for the server using the IBM Web Administration for i interface. The association of a server certificate with the application is accomplished by the system administrator using DCM. After a secure application is registered, and before attempting to start the server with SSL enabled, the user must use DCM to assign a server certificate to the corresponding secure application. Since this directive is valid at the virtual host level, the server may have more than one certificate assigned, with each virtual host having a different application name. The specified value on this directive is the name of the application that the server or virtual host is known as. If the server certificate association for the application name is not configured through DCM, then the SSL connection cannot be initialized and the server will not start.

**Note:**

Please leave the SSL fields to the default if creating an application ID via DCM for Apache Server as those settings override the same settings used by Apache server directives in the HTTP Server configuration file. There is a configured limit of 64 secure application environments (SSLAppName's) that can be active at once. To increase this limit contact customer support.

**Parameter:** *server\_application\_name*

- The *server\_application\_name* parameter value specifies the name of the application that the server or virtual host.

**SSLAuthType:**

**Module:** mod\_ibm\_ssl

**Syntax:** SSLAuthType *option*

**Default:** none

**Context:** directory, .htaccess

**Override:** AuthConfig

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ssl\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVSSL.SRVPGM

**Example:** SSLAuthType cert

The SSLAuthType directive is used to specify the type certificate validation/authentication required for access to a directory. This option is used to ensure that a certificate received from the client is associated with a user ID or an Internet User validation list. If this is not the case, the client may be prompted for the user ID.

**Parameter:** *option*

- The *option* parameter value can be one of the following:

*Cert* This option indicates to the server that the certificate received from the client must be in an Internet User list or be associated with an IBM i user ID convention. Note : If SSLAuthType Cert is specified, then AuthType should be set to SSL.

*CertOrBasic*

This option indicates to the server that the certificate, if there is one, that is received from the client may be associated with a user ID or may be in an Internet User validation list. If it is not, then the client is authenticated based on

the value of HTTP Server AuthType directive. In order to simulate HTTP Server (original) behavior of AuthType CertOrBasic , HTTP Server AuthType directive must be Basic. This will cause the client to be prompted for a user ID and password, and this provided user ID and password will then be used to access the directory/file. If SSLAuthType CertOrBasic is used, then AuthType should be set to Basic.

The certificate does not need to be valid. This directive only refers to the existence of a certificate. If the certificate must be valid, then the SSLClientCertEnable directive must also be specified.

There are no default values for this directive. If the directive is not used, then if a certificate is present, association with a user ID or Internet User validation list is not checked. This directive's scope is the directory level. This directive is only to be specified once for a directory. Any subsequent uses of this directive override any previously specified values.

This directive may be used in conjunction with the SSLClientCertEnable directive. This will cause very specific behavior to occur, depending on the value specified on the SSLAuthType directive. If the SSLClientCert directive is used in addition to SSLAuthType Cert, the certificate received from the client must be valid, as well as associated with a user ID or in an Internet User validation list. If the SSLClientCert directive is used in addition to SSLAuthType CertOrBasic, a certificate must be received from the client, but does not need to be associated with a user ID or in an Internet User validation list. If the association is not present, the client will be authenticated based on the protection setup (basic or ldap).

This directive also interacts with the PasswdFile directive. This directive is used to help determine the type of certificate authentication to be used. If the PasswdFile directive is set to %%SYSTEM%%, then the certificate received from the client must be associated with an IBM i user profile in order for it the client to be authenticated. If the PasswdFile directive is set to an internet user list, then the certificate received must be in the internet user list in order for the client to be authenticated. Again, this authentication is only required if the Cert option is selected on the SSLAuthType directive. Otherwise it is only optional.

#### **SSLCacheDisable:**

**Module:** mod\_ibm\_ssl

**Syntax:** SSLCacheDisable

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ssl\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVSSL.SRVPGM

**Example:** SSLCacheDisable

The SSLCacheDisable directive will cause SSL session ID caching to be disabled. The effect of this directive will depend on the location of the directive. If the directive is located in the configuration file for the main server, SSL session ID caching will not be done for the server. If the directive is located in a <Virtual Host> container, then SSL session ID caching will not be done for the virtual host. The directive located at the server level can be overridden for a particular virtual host using the SSLCacheEnable directive. Directives SSLV2Timeout and SSLV3Timeout will be ignored when SSLCacheDisable is set.

**Note:** This directive does not contain parameters.

#### **SSLCacheEnable:**

**Module:** mod\_ibm\_ssl

**Syntax:** SSLCacheEnable

**Default:** SSLCacheEnable

**Context:** server config, virtual host

**Override:** none

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ssl\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVSSL.SRVPGM

**Example:** SSLCacheEnable

The SSLCacheEnable directive will cause SSL session ID caching to be enabled. The effect of this directive will depend on the location of the directive. If the directive is located in the configuration file for the main server, SSL session ID caching will be done for the server. If the directive is located in a <Virtual Host> container, then SSL session ID caching will be done for the virtual host. The directive located at the server level can be overridden for a particular virtual host using the SSLCacheDisable directive. A abbreviated handshake will be done whenever a handshake is necessary. Directives SSLV2Timeout and SSLV3Timeout will be ignored.

**Note:** This directive does not contain parameters.

### SSLCipherBan:

**Module:** mod\_ibm\_ssl

**Syntax:** SSLCipherBan *string*

**Default:** none

**Context:** directory, .htaccess

**Override:** AuthConfig

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ssl\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVSSL.SRVPGM

**Example:** SSLCipherBan 3A

**Example:** SSLCipherBan SSL\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA

The SSLCipherBan directive allows for banning access to a directory based on the cipher that is negotiated during the SSL handshake. A set of ciphers can either be defaulted or specified using the SSLCipherSpec directive. The cipher list then can be shortened for a specific directory. This directive will enforce a greater level of security through the use of cipher specs.

The SSLCipherBan directive will directly interact with the SSLCipherRequire directive. If a negotiated cipher is listed on the ban list, then the request will be rejected, even if the cipher is also on the require list.

**Parameter:** *string*

- The *string* parameter value specifies the cipher to be used. Either the short name or the long name in the table below may be specified.

Table 19. Version 2 ciphers

Long name	Short name
SSL_WITH_3DES_EDE_CBC_MD5	27
SSL_WITH_RC4_128_MD5	21
SSL_WITH_RC2_CBC_128_MD5	23
SSL_WITH_DES_CBC_MD5	26
SSL_EXPORT_WITH_RC4_40_MD5	22
SSL_EXPORT_WITH_RC2_CBC_40_MD5	24

Table 20. Version 3 and TLS ciphers

Long name	Short name
SSL_RSA_WITH_3DES_EDE_CBC_SHA	3A
SSL_RSA_WITH_RC4_128_SHA	35
SSL_RSA_WITH_RC4_128_MD5	34
SSL_RSA_WITH_DES_CBC_SHA	39
SSL_RSA_EXPORT_WITH_RC4_40_MD5	33
SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5	36
SSL_RSA_WITH_NULL_SHA	32
SSL_RSA_WITH_NULL_MD5	31

### SSLCipherRequire:

**Module:** mod\_ibm\_ssl

**Syntax:** SSLCipherRequire *string*

**Default:** none

**Context:** directory, .htaccess

**Override:** AuthConfig

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ssl\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVSSL.SRVPGM

**Example:** SSLCipherRequire "27"

**Example:** SSLCipherRequire SSL\_WITH\_3DES\_EDE\_CBC\_MD5

The SSLCipherRequire directive allows for the user to require that certain ciphers to be negotiated with the client during the SSL handshake. Specifying that a subset of ciphers are required will force a greater level of security for a particular directory which may not be required for all directories. The ciphers listed here may or may not be listed using the SSLCipherSpec directive.

**Parameter:** *string*

- The *string* parameter value specifies the cipher to be used. Either the short name or the long name in the table below may be specified.

Table 21. Version 2 ciphers

Long name	Short name
SSL_WITH_3DES_EDE_CBC_MD5	27
SSL_WITH_RC4_128_MD5	21
SSL_WITH_RC2_CBC_128_MD5	23
SSL_WITH_DES_CBC_MD5	26
SSL_EXPORT_WITH_RC4_40_MD5	22
SSL_EXPORT_WITH_RC2_CBC_40_MD5	24

Table 22. Version 3 and TLS ciphers

Long name	Short name
SSL_RSA_WITH_3DES_EDE_CBC_SHA	3A
SSL_RSA_WITH_RC4_128_SHA	35
SSL_RSA_WITH_RC4_128_MD5	34



Table 22. Version 3 and TLS ciphers (continued)

Long name	Short name
SSL_RSA_WITH_DES_CBC_SHA	39
SSL_RSA_EXPORT_WITH_RC4_40_MD5	33
SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5	36
SSL_RSA_WITH_NULL_SHA	32
SSL_RSA_WITH_NULL_MD5	31

**Note:** The short and long names can be quoted. For example, SSLCipherRequire "SSL\_WITH\_3DES\_EDE\_CBC\_MD5".

**SSLCipherSpec:**

**Module:** mod\_ibm\_ssl

**Syntax:** SSLCipherSpec *string*

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ssl\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVSSL.SRVPGM

**Example:** SSLCipherSpec "3A"

Text

**Parameter:** *string*

- The *string* parameter value specifies the cipher to be used. Either the short name or the long name in the table below may be specified.

Table 23. Version 2 ciphers

Long name	Short name
SSL_WITH_3DES_EDE_CBC_MD5	27
SSL_WITH_RC4_128_MD5	21
SSL_WITH_RC2_CBC_128_MD5	23
SSL_WITH_DES_CBC_MD5	26
SSL_EXPORT_WITH_RC4_40_MD5	22
SSL_EXPORT_WITH_RC2_CBC_40_MD5	24

Table 24. Version 3 and TLS ciphers

Long name	Short name
SSL_RSA_WITH_3DES_EDE_CBC_SHA	3A
SSL_RSA_WITH_RC4_128_SHA	35
SSL_RSA_WITH_RC4_128_MD5	34
SSL_RSA_WITH_DES_CBC_SHA	39
SSL_RSA_EXPORT_WITH_RC4_40_MD5	33
SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5	36
SSL_RSA_WITH_NULL_SHA	32

Table 24. Version 3 and TLS ciphers (continued)

Long name	Short name
SSL_RSA_WITH_NULL_MD5	31

The order of the SSLCipherSpec directives is important. The cipher suite list passed to SSL is created by putting the first cipher listed in the configuration file at the top of the cipher suite list. SSL uses this list as the preferred order of ciphers.

This directive works in conjunction with the SSLVersion directive during the SSL handshake. The values specified for the SSLCipherSpec directive must correspond with the value specified on the SSLVersion directive. If this directive is not used, a default cipher suite list is used.

**Note:** The short and long names can be quoted. For example, SSLCipherSpec "3A".

### SSLClientAuth:

**Module:** mod\_ibm\_ssl

**Syntax:** SSLClientAuth *type*

**Default:** SSLClientAuth none

**Context:** server config, virtual host

**Override:** none

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ssl\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVSSL.SRVPGM

**Example:** SSLClientAuth 2

The SSLClientAuth directive is used to indicate the type of client-side SSL certificate validation is required for the server.

**Parameter:** *type*

- The *type* parameter value specifies the client-side SSL certificate validation required for the server. Valid values include:

*0* or *none*

No client certificate is required.

*1* or *optional*

The client may present a valid certificate.

*2* or *required*

The client must present a valid certificate.

The default value of this directive is *0*, or *none*, indicating that no certificate is requested or required from the client. If an incorrect value is specified, an error message is issued and the server will not start. A value of *1*, or *optional*, will cause the server to request a certificate from the client, and the SSL connection will be made even if a certificate is not received. A value of *1* does not require the certificate received from the client to be valid. A value of *2*, or *required*, will cause the server to request a certificate from the client. If a valid certificate is not received, the client request will be rejected.

### SSLClientAuthGroup:

**Module:** mod\_ibm\_ssl

**Syntax:** SSLClientAuthGroup *groupname attribute-expression*

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ssl\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVSSL.SRVPGM

**Example:** SSLClientAuthGroup IBMpeople Org = IBM

The SSLClientAuthGroup directive is used to define a group name to a set of specific client certificate attributes to be used on the SSLClientAuthRequire directive. To indicate the attributes, a validated certificate must be presented before the server will allow access to the directory.

**Parameter One:** *groupname*

- The *groupname* parameter value specifies the group name for the client certificate. A group name cannot include spaces.

**Parameter Two:** *attribute-expression*

- The *attribute-expression* parameter value specifies the attribute for a validated certificate to be used for client authentication. Either the long name or the short name may be used in this directive. Valid values include:

Table 25. Attribute values

Long name	Short name
IssuerStateOrProvince	IST
IssuerCommonName	ICN
IssuerOrgUnit	IOU
IssuerCountry	IC
IssuerLocality	IL
IssuerOrg	IO
IssuerEmail	IE
IssuerPostalCode	IPC
StateOrProvince	ST
CommonName	CN
OrgUnit	OU
Country	C
Locality	L
Org	O
PostalCode	PC
SerialNumber	SN

**Note:** The short and long names can be quoted. For example, SSLClientAuthGroup IBMpeople "Org = IBM".

The user specifies a logic string of specific client certificate attributes and a group name is assigned to these attributes. Multiple subexpressions can be logically ANDed , ORed, or NOTed to configure the desired group of client certificate attributes. Valid equalities include '=' and '!='.

**Example One**

```
SSLClientAuthGroup IBMpeople Org=IBM
```

**Example Two**

```
SSLClientAuthGroup MNIBM ST=MN && Org=IMB
```

## SSLClientAuthRequire:

**Module:** mod\_ibm\_ssl

**Syntax:** SSLClientAuthRequire attribute-expression

**Default:** none

**Context:** directory, .htaccess

**Override:** AuthConfig

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ssl\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVSSL.SRVPGM

**Example:** SSLClientAuthRequire group != IBMpeople && ST= MN

The SSLClientAuthRequire directive is used to provide a specific client certificate attributes, or groups of attributes, that must be validated before the server will allow access to the directory. If the certificate received does not have a particular attribute, then we do not check for an attribute match. Even if the matching value is " ", this may still not be the same as not having the attribute there at all. Any attribute specified on the SSLClientAuthRequire and not available on the certificate causes the request to be rejected.

The following is a list of the attribute values that may be specified on this directive:

*Table 26. Attribute values*

Long name	Short name
IssuerStateOrProvince	IST
IssuerCommonName	ICN
IssuerOrgUnit	IOU
IssuerCountry	IC
IssuerLocality	IL
IssuerOrg	IO
IssuerEmail	IE
IssuerPostalCode	IPC
StateOrProvince	ST
CommonName	CN
OrgUnit	OU
Country	C
Locality	L
Org	O
PostalCode	PC
SerialNumber	SN

Either the long name or the short name may be used in this directive.

The user specified a logic string of specific client certificate attributes. Multiple subexpressions can be logically ANDed, ORed, or NOTed to configure the desired client certificate attributes. Valid logical symbols include '=' and '!='. The user may also specify a group name, configured on the SSLClientAuthGroup, that allows a group of attributes to be configured.

Multiple SSLClientAuthRequire directives may be specified for each directory, and each attribute specified is used to check the attributes in the client certificate. Multiple directives place a logical AND on the attributes specified with the directives.

**Example 1:** SSLClientAuthRequire (CommonName="John Doe" || StateOrProvince=MN) && Org !=IBM

**Example 2:** SSLClientAuthRequire group!=IBMpeople && ST=MN

**Note:** The short and long names can be quoted. For example, SSLClientAuthRequire group != IBMpeople && "ST= MN"

### SSLClientCertDisable:

**Module:** mod\_ibm\_ssl

**Syntax:** SSLClientCertDisable

**Default:** none

**Context:** directory, .htaccess

**Override:** AuthConfig

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ssl\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVSSL.SRVPGM

**Example:** SSLClientCertDisable

The SSLClientCertDisable directive indicates to the server that a valid certificate is not required in order to access this directory.

This directive may be used in conjunction with the SSLAuthType directive. If specified in addition to the SSLAuthTypeCert directive, the certificate received only needs to be associated with a user ID or an Internet user.

This directive negates the SSLClientCertEnable directive.

### SSLClientCertEnable:

**Module:** mod\_ibm\_ssl

**Syntax:** SSLClientCertEnable

**Default:** none

**Context:** directory, .htaccess

**Override:** AuthConfig

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ssl\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVSSL.SRVPGM

**Example:** SSLClientCert Enable

The SSLClientCertEnable directive indicates to the server that a valid certificate is required in order to access this directory.

This directive may be used in conjunction with the SSLAuthType directive.

If specified in addition to the SSLAuthTypeCert directive, the certificate received needs to be valid, as well as associated with a user ID or an Internet user. This directive is negated by the SSLClientCertDisable directive.

## SSLDenySSL:

**Module:** mod\_ibm\_ssl

**Syntax:** SSLDenySSL

**Default:** none

**Context:** directory, .htaccess

**Override:** AuthConfig

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ssl\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVSSL.SRVPGM

**Example:** SSLDenySSL

The SSLDenySSL directive will deny access to the directory when SSL is used for the request. This directive interacts somewhat with the SSLRequireSSL directive. If a directory has both the SSLRequireSSL and the SSLDenySSL directives specified, then the last directive in the directory scope will take effect. Since this directive is scoped to a directory, a server or a virtual host may also have SSLRequireSSL for some directories, but SSLDenySSL for other directories. Also, more specific directory container directives will override previously specified directives for a less specific directory.

### Example:

```
<Directory /ABC>
  SSLRequireSSL
</Directory>
<Directory /ABC/DEF>
  SSLDenySSL
</Directory>
```

This example will require SSL for directory /ABC, but deny SSL for directory /ABC/DEF.

## SSLDisable:

**Module:** mod\_ibm\_ssl

**Syntax:** SSLDisable

**Default:** SSLDisable

**Context:** server config, virtual host

**Override:** none

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ssl\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVSSL.SRVPGM

**Example:** SSLDisable

The SSLDisable directive causes SSL to be disabled for the server or virtual host. The effect of this directive will depend on the location of the directive. If the directive is located in the configuration file for the main server, SSL will not be allowed for the server. If the directive is located in a <Virtual Host> container, then SSL will not be allowed for the virtual host. The directive located at the server level can be overridden for a particular virtual host using the SSLEnable directive.

## SSLEnable:

**Module:** mod\_ibm\_ssl

**Syntax:** SSLEnable

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ssl\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVSSL.SRVPGM

**Example:** SSLEnable

The SSLEnable directive will cause SSL to be enabled. The effect of this directive will depend on the location of the directive. If the directive is located in the configuration file for the main server, SSL will be required for the server. If the directive is located in a <Virtual Host> container, then SSL will be required for the virtual host. The directive, located at the server level, can be overridden for a particular virtual host using the SSLDisable directive. This directive requires that the directive SSLAppName be set.

**Note:** Some applications need SetEnv HTTPS\_PORT <port> configured when SSLEnable is configured.

### SSLEngine:

**Module:** mod\_ibm\_ssl

**Syntax:** SLEngine *On* | *Off* | *Optional*

**Default:** SSLEngine Off

**Context:** server, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** The server must be restarted prior to using the directive. A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ssl\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVSSL.SRVPGM

**Example:** SSLEngine On

The SSLEngine directive toggles the usage of SSL processing. If SSLEngine *On* is specified, SSL processing is enabled. If SSLEngine *Off* is specified, SSL processing is disabled. If SSLEngine *Optional* is specified, SSL processing is turned on to handle upgrading a non-SSL connection to an SSL connection. The effect of this directive depends on the location of the directive. If the directive is located in the configuration file for the main server, the type of SSL processing is set for the entire server. If the directive is located in a <VirtualHost> container, then the type of SSL processing is set for only that virtual host. If this directive is set at the server level, it can be overridden for a particular virtual host by specifying the other allowed option. SSLEngine *On* is equivalent to SSLEnable, SSLEngine *Off* is equivalent to SSLDisable, and SSLEngine *Optional* is equivalent to SSLUpgrade. These directives can be used interchangeably. The SSLEngine directive is being added in order to be compatible with Apache's mod\_ssl.

If SSLEngine *On* or SSLEngine *Optional* is configured, the directive SSLAppName must also be configured.

See also SSLEnable, SSLDisable, SSLUpgrade, and SSLAppName.

**Parameter:** *seconds*

- The *seconds* parameter has a valid value range of 1 to 86400 seconds. If the value specified is greater than 86400, or less than 1, then the default value of 86400 seconds will be used as the timeout value. This value is used for negotiated SSLVersion 3, or TLS Version 1, sessions.

### SSLProxyAppName:

**Module:** mod\_ibm\_ssl

**Syntax:** SSLProxyAppName *server\_application\_name*

**Default:** none

**Context:** server, virtual host

**Override:** none

**Origin:** IBM

**Usage Considerations:** The server must be restarted prior to using the directive.

**Example:** SSLProxyAppName QIBM\_HTTP\_CLIENT\_APACHE

The SSLProxyAppName directive is used to:

- to uniquely label the proxy server as a client application that intends to use SSL to a remote content server.
- to keep track of the registered name used by the proxy server.
- to identify the server when association of a client certificate with a secure application is done in the Digital Certificate Manager (DCM).
- to identify the server to the SSL API's so that the SSL API's can use the certificate that is associated with the server.

The registration of the secure client application and the creation of the SSLProxyAppName is done automatically when the system administrator enables the SSL Proxy engine for the server using the HTTP Server configuration GUI. The association of a client certificate with the application is accomplished by the system administrator using DCM: after a secure client application is registered, and before attempting to start the server with the SSL proxy engine enabled and SSLProxyAppName configured, the user must use DCM to assign a client certificate to the corresponding secure application. Since this directive is valid at the virtual host level, the server may have more than one certificate assigned, with each virtual host having a different application name. The specified value on this directive is the name of the application that the server or virtual host is known as. If both the SSLProxyAppName directive and the SSLProxyMachineCertificateFile directive are configured for the server, then the SSLProxyAppName directive is used to identify the client certificate and the handshake processing.

**Note:** Please leave the SSL fields to the default if creating an application ID via DCM for Apache Server as those settings override the same settings used by Apache server directives in the HTTP Server configuration file.

#### | **SSLProxyCipherSpec:**

| **Module:** mod\_ibm\_ssl

| **Syntax:** SSLProxyCipherSpec *cipher-spec*

| **Default:** none

| **Context:** server config, virtual host

| **Override:** none

| **Origin:** Modified

| **Usage Considerations:** The server must be restarted prior to using the directive. A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ssl\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVSSL.SRVPGM.

| **Example:** SSLProxyCipherSpec "3A"

| The SSLProxyCipherSpec directive allows for specifying a cipher specification to be used for the SSL connection. Each occurrence of this directive will add the associated cipher specification to that context's existing cipher suite list. The cipher specification is used on the SSL handshake, which then uses the cipher suite list to negotiate the cipher used for communications between the proxy server and the content server.

| The order of the SSLProxyCipherSpec directives is important. The cipher suite list passed to SSL is created by putting the first cipher listed in the configuration file at the top of the cipher suite list. SSL uses this list as the preferred order of ciphers.

| This directive works in conjunction with the SSLProxyVersion directive during the SSL handshake. The values specified for the SSLProxyCipherSpec directive must correspond with the value specified on the SSLProxyVersion directive. If this directive is not used, a default cipher suite list is used.



**Parameter:** *cipher-spec*

- The *cipher-spec* parameter specifies the cipher specification to be used. Either the short name or the long name in the following table may be specified.

Table 27. SSLProxyCipherSpec directive values

Cipher specifications	Long name	Short name
Version 2 ciphers	SSL_WITH_3DES_EDE_CBC_MD5	27
	SSL_WITH_RC4_128_MD5	21
	SSL_WITH_RC2_CBC_128_MD5	23
	SSL_WITH_DES_CBC_MD5	26
	SSL_EXPORT_WITH_RC4_40_MD5	22
	SSL_EXPORT_WITH_RC2_CBC_40_MD5	24
Version 3 and TLS ciphers	SSL_RSA_WITH_3DES_EDE_CBC_SHA	3A
	SSL_RSA_WITH_RC4_128_SHA	35
	SSL_RSA_WITH_RC4_128_MD5	34
	SSL_RSA_WITH_NULL_SHA	32
	SSL_RSA_WITH_NULL_MD5	31
	SSL_RSA_WITH_DES_CBC_SHA	39
	SSL_RSA_EXPORT_WITH_RC4_40_MD5	33
	SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5	36

**Note:** The short and long names can be quoted. For example, SSLProxyCipherSpec "SSL\_WITH\_DES\_CBC\_MD5".

### SSLProxyEngine:

**Module:** mod\_ibm\_ssl

**Syntax:** SSLProxyEngine *On* | *Off*

**Default:** SSLProxyEngine *Off*

**Context:** server, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** The server must be restarted prior to using the directive. This directive requires that either the SSLProxyAppName directive or the SSLProxyMachineCertificateFile be configured. Use of the SSLProxyMachineCertificateFile directive is required if the remote content server does not require a client certificate to be sent by the proxy server during the handshake process. If a certificate will be required by the remote content server, then the SSLProxyAppName should be used to identify the client certificate to use on the handshake.

**Example:** SSLProxyEngine On

The SSLProxyEngine directive toggles the usage of SSL connections to be used by the proxy to connect to the content server. This is usually used inside a <VirtualHost> section to enable SSL/TLS for proxy usage in a particular virtual host.

### SSLProxyVerify:

**Module:** mod\_ibm\_ssl

**Syntax:** SSLProxyVerify | 1 | *Optional* | 2 | *Required*

**Default:** SSLProxyVerify *Required*

**Context:** server, virtual host

**Override:** none

**Origin:** Apache

**Example:**

1. SSLProxyVerify 2
2. SSLProxyVerify *Required*

The SSLProxyVerify directive is used to indicate the type of server-side SSL certificate validation is required by the proxy server. The following values are valid for the SSLProxyVerify directive:

- (1 or Optional) - The content server may present a valid certificate.
- (2 or Required) - The content server must present a valid, trusted certificate.

The default value of this directive is 2 or Required, indicating that the content server certificate must be valid and have a trusted root. If an incorrect value is specified, an error message is issued and the server will not start.

The proxy server requires a certificate to be received from the content server. However, this certificate may be expired, or not be trusted by the server CA, as configured on the SSLProxyAppName directive or the SSLProxyMachineCertificatePath directive. This will result in a handshake failure if 2 or Required is configured.

A value of 1 or Optional, will cause the proxy server to allow for an expired content server certificate, or allow for the content server certificate to not be trusted by the server application ID configured. This will result in the handshake completing successfully.

**SSLProxyVersion:**

**Module:** mod\_ibm\_ssl

**Syntax:** SSLProxyVersion SSLV2 | SSLV3 | TLSV1 | TLSV1\_SSLV3 | ALL

**Default:** SSLProxyVersion ALL

**Context:** server, virtual host

**Override:** none

**Origin:** Modified

**Example:** SSLVersion TLSV1

The SSLProxyVersion directive specifies the SSL version that is negotiated with the remote content server during the SSL agreement that takes place when connecting the Apache proxy server to the content server via the SSL protocol. The version specified must be negotiated or access to content server is denied.

There are five possible values for this directive:

**SSLV2** SSL Version 2.0 only

**SSLV3** SSL Version 3.0 only

**TLSV1**

TLS Version 1.0 only

**TLSV1\_SSLV3**

TLS Version 1.0 with SSL V3.0 compatibility

**ALL (default)**

TLS Version 1.0 with SSLV2.0 & SSL V3.0 compatibility

The server defaults to ALL indicating that the server accepts any version that is negotiated.

## SSLRequireSSL:

**Module:** mod\_ibm\_ssl

**Syntax:** SSLRequireSSL

**Default:** none ( if neither SSLRequireSSL or SSLDenySSL are configured, the client may access the container using a secure or non-secure connection)

**Context:** directory, .htaccess

**Override:** none

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ssl\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVSSL.SRVPGM

**Example:** SSLRequireSSL

The SSLRequireSSL directive will deny access to the directory whenever SSL is not used for the request. This is used to ensure that the client uses the SSL protocol to access a directory, and helps protect the resources in the directory from being accessed, even though there may be errors in the server configuration.

This directive interacts with the SSLDenySSL directive. If a directory has both the SSLRequireSSL and the SSLDenySSL directives specified, the last directive in the directory scope will take effect. Since this directive is scoped to a directory, a server or a virtual host may also have SSLRequireSSL for some directories, but SSLDenySSL for other directories. Also, more specific directory container directives will override previously specified directives for a less specific directory.

### Example:

```
<Directory /ABC>
  SSLRequireSSL
</Directory>
<Directory /ABC/DEF>
  SSLDenySSL
</Directory>
```

This example will require SSL for directory /ABC, but deny SSL for directory /ABC/DEF.

## SSLUpgrade:

**Module:** mod\_ibm\_ssl

**Syntax:** SSLUpgrad

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ssl\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVSSL.SRVPGM

**Example:** SSLUpgrade

The SSLUpgrade directive enables a server to support a client request to upgrade a normal non-SSL connection to a Transport Layer Security (TLS) connection (for a single request). This directive's effectiveness will depend on the directive location. If the directive is located in the main server configuration file, any connection to the server will be eligible for a TLS upgrade. If the directive is located in a <Virtual Host> container, only the connection to that virtual host will be eligible for the upgrade. The directive, located at the server level, can be overridden for a particular virtual host using the SSLDisable or SSLEnable directives. SSLUpgrade requires that the directive SSLAppName is defined.

The SSLVersion directive is affected by SSLUpgrade. If SSLUpgrade is configured, the SSLVersion that is negotiated on the handshake will only be TLS. The SSLVersion specified in the configuration file will be ignored.

The SSLCipherSpec directive is also affected by SSLUpgrade. If SSLUpgrade is configured, only SSLV3/TLS ciphers are allowed. If SSLCipherSpec specifies SSL version 2 ciphers, these ciphers will be ignored, and only configured SSLV3/ TLS ciphers will be allowed. If there are no SSLV3/TLS ciphers configured, the defined default system cipher list will be used.

The SSLRequireSSL directive may be configured for a resource that is accessed through an upgraded connection. If the upgrade is requested as a part of the request through the use of the upgrade header, the SSLRequireSSL directive will be enforced before the connection is upgraded. This will allow the request to be processed, since the connection will be upgraded to SSL before the request has been handled, and the reply has been sent.

The SSLDenySSL directive will be enforced in the same manner as the SSLRequireSSL directive. If the request for the resource is received along with the upgrade header request, the request will be denied with a 403, Forbidden, response returned to the client, since the request will be processed after the connection has been upgraded.

**SSLVersion:**

**Module:** mod\_ibm\_ssl

**Syntax:** SSLVersion *version*

**Default:** SSLVersion ALL

**Context:** server config, virtual host

**Override:** none

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ssl\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVSSL.SRVPGM

**Example:** SSLVersion TLSV1

he SSLVersion directive specifies the SSL version that will be negotiated with the client during the SSL handshake. The version specified must be negotiated or access to specified resource will be denied.

There are five possible values for this directive:

*Table 28. Directive values*

Value	Description
SSLV2	SSL Version 2.0 only
SSLV3	SSL Version 3.0 only
TLSV1	TLS Version 1.0 only
TLSV1_SSLV3	TLS Version 1.0 with SSL Version 3.0 compatibility
ALL	TLS Version 1.0 with SSL Version 2.0 and SSL Version 3.0 compatability

The server will default to ALL indicating that the server will accept any version that is negotiated.

**SSLV2Timeout:**

**Module:** mod\_ibm\_ssl

**Syntax:** SSLV2Timeout *seconds*

**Default:** SSLV2Timeout 100

**Context:** server config, virtual host

**Override:** none

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ssl\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVSSL.SRVPGM

**Example:** SSLV2Timeout 32

The SSLV2Timeout directive specifies the timeout value for the session ID caching done by sockets that will be used on the SSL session. This directive indicates the number of seconds in which the internal SSL session identifier will expire. The session identifier is maintained by sockets. It allows caching of handshake information in order to allow for a shortened handshake to be done if the timeout value has not been reached. Lower values are safer but slower, because the complete handshake will be done after each timeout. If client certificates are being requested by the server, they will also be required to be represented at each timeout.

**Parameter:** *seconds*

- The *seconds* parameter has a valid value range of 1 to 100 seconds. If the value specified is greater than 100, or less than 1, then the default value of 100 seconds will be used as the timeout value. This value is used for negotiated SSL Version 2 sessions.

### SSLV3Timeout:

**Module:** mod\_ibm\_ssl

**Syntax:** SSLV3Timeout *seconds*

**Default:** SSLV3Timeout 86400

**Context:** server config, virtual host

**Override:** none

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule ibm\_ssl\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVSSL.SRVPGM

**Example:** SSLV3Timeout 32

The SSLV3Timeout directive specifies the timeout value for the session ID caching done by sockets that will be used on the SSL session. This directive indicates the number of seconds in which the internal SSL session identifier will expire. The session identifier is maintained by sockets, and allows caching of handshake information in order to allow for a shortened handshake to be done if the timeout value has not been reached. Lower values are safer, but also slower, as the complete handshake will be done after each timeout. If client certificates are being requested by the server, they will also be required to be represented at each timeout.

**Parameter:** *seconds*

- The *seconds* parameter has a valid value range of 1 to 86400 seconds. If the value specified is greater than 86400, or less than 1, then the default value of 86400 seconds will be used as the timeout value. This value is used for negotiated SSLVersion 3, or TLS Version 1, sessions.

## Module mod\_imap

Module mod\_imap supports directives for the IBM HTTP Server for i Web server.

### Summary

The module mod\_imap provides for .map files, replacing the functionality of the imagemap CGI program. Any directory or document type configured to use the handler imap-file (using either AddHandler or SetHandler) will be processed by this module. This module is in the default HTTP Server distribution. The following directive will activate files ending with Map as imagemap files:

```
AddHandler imap-file map
```

**Note:** The following is still supported:

AddType application/x-httpd-imap map

### Features

- URL references relative to the Referer: information
- Default <BASE> assignment through a new map directive base
- No need for imagemap.conf file
- Point references
- Configurable generation of imagemap lists

See “Additional information on Imagemap files” on page 401 for more information on Imagemaps.

### Directives

- ImapBase
- ImapDefault
- ImapMenu

#### ImapBase:

**Module:** mod\_imap

**Syntax:** ImapBase *map* | *referer* | *URL*

**Default:** ImapBase map

**Context:** server config, virtual host, directory, .htaccess

**Override:** Indexes

**Origin:** Apache

**Example:** ImapBase map

The ImapBase directive sets the default base used in the imagemap files. Its value is overridden by a base directive within the imagemap file. If not present, the base defaults to `http://servername/`.

**Parameter:** *map* | *referer* | *URL*

- The *map* parameter is equivalent to the URL of the imagemap file itself. No coordinates are sent with this, so a list will be generated unless ImapMenu is set to none.
- The *referer* parameter is equivalent to the URL of the referring document. Defaults to `http://servername/` if no Referer.
- The *URL* parameter can be relative or absolute. Relative URLs can contain `'..'` syntax and will be resolved relative to the base value. The base value itself will not be resolved according to the current value. The statement `base mailto:` will work properly, though.

#### ImapDefault:

**Module:** mod\_imap

**Syntax:** ImapDefault *error* | *nocontent* | *map* | *referer* | *URL*

**Default:** ImapDefault nocontent

**Context:** server config, virtual host, directory, .htaccess

**Override:** Indexes

**Origin:** Apache

**Example:** ImapDefault nocontent

The ImapDefault directive sets the default used in the imagemap files. Its value is overridden by a default directive within the imagemap file. If not present, the default action is nocontent, which means that a 204 No Content is sent to the client. In this case, the client should continue to display the original page.

**Parameter:** *error* | *nocontent* | *map* | *referer* | *URL*

- The *error* parameter fails with a 500 Server Error. Valid for all but base , but sort of useless for anything but default.
- The *nocontent* parameter sends a status code of 204 No Content, telling the client to keep the same page displayed. Valid for all but base.
- The *map* parameter is equivalent to the URL of the imagemap file itself. No coordinates are sent with this, so a list will be generated unless *ImapMenu* is set to none.
- The *referer* parameter is equivalent to the URL of the referring document. Defaults to `http://servername/` if no *Referer*.
- The *URL* parameter can be relative or absolute. Relative URLs can contain `'.'` syntax and will be resolved relative to the base value . The base value itself will not be resolved according to the current value. However, the statement `base mailto:` will work properly.

## **ImapMenu:**

**Module:** `mod_imap`

**Syntax:** `ImapMenu none | formatted | semiformatted | unformatted`

**Default:** `ImapMenu formatted`

**Context:** server config, virtual host, directory, `.htaccess`

**Override:** `Indexes`

**Origin:** Apache

**Example:** `ImapMenu formatted`

The `ImapMenu` directive determines the action taken if an imagemap file is called without valid coordinates.

**Parameter:** *none* | *formatted* | *semiformatted* | *unformatted*

- The *none* parameter means no menu is generated and the default action is performed
- The *formatted* parameter formatted menu which is the simplest menu. Comments in the imagemap file are ignored. A level one header is printed, then an hrule, then the links, each on a separate line. The menu has a consistent, plain look close to that of a directory listing.
- The *semiformatted* parameter generates a semiformatted menu, comments are printed where they occur in the imagemap file. Blank lines are turned into HTML breaks. No header or hrule is printed, but otherwise the menu is the same as a formatted menu.
- The *unformatted* parameter generates an unformatted menu, comments are printed, blank lines are ignored. Nothing is printed that does not appear in the imagemap file. All breaks and headers must be included as comments in the imagemap file. This gives you the most flexibility over the appearance of your menu, but requires you to treat your map files as HTML instead of plaintext.

**Additional information on Imagemap files:** The lines in the imagemap files can have one of several formats:

```
directive value [x,y ...]
directive value "Menu text" [x,y ...]
directive value x,y ... "Menu text"
```

The directive is one of `base`, `default`, `poly`, `circle`, `rect`, or `point`. The value is an absolute or relative URL, or one of the special values listed below. The coordinates are `x,y` pairs separated by whitespace. The quoted text is used as the text of the link if a imagemap list is generated. Lines beginning with `#` are comments.

## **Imagemap File Directives**

There are six directives allowed in the imagemap file. The directives can come in any order, but are processed in the order they are found in the imagemap file.

- **base directive** - Has the effect of `<BASE HREF="value">`. The non-absolute URLs of the map-file are taken relative to this value. The base directive overrides `ImapBase` as set in a `.htaccess` file or in the server configuration files. In the absence of an `ImapBase` configuration directive, base defaults to `http://server_name/`.
- **base\_uri** - Is synonymous with `base`. Note that a trailing slash on the URL is significant.
- **default directive** - The action taken if the coordinates given do not fit any of the `poly`, `circle` or `rect` directives, and there are no point directives. Defaults to `nocontent` in the absence of an `ImapDefault` configuration setting, causing a status code of 204 No Content to be returned. The client should keep the same page displayed.
- **poly directive** - Takes three to one-hundred points, and is obeyed if the user selected coordinates fall within the polygon defined by these points.
- **circle directive** - Takes the center coordinates of a circle and a point on the circle. Is obeyed if the user selected point is within the circle.
- **rect directive** - Takes the coordinates of two opposing corners of a rectangle. Obeyed if the point selected is within this rectangle.
- **point directive** - Takes a single point. The point directive closest to the user selected point is obeyed if no other directives are satisfied. Note that default will not be followed if a point directive is present and valid coordinates are given.

## Values

The values for each of the directives can be any of the following:

- **URL** - The URL can be relative or absolute. Relative URLs can contain `'..'` syntax and will be resolved relative to the base value. The base value itself will not be resolved according to the current value. The statement `base mailto:` will work properly, though.
- **Map** - Equivalent to the URL of the imagemap file itself. No coordinates are sent with this, so a list will be generated unless `ImapMenu` is set to `none`.
- **Menu** - Synonymous with `map`.
- **Referer** - Equivalent to the URL of the referring document. Defaults to `http://servername/` if no `Referer:`
- **nocontent** - Sends a status code of 204 No Content, telling the client to keep the same page displayed. Valid for all but `base`.
- **Error** - Fails with a 500 Server Error. Valid for all but `base`, but sort of useless for anything but `default`.

## Coordinates

`0,0 200,200` - A coordinate consists of an `x` and a `y` value separated by a comma. The coordinates are separated from each other by whitespace. To accommodate the way Lynx handles imagemaps, should a user select the coordinate `0,0`, it is as if no coordinate had been selected.

## Quoted Text

`list Text` - After the value or after the coordinates, the line optionally may contain text within double quotes. This string is used as the text for the link if a list is generated:

```
<A HREF="http://QIBM.com/">list text</A>
```

If quoted text is not present, the name of the link will be used as the text:

```
<A HREF="http://QIBM.com/">http://QIBM.com</A>
```

It is impossible to escape double quotes within this text.

## Example Mapfile



```
#Comments are printed in a 'formatted' or 'semiformatted' list.  
#And can contain html tags. <hr>  
base referer  
poly map "Could I have a list, please?" 0,0 0,10 10,10 10,0  
rect .. 0,0 77,27 "the directory of the referer"  
circle http://www.ibm.com/lincoln/feedback/ 195,0 305,27  
rect another_file "in same directory as referer" 306,0 419,27  
point http://www.ibm.com/ 100,100  
point http://www.ibm.com/ 200,200  
rect mailto:me@ibm.com 100,150 200,0 "Bugs?"
```

### Referencing your mapfile

```
<A HREF="/maps/imagemap1.map">  
<IMG ISMAP SRC="/images/imagemap1.gif">  
</A>
```

## Module mod\_include

Module mod\_include supports directives for the IBM HTTP Server for i Web server.

### Summary

The module mod\_include provides for server-parsed html documents.

**Note:** A configuration change is required in order for mod\_include to work correctly. Previously, mod\_include was a handler, and the config file had a AddHandler server-parsed .htmls directive in order to define mod\_include as a handler for extensions of .htmls. mod\_include is now a filter. Thus, the AddHandler directive no longer applies. Directive AddOutputFilter must be used and associated with a file extension, much like directive AddHandler. For example:

```
AddOutputFilter INCLUDES .shtml
```

### Enabling Server-Side Includes

Server-Side Includes (SSI) are implemented by the INCLUDES filter. If documents containing SSI directives are given the extension .shtml, the following directives makes the HTTP Server parse and assign the resulting documents as MIME type text/html. For example:

```
AddType text/html .shtml  
AddOutputFilter INCLUDE .shtml
```

The following directive must be given for the directories containing the shtml files (typically in a <Directory> section, but this directive is also valid .htaccess files if AllowOverride Options is set). For example:

```
Options +Includes
```

See the “Options” on page 319 directive for more information.

**Note:** The IBM i system does not support XBitHack to enable server-side includes.

### Directives

- “SSIEndTag” on page 404
- “SSIErrorMsg” on page 404
- “SSIStartTag” on page 404
- “SSITimeFormat” on page 405
- “SSIUndefinedEcho” on page 405

## SSIEndTag:

**Module:** mod\_include

**Syntax:** SSIEndTag *string*

**Default:** SSIEndTag "-->"

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Example:** SSIEndTag "-->"

The SSIEndTag directive changes the string that mod\_include looks for to mark the end of a include command.

**Parameter:** *string*

- The *string* parameter represents the string that mod\_include looks for to mark the end of a include command.

## SSIErrorMsg:

**Module:** mod\_include

**Syntax:** SSIErrorMsg *string*

**Default:** SSIErrorMsg "[an error occurred while processing this directive]"

**Context:** server config, virtual host, directory, .htaccess

**Override:** none

**Origin:** Apache

**Example:** SSIErrorMsg "This is the default error message"

This SSIErrorMsg directive defines the default error message that is used when an error is encountered while processing SSI tags in a file. This configuration directive can be used instead of the **config errmsg** SSI tag.

**Parameter:** *string*

- The string parameter defines the default error message that is used when an error is encountered while processing SSI tags in a file. For example:  
SSIErrorMsg "This is the default error message"

## SSIStartTag:

**Module:** mod\_include

**Syntax:** SSIStartTag *string*

**Default:** SSIStartTab "<!--#"

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Example:** SSIStartTab "<!--#"

The SSIEndTag directive changes the string that mod\_include looks for to mark an include element to process. You may want to use this option if you have 2 servers parsing the output of a file (each processing different commands, possibly at different times).

**Parameter:** *string*

- The *string* parameter represents the string that mod\_include looks for to mark an include element to process.

### Example 1

```
SSIStartTag "!ENTITY!*"
SSIEndTag "%>"
```

The example above, which specifies a matching SSIEndTag, allows you to use SSI directives as shown in the example below:

### Example 2: SSI directives with alternate start and end tags

```
*!ENTITY!*%printenv %>
```

### SSITimeFormat:

**Module:** mod\_include

**Syntax:** SSITimeFormat *strftime string*

**Default:** SSITimeFormat "%A, %d-%b-%Y %H:%M:%S %Z"

**Context:** server config, virtual host, directory, .htaccess

**Override:** none

**Origin:** Apache

**Example:** SSITimeFormat "%H:%M:%S %m-%d-%y"

The SSITimeFormat directive defines the default dates/times format that are returned to the browser while processing SSI tags. This configuration directive can be used instead of the config timefmt SSI tag.

**Parameter:** *strftime string*

- The *strftime string* parameter defines the default dates/times format that are returned to the browser while processing SSI tags. For example,

```
SSITimeFormat "%H:%M:%S %m-%d-%y"
```

See "Server-side include commands for HTTP Server" on page 502 for a list of supported server-side include directives.

**Note:** HTTP Server does not support the %Z (time zone) format.

### SSIUndefinedEcho:

**Module:** mod\_include

**Syntax:** SSIUndefinedEcho *string*

**Default:** SSIUndefinedEcho "(none)"

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Example:** SSIUndefinedEcho "The value on an SSI Echo request is not defined"

The SSIUndefinedEcho directive is used to define the default message that is used when an "echo" SSI tag is requesting a variable whose value has not been set.

**Parameter:** *string*

- The *string* parameter defines the default message that is used when an "echo" SSI tag is requesting a variable whose value has not been set.

### Module mod\_log\_config

Module mod\_log\_config supports directives for the IBM HTTP Server for i Web server.

### Summary

The module `mod_log_config` provides for logging of the requests made to the server, using the Common Log Format or a user-specified format. There are 3 directives that control log file creation in this module. The `TransferLog`, `LogFormat`, and `CustomLog` directives are used for log file creation. The `TransferLog` directive is used to create a log file. The `LogFormat` directive is used to set a custom format. The `CustomLog` directive to define a log file and format in one go. The `TransferLog` and `CustomLog` directives can be used multiple times in each server to cause each request to be logged to multiple files. The other directives in this module control log file archiving. See “Log formats for HTTP Server” on page 27 for information on the log file formats supported on HTTP Server.

### Use with virtual hosts

If a “<VirtualHost>” on page 332 section does not contain any `TransferLog` or `CustomLog` directives, the logs defined for the main server will be used. If it does contain one or more of these directives, requests serviced by this virtual host will only be logged in the log files defined within its definition, not in any of the main server’s log files. See the examples below.

### Security considerations

See “Security tips for HTTP Server” on page 29 for details on why your security could be compromised if the directory where log files are stored is writable by anyone other than the user that starts the server.

### Directives

- “`CustomLog`”
- “`FRCACustomLog`” on page 408
- “`LogFormat`” on page 411
- “`TransferLog`” on page 411

### CustomLog:

**Module:** `mod_log_config`

**Syntax:** `CustomLog file-or-pipe format-or-nickname [env=[!]environment-variable]`

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** The directive can be specified multiple times in the same configuration file. This is how one would generate multiple log files for the same server instance. For example, if you want an access log, agent log, and referer log, you could specify this directive three separate times with a different file and format. Log files created with `CustomLog` will be created with a CCSID of UTF-8 (1208) for Integrated File System.

**Example:** See below.

The `CustomLog` directive is used to log requests to the server. A log format is specified, and the logging can optionally be made conditional on request characteristics using environment variables.

#### Parameter One: *file-or-pipe*

- The *file-or-pipe* value indicates the filename to which log records should be written. This is used exactly like the argument to `TransferLog`; that is, it is either a full path or relative to the current server root. If a pipe is specified, it would be the name of a program that would receive the log file information on standard in. A pipe is specified by using the pipe character (`|`) followed by a path to the program name (no space between them). The program name can be either a path to a QSYS program object or an IFS path to a symbolic link. The symbolic link would then link to a QSYS program. Data written to the pipe from the server will be in the FSCCSID that is in use by the server.

#### Parameter Two: *format-or-nickname*

- If the value is *format*, it specifies a format for each line of the log file. The options available for the format are exactly the same as for the argument of the LogFormat directive. If the format includes any spaces (which it will in almost all cases) they should be enclosed in double quotes. If the argument is nickname, that nickname will tie back to a LogFormat directive with the same specified nickname.

If the nickname "DDS" is specified, the server will create a DDS log file and each record will contain the format described by file QHTTPSVR/QAZHBLOG. When the second argument is "DDS", a path name to a file in the QSYS.LIB file system must also be specified. When "DDS" is specified, it is not necessary to use the Logformat directive to define the format. The nickname "DDS" is a special nickname that is predefined in HTTP Server.

**Parameter Three:** *[env=[!]*environment-variable*]*

- The optional *env=* clause controls whether a particular request will be logged in the specified file or not. If the specified environment variable is set for the request (or is not set, in the case of a 'env=!name' clause), then the request will be logged. Environment variables can be set on a per-request basis using the mod\_setenvif and/or mod\_rewrite modules.

There is no way to specify conditional logging for requests handled by Fast Response Cache Accelerator (FRCA). That is, environment variable conditions have no effect on the selection of FRCA requests that are logged. If FRCA is being used and a FRCACustomLog is not configured, all requests handled by FRCA will be logged in the CustomLog. The environment variable conditions continue to apply to requests not served from FRCA.

For example, if you want to record requests for all GIF images on your server in a separate log file, but not your main log, you can use:

```
SetEnvIf Request_URI \.gif$ gif-image
CustomLog gif-requests.log common env=gif-image
CustomLog nongif-requests.log common env=!gif-image
```

Examples of CustomLog:

```
# CustomLog with format nickname
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log common

# CustomLog in QSYS with format nickname
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog /QSYS.LIB/MYLIB.LIB/MYLOG.FILE common

# CustomLog with explicit format string
CustomLog logs/access_log "%h %l %u %t \"%r\" %>s %b"

# CustomLog with env specified
SetEnvIf Request_URI \.gif$ gif-image
CustomLog gif-requests.log common env=gif-image
CustomLog nongif-requests.log common env=!gif-image

# CustomLog defining a piped log
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog |/QSYS.LIB/MYLIB.LIB/CUSTOMPIPE.PGM common
```

For IFS files, the user must create the directories that contain the log file and must grant the QTMHHTTP user write access to the directory. For QSYS.LIB logs, the user must create the library that contains the logs. The server will create the file and members in the specified library.

**Note:** It is recommended that HTTP Server create the QSYS.LIB log file. If the QSYS.LIB log file is created with a record length that is too small, log information may be truncated and lost. By default the server creates all QSYS.LIB log files with a record size of 512 or greater.

If the filename does not begin with a slash (/) then it is assumed to be relative to the ServerRoot. If "LogCycle" on page 313 is active and if the path ends without a (/) character, then the path is considered

to be the complete log file name. In this case, the server will add an extension in the format QCYMMDDHH, where these variables have the following values:

- **Q** is a default value that indicates to the server that this is a log file.
- **C** is the century indicator (0 for pre-2000, 1 for post-2000).
- **YY** is the year indicator.
- **MM** is the month indicator.
- **DD** is the day indicator.
- **HH** is the hour indicator (00 = 00:00 (midnight), 23=23:00).

**Note:** This variable will not be generated for filesystem QDLS

For example, a path of "/logs/errorlog" results in a file such as "/logs/errorlog.Q100030300".

If "LogCycle" on page 313 is active and if the path ends with a (/) character, then the path is considered to be the directory that will contain the log file. In this case, the server will create log files named in the QCYMMDDHH format. For example, a path of "/logs/errorlog/" results in a file such as "/logs/errorlog/Q100030300". If "LogCycle" on page 313 is active and the logfile is in the QSYS filesystem, the name must end in the file component of the IFS path. Example:

```
# Config file directives
LogCycle Daily
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog /QSYS.LIB/MYLIB.LIB/MYLOGS.FILE common
```

The resulting daily log rollovers will be of the form /QSYS.LIB/MYLIB.LIB/MYLOGS.FILE/Qcyymmddhh.MBR.

"LogCycle" on page 313 Hourly is not valid if the logfile is in the QDLS filesystem as that filesystem only supports 8 character file names and 3 character extensions. For QDLS, the path given on the CustomLog directive must be a directory. For example

```
CustomLog /QDLS/MYPATH/LOGS/ common
```

If "LogCycle" on page 313 is not active, no special naming is used. The name of the log file given on the CustomLog directive is used as given for the name of the log file. If the name is a directory, a default name of http.log will be concatenated to the directory name to create the log file. For example:

```
# Config file directives
LogCycle Off
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog /logs/path/ common
```

The resulting log file will be /logs/path/http.log.

**Security:** See "Security tips for HTTP Server" on page 29 for details on why your security could be compromised if the directory where log files are stored is writable by anyone other than the user that starts the server. If a program is used, then it will be run under the user who started httpd. This will be root if the server was started by root (be sure that the program is secure).

### **FRCACustomLog:**

**Module:** mod\_log\_config

**Syntax:** FRCACustomLog *file-or-pipe file format-or-nickname*

**Default:** none

**Context:** server config

**Override:** none

**Origin:** IBM

**Usage Considerations:** The directive can be specified multiple times in the same configuration file. This is how one would generate multiple log files for the same server instance. For example, if you want an access log, agent log, and referer log, you could specify this directive three separate times with a different file and format. Log files created with `FRCACustomLog` will be created with a CCSID of UTF-8 (1208) for Integrated File System.

**Example:** See below.

The `FRCACustomLog` directive is used to log FRCA requests to the server.

**Parameter One:** *file-or-pipe file*

- The *file-or-pipe* file value indicates the filename to which log records should be written. It is either a full path or relative to the current server root. If a pipe is specified, it would be the name of a program that would receive the log file information on standard in. A pipe is specified by using the pipe character "|" followed by a path to the program name (no space between them). The program name can be either a path to a QSYS program object or an IFS path to a symbolic link. The symbolic link would then link to a QSYS program. Note that data written to the pipe from the server will be in the FSCCSID that is in use by the server.

**Parameter Two:** *format-or-nickname*

- The *format-or-nickname* argument specifies a format or nickname for each line of the log file. If it is a format, it specifies a format for each line of the log file. The options available for the format are exactly the same as for the argument of the `LogFormat` directive. If the format includes any spaces (which it will in almost all cases) they should be enclosed in double quotes. If the argument is a nickname, that nickname will tie back to a `LogFormat` directive with the same specified nickname.

If the nickname "DDS" is specified, the server will create a DDS log file and each record will contain the format described by file `QHTTPSVR/QAZHBLOG`. When the second argument is "DDS" a path name to a file in the `QSYS.LIB` file system must also be specified. When "DDS" is specified, it is not necessary to use the `Logformat` directive to define the format. The nickname "DDS" is a special nickname that is pre-defined in the server. See directive `Logformat` for additional considerations for the DDS nickname.

Examples of `FRCACustomLog`:

```
# FRCACustomLog with format nickname
LogFormat "%h %l %u %t \"%r\" %>s %b" common
FRCACustomLog logs/FRCAaccess_log common

# FRCACustomLog in QSYS with format nickname
LogFormat "%h %l %u %t \"%r\" %>s %b" common
FRCACustomLog /QSYS.LIB/MYLIB.LIB/MYFRCALOG.FILE common

# CustomLog in QSYS with DDS format
FRCACustomLog /QSYS.LIB/MYLIB.LIB/FRCADDSLOG.FILE DDS

# FRCACustomLog with explicit format string
FRCACustomLog logs/FRCAaccess_log "%h %l %u %t \"%r\" %>s %b"

# FRCACustomLog defining a piped log
LogFormat "%h %l %u %t \"%r\" %>s %b" common
FRCACustomLog |/QSYS.LIB/MYLIB.LIB/PIPELOG.PGM common
```

For IFS log files and QSYS log files, the user must create the directories that contain the log file and must grant the `QTMHHTTP` user write access to the directory. For `QSYS.LIB` logs, the user must create the library that contains the logs. The server will create the file and members in the specified library.

**Note:** It is recommended that HTTP Server create the `QSYS.LIB` log file. If the `QSYS.LIB` log file is created with a record length that is too small, log information may be truncated and lost. By default the server creates all `QSYS.LIB` log files with a record size of 512 or greater.

If the filename does not begin with a slash (/) then it is assumed to be relative to the ServerRoot. If "LogCycle" on page 313 is active and if the path ends without a (/) character, then the path is considered to be the complete log file name. In this case, the server will add an extension in the format QCYMMDDHH, where these variables have the following values:

- **Q** is a default value that indicates to the server that this is a log file.
- **C** is the century indicator (0 for pre-2000, 1 for post-2000).
- **YY** is the year indicator.
- **MM** is the month indicator.
- **DD** is the day indicator.
- **HH** is the hour indicator (00 = 00:00 (midnight), 23=23:00).

**Note:** this variable will not be generated for filesystem QDLS

For example, a path of "/logs/errorlog" results in a file such as "/logs/errorlog.Q100030300".

If "LogCycle" on page 313 is active and if the path ends with a (/) character, then the path is considered to be the directory that will contain the log file. In this case, the server will create log files named in the QCYMMDDHH format. For example, a path of "/logs/errorlog/" results in a file such as "/logs/errorlog/Q100030300". If "LogCycle" on page 313 is active and the logfile is in the QSYS filesystem, the name must end in the file component of the IFS path. Example:

```
# Config file directives
LogCycle Daily
LogFormat "%h %l %u %t \"%r\" %>s %b" common
FRCACustomLog /QSYS.LIB/MYLIB.LIB/MYLOGS.FILE common
```

The resulting daily log rollovers will be of the form /QSYS.LIB/MYLIB.LIB/MYLOGS.FILE/Qcyymmddhh.MBR.

"LogCycle" on page 313 Hourly is not valid if the logfile is in the QDLS filesystem as that filesystem only supports 8 character file names and 3 character extensions. Also for QDLS, the path given on the FRCACustomLog directive must be a directory. For example:

```
FRCACustomLog /QDLS/MYPATH/LOGS/ common
```

The resulting log files would be /QDLS/MYPATH/LOGS/Qcyymmdd.

If "LogCycle" on page 313 is not active, no special naming is used. The name of the log file given on the FRCACustomLog directive is used as given for the name of the log file. If the name is a directory, a default name of http.log will be concatenated to the directory name to create the log file. For example:

```
# Config file directives
LogCycle Off
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog /logs/path/ common
```

The resulting log file will be /logs/path/http.log.

If FRCACustomLog is in the configuration, FRCA requests will be logged to the file specified on the FRCACustomLog directive. All non-FRCA related requests will be logged to any other custom logs configured with the CustomLog directive. Example:

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log common
FRCACustomLog logs/FRCAaccess_log common
```

All FRCA requests will be logged to logs/FRCAaccess\_log and all non-FRCA requests will be logged to logs/access\_log. If FRCACustomLog is not specified in the configuration of the server instance, ALL requests are logged to any custom logs configured with CustomLog including FRCA requests.



## LogFormat:

**Module:** mod\_log\_config

**Syntax:** LogFormat *format* [*nickname*]

**Default:** LogFormat "%h %l %u %t \"%r\" %s %b"

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** The directive can be specified multiple times in the same configuration file. This is how one would generate multiple log file formats. For example, if you want an access log, agent log, and referer log, you could specify this directive three separate times to define the formats of your log files.

**Example:** LogFormat "%h %l %u %t \"%r\" %s %b \"%{Referer}i\" \"%{User-agent}i\""

The LogFormat directive sets the format of the default log file named by the TransferLog directive. See the section on Custom Log Formats for details on the format arguments. If you include a nickname for the format on the directive line, you can use that nickname in FRCACustomLog and CustomLog directives rather than repeating the entire format string.

### Parameter One: *format*

- The *format* parameter sets the format of the default log file named by the TransferLog directive. See the section on Custom Log Formats for details on the format arguments.

### Parameter Two: [*nickname*]

- The optional *nickname* parameter allows you to include a nickname for the format on the directive line.

A LogFormat directive that defines a nickname does nothing else. That is, it only defines the nickname, it doesn't actually apply the format and make it the default.

If LogFormat is used without a nickname, then any TransferLog directive that does not specify a format will use the format defined with this directive, if it happened to be the most recent LogFormat directive in the configuration file. If another LogFormat directive (without a nickname) is placed in the configuration file, then that format becomes the new log format to be used on subsequent TransferLog directives.

The nickname "DDS" is a log format reserved for use in configuring data description specification (DDS) log files. The server will automatically recognize this format and create a DDS log file based on QHTTSPVR/QAZHBLOG. The "DDS" nickname should not be used when defining a new LogFormat. A LogFormat directive with a nickname of "DDS" will be ignored by the server. The server will assume a DDS file in QSYS.LIB when the "DDS" nickname appears on the CustomLog or FRCACustomLog directives.

See "Log formats for HTTP Server" on page 27 for information on the log file formats supported on HTTP Server.

## TransferLog:

**Module:** mod\_log\_config

**Syntax:** TransferLog *file-or-pipe*

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** The directive can be specified multiple times in the same configuration file. This is how one would generate multiple log files for the same server instance. For example, if you want an access log, agent log, and referer log, you could specify this directive three separate times with a different file and most recent LogFormat. Log files created with TransferLog will be created with a CCSID of UTF-8 (1208) for Integrated File System.

**Example:** TransferLog logs/access\_log

The TransferLog directive adds a log file in the format defined by the most recent LogFormat directive, or Common Log Format. This is only if no other default format has been specified.

**Parameter:** *file-or-pipe*

- The *file-or-pipe* parameter specifies either a filename relative to the ServerRoot or a program to pipe to. Use the pipe symbol (|) followed by a program to receive the log information in its standard input. Data written to the pipe from the server will be in UTF-8 (1208) in use by the server. The new program will not be started for a VirtualHost if it inherits the TransferLog from the main server.

Examples of TransferLog:

```
# IFS example
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\""
TransferLog logs/access_log

# QSYS example
LogFormat "%h %l %u %t \"%r\" %>s %b"
TransferLog /QSYS.LIB/MYLIB.LIB/MYLOGS.FILE

# Piped log example
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\""
TransferLog |/QSYS.LIB/MYLIB.LIB/TRANSPIPE.PGM
```

For IFS files, the user must create the directories that contain the log file and must grant the QTMHHTTP user write access to the directory. For QSYS.LIB logs, the user must create the library that contains the logs. The server will create the file and members in the specified library. If the filename does not begin with a slash (/) then it is assumed to be relative to the ServerRoot. If LogCycle is active and if the path ends without a (/) character, then the path is considered to be the complete log file name. In this case, the server will add an extension in the format QCYMMDDHH, where these variables have the following values:

- **Q** is a default value that indicates to the server that this is a log file.
- **C** is the century indicator (0 for pre-2000, 1 for post-2000).
- **YY** is the year indicator.
- **MM** is the month indicator.
- **DD** is the day indicator.
- **HH** is the hour indicator (00 = 00:00 (midnight), 23=23:00).

**Note:** this variable will not be generated for filesystem QDLS

For example, a path of "/logs/errorlog" results in a file such as "/logs/errorlog.Q100030300".

If "LogCycle" on page 313 is active and if the path ends with a (/) character, then the path is considered to be the directory that will contain the log file. In this case, the server will create log files named in the QCYMMDDHH format. For example, a path of "/logs/errorlog/" results in a file such as "/logs/errorlog/Q100030300". If "LogCycle" on page 313 is active and the logfile is in the QSYS filesystem, the name must end in the file component of the IFS path. For example:

```
# Config file directives
LogCycle Daily
LogFormat "%h %l %u %t \"%r\" %>s %b" common
TransferLog /QSYS.LIB/MYLIB.LIB/MYLOGS.FILE
```

The resulting daily log rollovers will be of the form /QSYS.LIB/MYLIB.LIB/MYLOGS.FILE/Qcymmddhh.MBR.

“LogCycle” on page 313 Hourly is not valid if the logfile is in the QDLS filesystem as that filesystem only supports 8 character file names and 3 character extensions. If “LogCycle” on page 313 is not active, no special naming is used. The name of the log file given on the TransferLog directive is used as given for the name of the log file. If the name is a directory, a default name of http.log will be concatenated to the directory name to create the log file. For example:

```
# Config file directives
LogCycle Off
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog /logs/path/ common
```

The resulting log file will be /logs/path/http.log.

**Note:** See “Security tips for HTTP Server” on page 29 for details on why your security could be compromised if the directory where log files are stored is writable by anyone other than the user that starts the server. If a program is used, then it will be run under the user who started httpd. This will be root if the server was started by root (be sure that the program is secure).

**Note:** When possible, you should use “CustomLog” on page 406 in place of TransferLog.

## Module mod\_log\_io

Module mod\_log\_io supports logging formats for the IBM HTTP Server for i Web server.

### Summary

This module provides the logging of input and output number of bytes received and sent per request. The numbers reflect the actual bytes received on the network, which then takes into account the headers and bodies of requests and responses. The counting is done before SSL/TLS on input and after SSL/TLS on output. The numbers will correctly reflect any changes made by encryption.

This module requires Module mod\_log\_config, and is loaded by default. No LoadModule statement is required.

This module adds two new logging formats. The characteristics of the request itself are logged by placing “%” directives in the format string, which are replaced in the log file by the values as follows:

Format String Description %...I

Bytes received, including request and headers, cannot be zero (%...0). Bytes sent, including headers, cannot be zero.

### Format String Description

String	Description
%...I	Bytes received, including request and headers, cannot be zero.
%...O	Bytes sent, including headers, cannot be zero.

### Example: Combined I/O log format

```
"%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\" %I %O"
```

## Module mod\_mime

Module mod\_mime supports directives for the IBM HTTP Server for i Web server.

### Summary

The module `mod_mime` associates the request filename's extensions (for example, `.html`) with the file's behavior (handlers and filters) and content (mime-type, language, character set and encoding). This module is used to determine various bits of "meta information" with files by their filename extensions. This information relates to the content of the document to its mime-type, language, character set and encoding. This information is sent to the browser, and participates in content negotiation. The user's preferences are respected when choosing one of several possible files to serve. In addition, a handler can be set for a document that determines how the document will be processed within the server. See "Module `mod_negotiation`" on page 425 for more information regarding content negotiation.

The directives `AddCharset`, `AddClient`, `AddEncoding`, `AddHandler`, `AddLanguage`, and `AddType` are all used to map file extensions onto the meta-information for that file. Respectively they set the character set, content-encoding, handler, content-language, browser, and MIME-type (content-type) of documents.

In addition, `mod_mime` may define the document handler that controls which module or script will serve the document. With the introduction of filters, `mod_mime` can also define the filters that the content should be processed through (for example, the `Includes` output filter for server side scripting) and what filters the client request and POST content should be processed through (the input filters).

The directives `AddHandler`, `AddOutputFilter`, and `AddInputFilter` control the modules or scripts that serve the document. The `MultiviewsMatch` directive allows `mod_negotiation` to consider these file extensions when testing `Multiviews` matches.

The directive `TypesConfig` is used to specify a file that also maps extensions onto MIME types. Most administrators use the provided `mime.types` file that associates common filename extensions with IANA registered content types.

The core directives `ForceType` and `SetHandler` are used to associate all the files in a given container (`<location>`, `<directory>`, or `<files>`) with a particular MIME-type or handler. These settings override any filename extension mappings defined in `mod_mime`.

Note that changing the type or encoding of a file does not change the value of the Last-Modified header. Therefore, previously cached copies may still be used by a client or proxy, with the previous headers. If you change the meta-information (language, content type, character set or encoding) you may need to update affected files (updating their last modified date) to ensure that all visitors are receiving the corrected content headers.

### Files with Multiple Extensions

Files can have more than one extension, and the order of the extensions is normally irrelevant. For example, if the file `welcome.html.fr` maps onto content type `text/html` and then language French, the file `welcome.fr.html` will map onto exactly the same information. The only exception to this is if an extension is given which HTTP Server does not handle. In this case it will forget about any information it obtained from extensions to the left of the unknown extension. For example, if the extensions `fr` and `html` are mapped to the appropriate language and type, but extension `xxx` is not assigned to anything, then the file `welcome.fr.xxx.html` will be associated with content-type `text/html` but no language.

If more than one extension is given that maps onto the same type of meta-information, then the one to the right will be used. For example, if `".gif"` maps to the MIME-type `image/gif` and `".html"` maps to the MIME-type `text/html`, then the file `welcome.gif.html` will be associated with the MIME-type `"text/html"`.

When a file with multiple extensions gets associated with both a MIME-type and a handler be careful. This will usually result in the module associating a request with the handler. For example, if the `.imap` extension is mapped to the handler `"imap-file"` (from `mod_imap`) and the `.html` extension is mapped to the MIME-type `"text/html"`, then the file `world.imap.html` will be associated with both the `"imap-file"` handler and `"text/html"` MIME-type. When it is processed, the `imap-file` handler will be used, and it will be treated as a `mod_imap` `imagemap` file.

## Directives

- “AddCharset”
- “AddClient” on page 416
- “AddEncoding” on page 416
- “AddHandler” on page 417
- “AddInputFilter” on page 417
- “AddLanguage” on page 418
- “AddOutputFilter” on page 418
- “AddType” on page 419
- “DefaultLanguage” on page 419
- “ModMimeUsePathInfo” on page 420
- “MultiviewsMatch” on page 420
- “RemoveCharset” on page 421
- “RemoveClient” on page 421
- “RemoveEncoding” on page 422
- “RemoveHandler” on page 422
- “RemoveInputFilter” on page 423
- “RemoveLanguage” on page 423
- “RemoveOutputFilter” on page 423
- “RemoveType” on page 424
- “SuffixCaseSense” on page 424
- “TypesConfig” on page 425

### AddCharset:

**Module:** mod\_mime

**Syntax:** AddCharset *charset extension [extension...]*

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** FileInfo

**Origin:** Apache

**Example:** AddCharset ISO-2022-JP .jis

The AddCharset directive maps the given filename extensions to the specified content charset. Charset is the MIME charset parameter of filenames containing extension. This mapping is added to any already in force, overriding any mappings that already exist for the same extension.

This directive is useful for informing the client about the character encoding of the document so it can be interpreted and displayed appropriately. It also used for content negotiation. Content Negotiation is where the server returns one from several documents based on the client's charset preference.

#### **Parameter One:** *charset*

- The *charset* parameter value is any valid MIME character set.

#### **Parameter Two:** *extension*

- The *extension* parameter value is any character string that is a valid file extension.

See “Module mod\_negotiation” on page 425 for more information.

## AddClient:

**Module:** mod\_mime

**Syntax:** AddClient *user-agent extension*

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** none

**Origin:** IBM

**Example:** AddClient Mozilla/2.0 .moz

**Example:** AddClient IBM\* .ibm

The AddClient directive binds files with a particular extension to the type and version of the browser (user-agent) that is sending the request. This is often referred to as Automatic Browser Detection. All HTTP requests contain a User-Agent header that identifies the client browser. Based on this User-Agent header, the server can respond with a specific version of the resource (with the extension specified) that is especially appropriate for the client browser.

**Parameter One:** *user-agent*

- The *user-agent* parameter value matched in the User-Agent header of the incoming request. This is case-sensitive. The asterisk may be used as a wildcard character.

**Parameter Two:** *extension*

- The *extension* parameter value is the file extension that should be associated with the browser. Wildcards cannot be used.

## AddEncoding:

**Module:** mod\_mime

**Syntax:** AddEncoding *MIME-enc extension [extension...]*

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** FileInfo

**Origin:** Apache

**Example:** AddEncoding x-gzip gz

The AddEncoding directive maps the given filename extensions to the specified encoding type. MIME-enc is the MIME encoding that is used for documents containing the extension. This mapping is added to any already in force, overriding any mappings that already exist for the same extension.

Old clients expect x-gzip and x-compress, however the standard dictates that they're equivalent to gzip and compress respectively. HTTP Server does content encoding comparisons by ignoring any leading x-. When responding with an encoding the HTTP Server will use whatever form (for example., x-QIBM or QIBM) the client requested. If the client didn't specifically request a particular form, the server will use the form given by the AddEncoding directive. In conclusion you should always use x-gzip and x-compress for these two specific encodings. More recent encodings, such as deflate should be specified without the x-.

**Parameter One:** *MIME-enc*

- The *MIME-enc* parameter value should be set to a content-encoding supported by HTTP/1.1. Currently, these values are 'gzip', 'compress' and 'deflate'.

**Parameter Two:** *extension*

- The *extension* parameter value is any string that is a valid file extension.

## AddHandler:

**Module:** mod\_mime

**Syntax:** AddHandler *handler-name extension [extension...]*

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** FileInfo

**Origin:** Apache

**Example:** AddHandler cgi-script cgi

The AddHandler directive maps the filename extensions to handler *handler-name*. This mapping is added to any already in force, overriding any mappings that already exist for the same extension. For example, to activate CGI scripts with the file extension ".cgi", you might use:

```
AddHandler cgi-script cgi
```

Once this has been put into your configuration file, any file containing the ".cgi" extension will be treated as a CGI program.

**Parameter One:** *handler-name*

- The *handler-name* parameter value is the name of the handler (program) that will process the request.

**Parameter Two:** *extension*

- The *extension* parameter value is any character string that is a valid file extension.

AddHandler can also be used to configure the use of Server Side Includes. This is done with the following directive combination:

```
AddType text/html .shtml
AddHandler server-parsed .shtml
```

See "Handler for HTTP Server" on page 200 for more information.

## AddInputFilter:

**Module:** mod\_mime

**Syntax:** AddInputFilter *filter extension [extension ...]*

**Default:** none

**Context:** directory, .htaccess

**Override:** none

**Origin:** Apache

**Example:** AddInputFilter gzip .zip

The AddInputFilter directive maps the filename extensions *extension* to the filters that will process client requests and POST input (when they are received by the server). This is in addition to any filters defined elsewhere, including the SetInputFilter directive. This mapping is merged over any already in force, overriding any mappings that already exist for the same extension.

If SuffixCaseSense is on (default is off), then the extension argument is case-insensitive. The extension can be specified with or without a leading dot.

**Parameter One:** *filter*


- The *filter* parameter value is the process that is applied to data that is sent or received by the server.

**Parameter Two:** *extension*

- The *extension* parameter value is any character string that is a valid file extension.

### Example

```
<Directory/www/data/>
AddInputFilter gzip Zip
</Directory>
```

See the Apache Software Foundation filter documentation  for more information.

### AddLanguage:

**Module:** mod\_mime

**Syntax:** AddLanguage *MIME-lang extension [extension...]*

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** FileInfo

**Origin:** Apache

**Example:** AddLanguage fr .fr

The AddLanguage directive maps the given filename extensions to the specified content language. MIME-lang is the MIME language of filenames containing extension. This mapping is added to any already in force, overriding any mappings that already exist for the same extension.

Even though the content language is reported to the client, the browser is unlikely to use this information. The AddLanguage directive is more useful for content negotiation, where the server returns one from several documents based on the client's language preference.

If multiple language assignments are made for the same extension, the last one encountered is the one that is used.

#### Parameter One: *MIME-lang*

- The *MIME-lang* parameter value is any valid MIME-language designation.

#### Parameter Two: *value*

- The *extension* parameter value is any character string that is a valid file extension.

See "Module mod\_negotiation" on page 425 for more information.

### AddOutputFilter:

**Module:** mod\_mime

**Syntax:** AddOutputFilter *filter extension [extension ...]*

**Default:** none

**Context:** directory, .htaccess

**Override:** none

**Origin:** Apache

**Example:** AddOutputFilter INCLUDES shtml

The AddOutputFilter directive maps the filename extensions extension to the filters that process responses from the server (before they are sent to the client). This is in addition to any filters defined elsewhere, including the SetOutputFilter directive. This mapping is merged over any already in force, overriding any mappings that already exist for the same extension.

For example, the following configuration will process all .shtml files for server-side includes.

```
AddOutputFilter INCLUDES shtml
```




If `SuffixCaseSense` is on (default is off), then the extension argument is case-insensitive. The extension can be specified with or without a leading dot.

**Parameter One:** *filter*

- The *filter* parameter value is the process that is applied to data that is sent or received by the server.

**Parameter Two:** *extension*

- The *extension* parameter value is any character string that is a valid file extension.

See the Apache Software Foundation filter documentation  for more information.

### AddType:

**Module:** mod\_mime

**Syntax:** AddType *MIME-type extension [extension...]*

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** FileInfo

**Origin:** Apache

**Example:** AddType image/gif GIF

The AddType directive maps the given filename extensions onto the specified content type. MIME-type is the MIME type to use for filenames containing extension. This mapping is added to any already in force, overriding any mappings that already exist for the same extension. This directive can be used to add mappings not listed in the MIME types file. It is recommended that new MIME types be added using the AddType directive rather than changing the TypesConfig file.

**Parameter One:** *MIME-type*

- The *MIME-type* parameter value is any valid MIME-type.

**Parameter Two:** *extension*

- The *extension* parameter value is any character string that is a valid file extension.

### DefaultLanguage:

**Module:** mod\_mime

**Syntax:** DefaultLanguage *MIME-lang*

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** FileInfo

**Origin:** Apache

**Example:** DefaultLanguage en-US

The DefaultLanguage directive tells HTTP Server that all files in the directive's scope (for example, all files covered by the current <Directory> container) that don't have an explicit language extension configured by AddLanguage should be considered to be in the specified MIME-lang language. This allows entire directories to be marked as containing Dutch content, for instance, without having to rename each file. Note that unlike using extensions to specify languages, DefaultLanguage can only specify a single language.

If no DefaultLanguage directive is in force, and a file does not have any language extensions configured by AddLanguage, then that file will be considered to have no language attribute.

**Parameter:** *MIME-lang*

- The *MIME-lang* parameter value is any valid MIME-language designation.

See “Module mod\_negotiation” on page 425 for more information.

### **ModMimeUsePathInfo:**

**Module:** mod\_mime

**Syntax:** ModMimeUsePathInfo *on* | *off*

**Default:** ModMimeUsePathInfo *off*

**Context:** directory

**Override:** none

**Origin:** Apache

**Example:** ModMimeUsePathInfo *on*

The ModMimeUsePathInfo directive is used to combine the filename with the path\_info URL component to apply mod\_mime's directives to the request. The default value is *off*, meaning the path\_info component is ignored. This directive is recommended when you have a virtual filesystem.

For example, if ModMimeUsePathInfo is set to *on*, then a request for /bar/file.shtml where /bar is a Location, mod\_mime will treat the incoming request as /bar/file.shtml and directives like AddOutputFilter INCLUDES .shtml will add the INCLUDES filter to the request. If ModMimeUsePathInfo is not set, the INCLUDES filter will not be added.

**Parameter:** *on* | *off*

- The *on* parameter value specifies that filenames will be combined with path\_info URL components.
- The *off* parameter value specifies that the path\_info component is ignored.

#### **Example**

```
ModMimeUsePathInfo on
```

If you have a request for /myfile/more.shtml where myfile is an existing file containing SSI, and AcceptPathInfo is set on in order to accept the actual file "myfile" as the requested file, and ModMimeUsePathInfo is *on*, mod\_mime will treat the incoming request as SSI and directives like AddOutputFilter INCLUDES .shtml will add the INCLUDES filter to the request. If ModMimeUsePathInfo is not set, the INCLUDES filter will not be added. When ModMimeUsePathInfo is set, the trailing path name can be used to determine the content type of the existing file.

### **MultiviewsMatch:**

**Module:** mod\_mime

**Syntax:** MultiviewsMatch *NegotiatedOnly* | *Handlers* | *Filters* | *Any*

**Default:** MultiviewsMatch *NegotiatedOnly*

**Context:** server config, virtual host, directory, .htaccess

**Override:** FileInfo

**Origin:** Apache

**Example:** MultiviewsMatch *Handlers*

**Example:** MultiviewsMatch *Handlers Filters*

The MultiviewsMatch directive permits three different behaviors for mod\_negotiation's Multiviews feature. Multiviews allows a request for a file (index.html for example) to match any negotiated extensions following the base request (for example, index.html.en, index.html.fr, or index.html.gz).

**Parameter:** *NegotiatedOnly* | *Handlers* | *Filters* | *Any*

- The *NegotiatedOnly* parameter value specifies that every extension following the base name must correlate to a recognized mod\_mime extension for content negotiation (for

example, Charset, Content-Type, Language, or Encoding). This is the strictest implementation with the fewest unexpected side effects, and is the default behavior.

- The *Handlers* and *Filters* parameter value set the `MultiviewsMatch` directive to either `Handlers`, `Filters`, or both option keywords. If all other factors are equal, the smallest file will be served (for example, in deciding between `index.html.cgi` of 500 characters and `index.html.pl` of 1000 bytes, the `.cgi` file would be served). Users of `.asis` files might prefer to use the `Handler` option, if `.asis` files are associated with the `asis-handler`.
- The *Any* parameter value specifies that any extensions to match, even if `mod_mime` doesn't recognize the extension. This was the behavior in Apache 1.3, and can cause unpredictable results, such as serving `.old` or `.bak` files the webmaster never expected to be served.

### RemoveCharset:

**Module:** `mod_mime`

**Syntax:** `RemoveCharset extension [extension...]`

**Default:** none

**Context:** directory, `.htaccess`

**Override:** none

**Origin:** Apache

**Example:** `RemoveCharset .ext`

The `RemoveCharset` directive removes any character set associations for files with the given extensions. This allows `.htaccess` files in subdirectories to undo any associations inherited from parent directories or the server configuration files.

**Parameter:** *extension*

- The *extension* parameter value is any character string that is a valid file extension.

**Note:** If `SuffixCaseSense` is on (default is off), then the extension argument is case-insensitive. The extension can be specified with or without a leading dot.

### RemoveClient:

**Module:** `mod_mime`

**Syntax:** `RemoveClient extension [extension...]`

**Default:** none

**Context:** directory, `.htaccess`

**Override:** none

**Origin:** IBM

**Example:** `RemoveClient .moz`

The `RemoveClient` directive removes any client (browser) associations for files with the given extensions. This allows `.htaccess` files in subdirectories to undo any associations inherited from parent directories or the server config files.

**Parameter:** *extension*

- The *extension* parameter value is any character string that is a valid file extension.

### Example

```
/work/.htaccess:  
RemoveClient .moz
```

If `SuffixCaseSense` is on (default is off), then the extension argument is case-insensitive. The extension can be specified with or without a leading dot. This removes any special handling of `.moz` files in the `/work/`

directory (and any subdirectories), thereby disabling automatic browser detection for files in this directory. The extension argument is case-insensitive, and can be specified with or without a leading dot.

**Note:** RemoveClient directives are processed after any “AddClient” on page 416 directives, so it is possible they may undo the effects of the latter if both occur within the same directory configuration.

### RemoveEncoding:

**Module:** mod\_mime

**Syntax:** RemoveEncoding *extension* [*extension...*]

**Default:** none

**Context:** directory, .htaccess

**Override:** none

**Origin:** Apache

**Example:** RemoveEncoding .gz

The RemoveEncoding directive removes any encoding associations for files with the given extensions. This allows .htaccess files in subdirectories to undo any associations inherited from parent directories or the server config files.

**Parameter:** *extension*

- The *extension* parameter value is any character string that is a valid file extension.

### Example

```
/work/.htaccess:  
AddEncoding x-gzip .gz  
AddType text/plain .asc  
<Files *.gz.asc>  
    RemoveEncoding .gz  
</Files>
```

The example will cause work.gz to be marked as encoded with the gzip method, but cause work.gz.asc to be marked as an unencoded plaintext file.

**Note:** RemoveEncoding directives are processed after any AddEncoding directives, so it is possible they may undo the effects of the latter if both occur within the same directory configuration. If SuffixCaseSense is on (default is off), then the extension argument is case-insensitive. The extension can be specified with or without a leading dot.

### RemoveHandler:

**Module:** mod\_mime

**Syntax:** RemoveHandler *extension* [*extension...*]

**Default:** none

**Context:** directory, .htaccess

**Override:** none

**Origin:** Apache

**Usage Considerations:** RemoveHandler .html

**Example:** example

The RemoveHandler directive removes any handler associations for files with the given extensions. This allows .htaccess files in subdirectories to undo any associations inherited from parent directories or the server config files.

**Parameter:** *extension*

- The *extension* parameter value is any character string that is a valid file extension.

### Example

```
/QIBM/.htaccess: AddHandler server-parsed .html  
/QIBM/bar/.htaccess: RemoveHandler .html
```

The example has the effect of returning .html files in the /QIBM/bar directory to being treated as normal files, rather than as candidates for parsing.

### RemoveInputFilter:

**Module:** mod\_mime

**Syntax:** RemoveInputFilter *extension* [*extension* ...]

**Default:** none

**Context:** directory, .htaccess

**Override:** none

**Origin:** Apache

**Example:** RemoveInputFilter .ext

The RemoveInputFilter directive removes any input filter associations for files with the given extensions. This allows .htaccess files in subdirectories to undo any associations inherited from parent directories or the server configuration files.

**Parameter:** *extension*

- The *extension* parameter value is any character string that is a valid file extension.

**Note:** If SuffixCaseSense is on (default is off), then the extension argument is case-insensitive. The extension can be specified with or without a leading dot.

### RemoveLanguage:

**Module:** mod\_mime

**Syntax:** RemoveLanguage *extension* [*extension* ...]

**Default:** none

**Context:** directory, .htaccess

**Override:** none

**Origin:** Apache

**Example:** RemoveLanguage Fr

The RemoveLanguage directive removes any language associations for files with the given extensions. This allows .htaccess files in subdirectories to undo any associations inherited from parent directories or the server configuration files.

**Parameter:** *extension*

- The *extension* parameter value is any character string that is a valid file extension.

**Note:** If SuffixCaseSense is on (default is off), then the extension argument is case-insensitive. The extension can be specified with or without a leading dot.

### RemoveOutputFilter:

**Module:** mod\_mime

**Syntax:** RemoveOutputFilter *extension* [*extension* ...]

**Default:** none

**Context:** directory, .htaccess

**Override:** none

**Origin:** Apache

**Example:** RemoveOutputFilter .ext

The `RemoveOutputFilter` directive removes any output filter associations for files with the given extensions. This allows `.htaccess` files in subdirectories to undo any associations inherited from parent directories or the server configuration files.

**Parameter:** *extension*

- The *extension* parameter value is any character string that is a valid file extension.

**Note:** If `SuffixCaseSense` is on (default is off), then the extension argument is case-insensitive. The extension can be specified with or without a leading dot.

### RemoveType:

**Module:** `mod_mime`

**Syntax:** `RemoveType extension [ extension...]`

**Default:** none

**Context:** directory, `.htaccess`

**Override:** none

**Origin:** Apache

**Example:** `RemoveType .cgi`

The `RemoveType` directive removes any MIME type associations for files with the given extensions. This allows `.htaccess` files in subdirectories to undo any associations inherited from parent directories or the server config files.

**Parameter:** *extension*

- The *extension* parameter value is any character string that is a valid file extension.

### Example

```
/work/.htaccess:  
RemoveType .cgi
```

The example removes any special handling of `.cgi` files in the `/work/` directory (and any beneath it), causing the files to be treated as the default type.

**Note:** `RemoveType` directives are processed after any `AddType` directives, so it is possible they may undo the effects of the latter if both occur within the same directory configuration. If `SuffixCaseSense` is on (default is off), then the extension argument is case-insensitive. The extension can be specified with or without a leading dot.

### SuffixCaseSense:

**Module:** `mod_mime`

**Syntax:** `SuffixCaseSense on | off`

**Default:** `SuffixCaseSense off`

**Context:** server config

**Override:** none

**Origin:** IBM

**Example:** `SuffixCaseSense on`

The `SuffixCaseSense` directive is used to specify whether the server should distinguish between uppercase and lowercase characters when it has to compare file extensions to the extension patterns on the following directives:

- `AddType`
- `AddClient`

- AddEncoding
- AddLanguage
- AddCharset
- AddHandler
- AddInputFilter
- AddOutputFilter
- RemoveType
- RemoveClient
- RemoveEncoding
- RemoveLanguage
- RemoveCharset
- RemoveHandler
- RemoveInputFilter
- RemoveOutputFilter

By default, the HTTP Server will not be sensitive to the case of the extensions.

**Parameter:** *on* | *off*

- The *on* parameter value specifies the server will be sensitive to the case of file extensions.
- The *off* parameter value specifies the server will not be sensitive to the case of file extensions.

### TypesConfig:

**Module:** mod\_mime

**Syntax:** TypesConfig *filename*

**Default:** TypesConfig /QIBM/UserData/HTTPPA/conf/mime.types

**Context:** server config

**Override:** none

**Origin:** Apache

**Example:** TypesConfig /conf/mime2.types

The TypesConfig directive sets the location of the MIME types configuration file. Filename is relative to the ServerRoot. This file sets the default list of mappings from filename extensions to content types; changing this file is not recommended. Use the AddType directive instead. The file contains lines in the format of the arguments to an AddType command:

```
MIME-type extension [extension ...]
```

Blank lines, and lines beginning with a hash character (#) are ignored.

**Parameter:** *filename*

- The *filename* parameter value is a filename where the MIME-type file can be located. This filename must be relative to the "ServerRoot" on page 326. This restricts the file to the IFS file system.

## Module mod\_negotiation

Module mod\_negotiation supports directives for the IBM HTTP Server for i Web server.

### Summary

Content negotiation is the selection of the document that best matches the clients capabilities from one of several available documents. There are two implementations of content negotiation:

- A type-map (a file with the handler type-map) which explicitly lists the files containing the variants.
- A MultiViews search (enabled by the MultiViews “Options” on page 319) where the server does an implicit filename pattern match and makes a choice from the results.

See “Content negotiation for HTTP Server” on page 17 for more information.

## Type maps

A type map has the same format as RFC822 mail headers. It contains document descriptions separated by blank lines, with lines beginning with a pound sign (#) are treated as comments. A document description consists of several header records. Records may be continued on multiple lines if the continuation lines start with spaces. The leading space will be deleted and the lines concatenated. A header record consists of a keyword name, which always ends in a colon, followed by a value. Whitespace is allowed between the header name and value, and between the tokens of value. The headers allowed are:

Header	Description
Content-Encoding	The encoding of the file. The server only recognizes encoding that is defined by an AddEncoding directive. This normally includes the encoding x-compress for compress'ed files, and x-gzip for gzip'ed files. The x-prefix is ignored for encoding comparisons.
Content-Language	The language of the variant, as an Internet standard language tag (RFC 1766). An example is en, meaning English.
Content-Length	The length of the file, in bytes. If this header is not present, then the actual length of the file is used.
Content-Type	<p>The MIME media type of the document, with optional parameters. Parameters are separated from the media type and from one another by a semicolon, with a syntax of name=value. Common parameters include:</p> <p><b>Parameter One:</b> <i>level</i></p> <ul style="list-style-type: none"> <li>• The <i>level</i> parameter is an integer specifying the version of the media type. For text/html, this defaults to '2', otherwise '0'.</li> </ul> <p><b>Parameter Two:</b> <i>qs</i></p> <ul style="list-style-type: none"> <li>• The <i>qs</i> parameter is a floating-point number with a value in the range of '0.0' to '1.0', indicating the relative quality of this variant compared to the other available variants, independent of the client's capabilities. For example, a '.jpeg' file is usually of higher source quality than an '.ascii' file it is attempting to represent a photograph. However, if the resource being represents is ASCII art, then an ASCII file would have a higher source quality than a '.jpeg' file. All Qs values therefore specific to a given source. For example: Content-Type: image/jpeg; Qs=0.8</li> </ul>
URL	The path to the file containing this variant, relative to the map file.

## MultiViews



A MultiViews search is enabled by the MultiViews Option. If the server receives a request for `/some/dir/QIBM` and `/some/dir/QIBM` does not exist, then the server reads the directory looking for all files named `QIBM.*`, and effectively makes up a type map which names all those files, assigning them the same media types and content-encodings it would have if the client had asked for one of them by name. It then chooses the best match to the client's requirements, and returns that document.

### Directives

- “CacheNegotiatedDocs”
- “ForceLanguagePriority”
- “LanguagePriority” on page 428

### CacheNegotiatedDocs:

**Module:** mod\_negotiation  
**Syntax:** CacheNegotiatedDocs *on* | *off*  
**Default:** CacheNegotiatedDocs *off*  
**Context:** server config, virtual host  
**Override:** none  
**Origin:** Apache  
**Example:** CacheNegotiatedDocs *on*

The CacheNegotiatedDocs directive allows content-negotiated documents requested using HTTP/1.0 to be cached by proxy servers.

**Parameter:** *on* | *off*

- Setting this directive to *on* could mean that clients behind proxies may retrieve versions of the documents that are not the best match for their abilities. The purpose of this directive is to make cache more efficient. This directive only applies to requests which come from HTTP/1.0 browsers. HTTP/1.1 provides much better control over the caching of negotiated documents, and this directive has no effect in responses to HTTP/1.1 requests.

### ForceLanguagePriority:

**Module:** mod\_negotiation  
**Syntax:** ForceLanguagePriority *None* | *Prefer* | *Fallback* [*Prefer* | *Fallback*]  
**Default:** ForceLanguagePriority *None*  
**Context:** server config, virtual host, directory, `.htaccess`  
**Override:** FileInfo  
**Origin:** Apache  
**Example:** See below.

The ForceLanguagePriority directive uses the given LanguagePriority to satisfy negotiation where the server could otherwise not return a single matching document.

**Parameter:** *None* | *Prefer* | *Fallback*

- The *Prefer* parameter uses LanguagePriority to serve one valid result, rather than returning an HTTP result 300 (MULTIPLE CHOICES) when there are several equally valid choices. If the directives below were given, and the user's Accept-Language header assigned *en* and *de* each as quality `.500` (equally acceptable) then the first matching variant (*en*) will be served.  
LanguagePriority *en* *Fr* *de*  
ForceLanguagePriority *Prefer*
- The *Fallback* parameter uses LanguagePriority to serve a valid result, rather than returning an HTTP result 406 (NOT ACCEPTABLE). If the directives below were given, and the user's

Accept-Language only permitted an *en* language response, but such a variant isn't found, then the first variant from the LanguagePriority list is served.

```
LanguagePriority en Fr de  
ForceLanguagePriority Fallback
```

Both options, *Prefer* and *Fallback*, may be specified, so either the first matching variant from LanguagePriority will be served if more than one variant is acceptable, or the first available document will be served if none of the variants match the client's acceptable list of languages.

**Note:** When specifying both *Prefer* and *Fallback* options, the behavior is the same regardless of the order in which they are specified.

See DefaultLanguage, AddLanguage and "LanguagePriority" for more information.

### LanguagePriority:

**Module:** mod\_negotiation

**Syntax:** LanguagePriority *MIME-lang* [*MIME-lang*...]

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** FileInfo

**Origin:** Apache

**Example:** LanguagePriority en Fr de

The LanguagePriority directive sets the precedence of language variants for the case where the client does not express a preference when handling a MultiViews request. The list of MIME-lang are in order of decreasing preference.

**Parameter:** *MIME-lang*

- The *MIME-lang* parameter is any Internet standard language tag or MIME language designation.

This directive may be configured multiple times in a container. The directives are processed from the first to the last occurrence.

**Note:** This directive only has an effect if a best language cannot be determined by any other means. If the client expresses a language preference, this directive has no effect on the file selected during content negotiation.

## Module mod\_proxy

Module mod\_proxy supports directives for the IBM HTTP Server for i Web server.

### Summary

Directives for forward proxy function are as follows:

**Required:** ProxyRequests

**Optional:** AllowCONNECT, ProxyBlock, ProxyDomain, ProxyReceiveBufferSize, ProxyVia

Directives for reverse proxy function are as follows:

**Required:** ProxyPass

**Optional:** ProxyBlock, ProxyPassReverse, ProxyReceiveBufferSize, ProxyVia

Directives for proxy chaining function are as follows:

**Required:** ProxyRemote

**Optional:** NoProxy, (see forward or reverse proxy, above, for additional directives).

For a detailed description of these proxy functions and how they may be used, see “Proxy server types and uses for HTTP Server” on page 24.

**Note:** The mod\_proxy directives require the following LoadModules in HTTP Server configuration file:

```
LoadModule proxy_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_ftp_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_http_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_connect_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

### Directives

- “AllowCONNECT”
- “NoProxy” on page 430
- “<Proxy>” on page 431
- “ProxyBadHeader” on page 433
- “ProxyBlock” on page 433
- “ProxyCacheOnly” on page 434
- “ProxyDomain” on page 435
- “ProxyErrorOverride” on page 436
- “ProxyForceCacheCompletion” on page 437
- “ProxyIOBufferSize” on page 438
- “<ProxyMatch>” on page 438
- “ProxyMaxForwards” on page 439
- “ProxyNoCache” on page 441
- “ProxyNoConnect” on page 442
- “ProxyPass” on page 443
- “ProxyPassReverse” on page 447
- “ProxyPassReverseCookieDomain” on page 449
- “ProxyPassReverseCookiePath” on page 449
- “ProxyPreserveHost” on page 449
- “ProxyReceiveBufferSize” on page 451
- “ProxyRemote” on page 451
- “ProxyRemoteMatch” on page 452
- “ProxyRequests” on page 452
- “ProxyReverse” on page 453
- “ProxyTimeout” on page 454
- “ProxyVia” on page 455

### AllowCONNECT:

**Module:** mod\_proxy

**Syntax:** AllowCONNECT *port\_list*

**Default:** AllowCONNECT 443 563

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule proxy_connect_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_ftp_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_http_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

**Example:** AllowCONNECT 443 563 1070 8088

The AllowCONNECT directive specifies a list of port numbers the server allows clients to specify when using the CONNECT method. Clients use the CONNECT method when HTTPS connections are requested and proxy tunneling over HTTP is in effect. By default, only the default HTTPS port (443) and the default SNEWS port (563) are enabled. Use this directive to override the default and only allow connections that use one of the listed ports.

**Parameter:** *port\_list*

- The *port\_list* parameter can consist of a string of port numbers separated by spaces (see example).

**Example**

```
AllowCONNECT 443 563 1070 8088
```

ProxyBlock may be used to block incoming requests prior to this directive's consideration. Setting ProxyRequests to off negates this directive.

This directive may be configured multiple times in a container. The directives are processed from the first to the last occurrence.

## NoProxy:

**Module:** mod\_proxy

**Syntax:** NoProxy *domain | subnet | ipaddr | hostname [domain | subnet | ipaddr | hostname ...]*

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule proxy_connect_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_ftp_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_http_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

**Example:** NoProxy .mycompany.com 192.168.112.0/21

The NoProxy directive specifies a list of domains, subnets, IP addresses, and/or hosts (in any combination) separated by spaces. Multiple NoProxy directives are allowed. Items in each list are used to match requests for which the server should attempt to handle directly rather than going through a remote proxy server (specified using the ProxyRemote directive). When a client sends a request that matches one or more listed items, the server attempts to connect directly to the server specified in the URL rather than to a remote proxy (specified by ProxyRemote) to chain the request.

**Parameter:** *domain | subnet | ipaddr | hostname*

- A *domain* is a partially qualified DNS domain name, preceded by a period. It represents a group of hosts that logically belong to the same DNS domain or zone (that is, the suffixes of the hostnames are all ending in Domain).
- A *subnet* is a partially qualified Internet address in a numeric (dotted quad) form, optionally followed by a slash (/) and the netmask, specified as the number of significant

bits in the subnet. It is used to represent a subnet of hosts that can be reached over a common network interface. In the absence of the explicit netmask it is assumed that omitted (or zero valued) trailing digits specify the mask. In this case, the netmask can only be multiples of '8 bits' wide. For example, the subnet '192.168.0.0' with an implied netmask of '16' valid bits (sometimes used in the netmask form 255.255.0.0).

- An *ipaddr* represents a fully qualified Internet address in numeric (dotted quad) form. Usually this address represents a host, but there need not necessarily be a DNS domain name connected with the address. For example: 192.168.123.7
- A *hostname* is a fully qualified DNS domain name that can be resolved to one or more IP addresses via the DNS domain name service. It represents a logical host (in contrast to domain, see above) and must be resolvable to at least one *ipaddr* (or often to a list of hosts with different IP addresses).

### Example

```
ProxyRemote * http://firewall.mycompany.com:81
NoProxy .mycompany.com 192.168.112.0/21
```

- ProxyBlock may be used to block incoming requests prior to consideration for this directive.
- This directive is commonly used in conjunction with the ProxyRemote and ProxyDomain directives for directing proxy requests within intranets.
- Setting ProxyNoConnect to on negates this directive.

This directive may be configured multiple times in a container. The directives are processed from the first to the last occurrence.

**Note:** Hostname and domain name comparisons are done without regard to the case, and are always assumed to be anchored in the root of the DNS tree.

### <Proxy>:

**Module:** mod\_proxy

**Syntax:** <Proxy *criteria*> ... </Proxy>

**Default:** none

**Context:** server config, virtual host, Not in Limit

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule proxy_connect_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_ftp_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_http_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
```

**Example:** Forward proxy

```
<Proxy http://www.ibm.com/>
  Allow from All
  Order Allow,Deny
</Proxy>
```

**Example:** Reverse proxy

```
<Proxy /docs/>
  Allow from All
  Order Allow,Deny
</Proxy>
```

The <Proxy> and </Proxy> directives are used to enclose (or contain) a group of directives that apply only to proxy requests that match the specified criteria. Multiple proxy containers are allowed, however they may not be nested. Requests that do not match container criteria are outside the context of the

enclosed directives. Any directive allowed within a directory context is also allowed within a proxy context (see <Directory> for details on directory containers).

**Parameter:** *criteria*

- The *criteria* parameter accepts a partial URL or virtual directory path used to identify requests to which the enclosed directives apply. Partial URLs are used to identify both forward and reverse proxy requests. A match is considered by comparing request URL strings to the specified criteria string, starting with the first character. A match is made if the two strings are identical, up to the length of the criteria string.

Refer to <ProxyMatch> for details regarding the use of regular expression criteria for proxy containers.

Directives within proxy containers apply only to matched requests handled by the proxy function (including both forward and reverse proxy). Requests not handled by the proxy function are not affected by directives within proxy containers.

**Example One**

```
<Proxy /user/local/httpd/htdocs>
  Allow from All
  Order Allow,Deny
</Proxy>
```

**Note:** Previously, directory containers were used to enclose groups of directives that applied to proxy requests by appending the prefix "proxy:" to the beginning of the directory name criteria specified for <Directory> or <DirectoryMatch> directives. This is no longer supported. The proxy function now ignores directives enclosed in <Directory> (or <File>) containers.

Directives within <Location> containers (if matched) take precedence over directives within <Proxy> containers. See <Location> or <LocationMatch> for more information on <Location> containers.

When request URLs match criteria strings of multiple proxy containers, directives within all matched containers are combined and applied. <Proxy> sections are processed in the order they appear in the configuration file. The following is an example of how directives are combined and applied according to order.

**Example Two: Forward Proxy**

```
ProxyRequest on
<Proxy http://>
  Deny from All
  ServerSignature on
</Proxy>
<Proxy http://www.ibm.com/>
  Allow from All
</Proxy>
```

For this example, a request for `http://www.ibm.com/docs/whitepaper.pdf` matches criteria specified for both proxy containers, therefore the server applies the directives within both containers. Since the criteria specified for the second container (<Proxy `http://www.ibm.com/`>) is more specific (a better match) than the criteria specified for the first container (<Proxy `http://`>) directives enclosed within the second container take precedence. The request is therefore allowed since the second container has an "Allow from All" directive. The `ServerSignature` directive would be applied to this request as well (if needed). A request for `http://web.samples.org/welcome.htm`, however, only matches the criteria for the first container, and is therefore denied since this container has a "Deny from All" directive.

If request URLs match criteria strings for one or more `<Proxy>` directives as well as regular expression criteria for one or more `<ProxyMatch>` directives, the server applies matched `<Proxy>` and `<ProxyMatch>` container directives in the order they appear in the configuration file.

**Example:**

```
ProxyRequest on
<Proxy http://www.ibm.com/>
  Allow from All
</Proxy>
<ProxyMatch ^(.*)>
  Deny from All
</ProxyMatch>
```

A request for `http://www.ibm.com/welcome.html` matches criteria specified for both proxy containers, therefore the server applies the directives within both containers. Directives for the `<Proxy>` container are applied first, then directives for the `<ProxyMatch>` container. Due to the order that directives are applied, the request is denied since the "Deny from All" directive (from the `<ProxyMatch>` container) is applied last, even though the `<Proxy>` container is a more exact match.

**Note:** Setting `ProxyRequests` to *off* does not negate this directive. It is available regardless of the forward proxy state.

**ProxyBadHeader:**

**Module:** mod\_proxy  
**Syntax:** ProxyBadHeader *IsError* | *Ignore* | *StartBody*  
**Default:** ProxyBadHeader *IsError*  
**Context:** server, virtual host  
**Override:** none  
**Origin:** Apache  
**Example:** ProxyBadHeader *Ignore*

This directive tells the server how to handle a bad header line in a response. The value *ignore* means the proxy ignores the bad header and continues. The value *IsError* means that the proxy fails out on the request. The value *StartBody* means that proxy (if it has seen other headers before this bad one) starts sending the rest of the headers as body and hopes that the server can handle it.

**ProxyBlock:**

**Module:** mod\_proxy  
**Syntax:** ProxyBlock *word* | *host* | *domain* [*word* | *host* | *domain* ...]  
**Default:** none  
**Context:** server config, virtual host  
**Override:** none  
**Origin:** Apache  
**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows:  
LoadModule proxy\_connect\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM  
LoadModule proxy\_ftp\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM  
LoadModule proxy\_http\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM  
LoadModule proxy\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM  
**Example:** ProxyBlock somecompany.com www-1.ibm.com www-2.ibm.com

The ProxyBlock directive specifies a list of words, hosts, and/or domains (in any combination), separated by spaces. Multiple ProxyBlock directives are allowed. Requests to sites whose URLs contain matched

words, hosts, or domains are blocked by the server. At startup the server attempts to determine list item IP addresses, that may be host names, and records them for a match test.

**Parameter:** *word* | *host* | *domain*

- A *word* can be any keyword (for example, ProxyBlock hello server good-bye).
- A *host* is a fully qualified DNS domain name that can be resolved to one or more IP addresses via the DNS domain name service. It represents a logical host (in contrast to domain, see below) and must be resolvable to at least one IP address (or often to a list of hosts with different IP addresses), otherwise it is simply treated as a word (see above).
- A *domain* is a partially qualified DNS domain name, preceded by a period. It represents a group of hosts that logically belong to the same DNS domain or zone (that is, the suffixes of the hostnames are all ending in Domain).

#### Example

```
ProxyBlock ibm.com www-1.ibm.com www-2.ibm.com server hello
```

The 'www-2.ibm.com' would also be matched if referenced by IP address since the server records addresses at startup for a match test. Note that either 'ibm.com' or 'ibm' is sufficient to match both 'www-1.ibm.com' and 'www-2.ibm.com' by word. However, their corresponding IP addresses would not be blocked since the server could not determine their addresses without having their hostnames specifically listed.

**Note:** " ProxyBlock \*" effectively blocks requests to all sites and therefore should be avoided.

#### ProxyCacheOnly:

**Module:** mod\_proxy

**Syntax:** ProxyCacheOnly *word* | *host* | *domain* [*word* | *host* | *domain* ...]

**Default:** none (meaning cache all documents satisfying other caching directives)

**Context:** server config, virtual host

**Override:** none (meaning cache all documents satisfying other caching directives)

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule proxy_connect_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_ftp_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_http_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

**Example:** ProxyCacheOnly ibm.com www-1.ibm.com www-2.ibm.com

The ProxyCacheOnly directive specifies a list of words, hosts, and domains (in any combination), separated by spaces. Multiple ProxyCacheOnly directives are allowed. Listed items are used to match requests for which the server should cache documents if caching is enabled. The server may then serve cached documents for subsequent requests. The server will also attempt to determine list item IP addresses and records them for a match test.

If this directive is absent, all documents satisfying all other caching directives (for example, ProxyNoCache, CacheMaxFileSize, CacheMinFileSize, etc.) are cached. If this directive is present, only documents from matched words, hosts, or domains are cached (as long as they also satisfy all other caching directives).

**Parameter:** *word* | *host* | *domain*

- A *word* can be any keyword (for example, ProxyCacheOnly hello server good-bye).
- A *host* is a fully qualified DNS domain name that can be resolved to one or more IP addresses via the DNS domain name service. It represents a logical host (in contrast to



domain, see below) and must be resolvable to at least one IP address (or often to a list of hosts with different IP addresses), otherwise it is simply treated as a word (see above).

- A *domain* is a partially qualified DNS domain name, preceded by a period. It represents a group of hosts that logically belong to the same DNS domain or zone (that is, the suffixes of the hostnames are all ending in Domain).

### Example

```
ProxyCacheOnly ibm.com www-1.ibm.com sample.server.edu
```

For this example, 'sample.server.edu' would also be matched if referenced by IP address since the server records addresses at startup for a match test. Note that 'sample', 'server', 'edu', 'sample.server', or 'server.edu' is sufficient to match 'sample.server.edu' by word, however documents for requests using IP addresses corresponding to 'sample.server.edu' would not be cached since the server could not determine the addresses unless the hostname is specifically listed.

- CacheMinFileSize, CacheMaxFileSize, and CacheTimeMargin may make documents ineligible for cache prior to consideration for this directive.
- ProxyNoCache provides counter function. Documents matching a previous ProxyNoCache template in the configuration will not be cached, regardless of whether they match a subsequent ProxyCacheOnly template. In other words, a ProxyNoCache directive may override a ProxyCacheOnly directive if configured prior to the ProxyCacheOnly directive.
- This directive is used only if CacheRoot is set.
- Setting ProxyNoConnect to off negates this directive.

**Note:** "ProxyCacheOnly \*" enables caching for all documents if not preceded and matched by a ProxyNoCache directive.

### ProxyDomain:

**Module:** mod\_proxy

**Syntax:** ProxyDomain *domain*

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule proxy_connect_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

```
LoadModule proxy_ftp_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

```
LoadModule proxy_http_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

```
LoadModule proxy_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

**Example:** ProxyDomain .mycompany.com

The ProxyDomain directive specifies the default domain to which the server belongs when acting as a forward proxy server. If a request specifies a host without a domain name, the server sends a response that redirects the client to the host with the configured domain appended. Possible values include all domain names starting with a dot (or period) and consisting only of the characters AZ, AZ, '.' (dot), '-' (dash), and 0-9.

**Parameter:** *domain*

- The *domain* is a partially qualified DNS domain name, preceded by a period. It represents a group of hosts that logically belong to the same DNS domain or zone (that is, the suffixes of the hostnames are all ending in Domain).

## Example

```
ProxyRemote * http://firewall.mycompany.com:81
NoProxy .mycompany.com 192.168.112.0/21
ProxyDomain .mycompany.com
```

For this example, if an unqualified request for `http://myserver/` comes in, the server will redirect the client to a fully qualified host name using the default domain. That is, the client will be redirected to `http://myserver.mycompany.com/`.

- ProxyBlock may be used to block incoming requests prior to consideration for this directive.
- This directive is commonly used in conjunction with the NoProxy and ProxyRemote directives for directing proxy requests within intranets.
- Setting ProxyRequests to off negates this directive

## ProxyErrorOverride:

**Module:** mod\_proxy

**Syntax:** ProxyErrorOverride *on* | *off*

**Default:** ProxyErrorOverride off

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule proxy_connect_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_ftp_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_http_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

**Example:** ProxyErrorOverride on

The ProxyErrorOverride directive specifies if the server is to override error response codes and message text sent by remote servers to enable local error messaging for remote server problems. If disabled (the default), all responses sent by remote servers (including errors) are relayed to clients (local error messaging is not used). If enabled, server related error codes and messages sent by remote servers (codes greater than or equal to 400<sup>®</sup>) are overridden and local error messaging is used to send responses that pertain to the local server, rather than the remote server. Non-server related error codes (codes less than 400) are not affected by this directive and are always relayed to clients.

**Parameter:** *on* | *off*

- If *off*, is specified (the default), all response codes and messages sent by remote servers are relayed to clients (unaltered).
- If *on* is specified, error response codes and messages sent by remote servers relating to server problems are overridden and local error messaging is used to send responses to clients.

By default, local error messaging will send hardcoded messages to clients. However, it may be configured to send custom web pages as well, or to redirect certain errors to local CGI programs (or servlets) or remote servers to handle. When ProxyErrorOverride is used in conjunction with ErrorDocument support, custom responses may be sent to clients when proxy requests fail due to remote server problems. This is useful for reverse proxy setups where remote server problems need to be concealed from clients or when web sites must have a common error reporting appearance. It may be used, however, for any proxy setup where remote server errors need to be handled in a certain (customized) manner.

For example, suppose the local server has address `http://www.ibm.com/` and the following directives are setup for reverse proxy:

```
ProxyPass /docs/ http://pubserver.ibm.com/public/documentation/  
ProxyErrorOverride on  
ErrorDocument proxyrmterror /cgi-bin/proxyerr.pgm
```

Now further suppose the local server was sent the request `http://www.ibm.com/docs/whitepaper.html`. The `ProxyPass` directive will cause the request to be internally converted into a request for `http://pubserver.ibm.com/public/documentation/whitepaper.html`. The proxy function will then be invoked to retrieve `/public/documentation/whitepaper.html` from `pubserver.ibm.com`. The remote server (`pubserver.ibm.com`) then has an error that causes it to return response code 500 (internal error) to the local server (`www.ibm.com`). Since `ProxyErrorOverride` is enabled, the local server overrides the response code (along with any message text) and enables local error messaging to handle the response. Furthermore, since `ErrorDocument` is setup for such a response (`proxyrmterror`), the error is passed to the cgi program `/cgi-bin/proxyerr.pgm` which handles the problem by sending a customized error page to the client.

In this example of a reverse proxy request process, internal server errors from a remote server (`pubserver.ibm.com`) are concealed from the client since local error messaging is enabled for proxy requests on `www.ibm.com`. Similar handling may be setup for forward proxy scenarios as well.

- If custom error messages are not defined (not enabled via `ErrorDocument`), local error messaging may still be used to send hardcoded messages pertaining to the local server.
- Setting `ProxyRequests` to off does not negate this directive. It is available regardless of the forward proxy state.

#### | **ProxyForceCacheCompletion:**

| **Module:** `mod_proxy`

| **Syntax:** `ProxyForceCacheCompletion percentage`

| **Default:** `ProxyForceCacheCompletion 90`

| **Context:** server config, virtual host

| **Override:** none

| **Origin:** Apache

| **Usage Considerations:** Multiple `LoadModule` directives are required in the configuration file prior to using the directive. The statements should be as follows:

| `LoadModule proxy_connect_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM`

| `LoadModule proxy_ftp_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM`

| `LoadModule proxy_http_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM`

| `LoadModule proxy_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM`

| **Example:** `ProxyForceCacheCompletion 60`

| The `ProxyForceCacheCompletion` directive specifies a download percentage used to determine whether the server should continue to cache documents after a client cancels a request. If a request for a document is canceled, the server will complete the cache transfer over the connection with the content server if more than the percentage specified has already been received. If the server has received less than the percentage specified, or if the proxy caching function is not enabled (see `CacheRoot` for details), all data is discarded and the server drops the connection with the content server.

| **Note:** This directive is used only if `CacheRoot` is set. In addition, if the `ProxyNoConnect` directive is set to on, it negates the `ProxyForceCacheCompletion` setting.

| **Parameter:** *percentage*

| The *percentage* parameter accepts an integer value between 0 and 100 to specify the minimum amount of data the server is to receive (specified as a percentage of the whole document) to continue caching a document, regardless of whether the client's request is canceled.

## | ProxyIOBufferSize:

**Module:** mod\_proxy

**Syntax:** ProxyIOBufferSize bytes

**Default:** ProxyIOBufferSize 8192

**Context:** server, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule proxy_connect_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_ftp_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_http_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
```

**Example:** CacheTimeMargin 300

The ProxyIOBufferSize directive adjusts the size of the internal buffer, which is used as a scratchpad for the data between input and output. The size must be less or equal 8192, and it is recommended that you do not change the size.

## <ProxyMatch>:

**Module:** mod\_proxy

**Syntax:** <ProxyMatch *criteria*> ... </ProxyMatch>

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule proxy_connect_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_ftp_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_http_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
```

**Example:** Reverse proxy

```
ProxyReverse on
ProxyPass /docs/v4r4m0/ http://pubserver.ibm.com/public/v4r4m0/
<ProxyMatch "^http://pubserver.ibm.com/public/v[0-9]r[0-9]m[0-9]/(.*)">
  Allow from All
  Order Allow,Deny
</ProxyMatch>
```

**Example:** Forward proxy

```
<ProxyMatch "^http://server[0-9]r[0-9]m[0-9]/docs/*">
  Allow from All
  Order Allow,Deny
</ProxyMatch>
```

The <ProxyMatch> directive is used to enclose a group of directives that apply only to proxy requests that match the specified criteria. Multiple proxy containers are allowed, however they may not be nested. Requests that do not match container criteria are outside the context of the enclosed directives. Any directive allowed within a directory context is also allowed within a proxy context.

### **Parameter:** *criteria*

- The *criteria* parameter accepts a UNIX-style extended regular expression used to identify requests to which the enclosed directives apply. Expressions are used to identify both forward and reverse proxy requests. A match is considered by comparing request URL strings to the specified expression. Subexpressions are grouped within parentheses. Then,

parenthetically enclosed regular expressions are substituted in a subsequent \$n statement. A match is made if the URL string matches the expression using regular expression logic. For reverse proxy, the specified expression must match the new outgoing URL.

Proxy containers defined by <ProxyMatch> directives (including the directives enclosed by them) are handled in the same way as those defined by <Proxy> directives. The only difference is in how the criteria is specified and handled using regular expressions (for <ProxyMatch>) rather than string literals (for <Proxy>). Refer to <Proxy> for further details regarding proxy containers.

For example, suppose the local server has address `http://as400.ibm.com/` and the following directives are setup for reverse proxy:

#### Example

```
ProxyPass /v4r3m0/docs/ http://pubserver.ibm.com/public/vrm430/
ProxyPass /v4r4m0/docs/ http://pubserver.ibm.com/public/vrm440/
ProxyPass /v4r5m0/docs/ http://pubserver.ibm.com/public/vrm450/
ProxyPass /v5r1m0/docs/ http://pubserver.ibm.com/public/vrm510/
<ProxyMatch "^http://pubserver.ibm.com/public/v[0-9]r[0-9]m[0-9]/(.*)">
  AuthName "i Document Server"
  AuthType Basic
  Require group admin
  PasswdFile QUSRSYS/DOC_USERS
  GroupFile /groups/doc_readers
</ProxyMatch>
```

For this example, a request for `/v4r5m0/docs/manual.html` is identified as a proxy request since it matches the third ProxyPass statement (`ProxyPass /v4r5m0/docs/http://pubserver.ibm.com/public/vrm450/`). Once identified as a proxy request, it is compared against criteria specified for the proxy container (`ProxyMatch "^http://pubserver.ibm.com/public/v[0-9]r[0-9]m[0-9]/(.*)"`) using regular expression logic. A match is made and the server applies the directives within the container that requires the client to provide basic authentication credentials (`AuthType Basic`). If the client is authenticated (`PasswdFile QUSRSYS/DOC_USERS`) and authorized (`GroupFile /groups/doc_readers`, or `Require group admin`) the request will be internally converted into a request for `http://publicserver.ibm.com/public/vrm450/manual.html` and further handled by the proxy function (see “ProxyPass” on page 443 for more information on reverse proxy). If the client is not authenticated or authorized, the request fails.

- The client is authenticated if a valid userid and password is provided, according to the `PasswdFile` directive.
- The client is authorized if the userid (or group) is allowed access, according to the `GroupFile` or `Require` directives.

Notice that in the above example the directives enclosed in the proxy container will apply to requests matching any of the ProxyPass directives since the regular expression criteria (specified for <ProxyMatch>) matches all four virtual directory path names specified for ProxyPass.

- Setting `ProxyRequests` to off does not negate this directive. It is available regardless of the forward proxy state.

#### ProxyMaxForwards:

**Module:** mod\_proxy

**Syntax:** ProxyMaxForwards *maximum*

**Default:** ProxyMaxForwards 10

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule proxy_connect_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_ftp_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_http_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
```

**Example:** ProxyMaxForwards 8

The ProxyMaxForwards directive specifies the value the server is to use when adding Max-Forwards request headers to requests that do not contain a Max-Forwards header. When the server receives requests that do not contain a Max-Forwards header, it automatically adds one using the specified value. This setting is not used for requests that already contain a Max-Forwards header.

**Parameter:** *maximum*

- The *maximum* parameter accepts an integer value between 1 and 2,147,483,648 to specify the value the server is to use when it adds Max-Forwards request headers to proxy requests.

The server uses Max-Forwards headers to prevent infinite proxy loops, and possibly certain types of denial of service attacks. This is accomplished by ensuring that a Max-Forwards header is set for all requests to control the maximum number of times it can be forwarded (or passed to subsequent servers).

When the server receives requests containing a Max-Forwards header, it will continue to process the requests only if the value for the header is greater than 0 (zero). If the value is greater than zero, the server decrements it and continues to process the request. If the request subsequently needs to be forwarded to another server, the Max-Forwards header is sent with the decremented value. This process is repeated until the request is fulfilled (or rejected) by a server, or until the value for the Max-Forwards header reaches zero. Once the value reaches zero (or less), the server will not forward the request and will respond immediately (see example, request 3) with the following response codes:

- If TRACE method is used, 200 (OK) is returned as well as any trace data.
- If OPTIONS method is used, 200 (OK) is returned as well as any options data.
- If any other method is used, 502 (BAD\_GATEWAY) is returned as well as the server's customized error page for "proxyfail" (if enabled, see "ErrorDocument" on page 294).

This setting is used for both forward and reverse proxy requests.

**Example: Forward Proxy**

```
ProxyRequests on
ProxyMaxForwards 8
```

For this example, consider the following three requests:

**Request 1**

```
GET http://docserver.ibm.com/manual.pdf HTTP/1.0
```

For this request, the server will use the value specified for ProxyMaxForwards (8) to add the new header "Max-Forwards : 8" to the request (since it is not already present), and then forward it to docserver.ibm.com as:

```
GET /manual.pdf HTTP/1.0
Max-Forwards : 8
```

**Request 2**

```
GET http://docserver.ibm.com/manual.pdf HTTP/1.0
Max-Forwards : 3
```

For this request, the server will decrement the value for the Max-Forwards header to 2, and then forward the request to docserver.ibm.com as:

```
GET /manual.pdf HTTP/1.0
Max-Forwards : 2
```

In this case, the value specified for ProxyMaxForwards is not used since the request already contained a Max-Forwards header.

### Request 3

```
GET http://docserver.ibm.com/manual.pdf HTTP/1.0
Max-Forwards : 0
```

For this request, the server will immediately return response code 502 (BAD\_GATEWAY) since the request cannot be forwarded any further due to the Max-Forwards header value. In this case, docserver.ibm.com is never contacted.

- Setting ProxyRequests to *off* does not negate this directive. It is available regardless of the forward proxy state.

### ProxyNoCache:

**Module:** mod\_proxy

**Syntax:** ProxyNoCache *word* | *host* | *domain*

**Default:** absent [meaning cache all files satisfying other caching directives]

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule proxy_connect_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_ftp_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_http_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
```

**Example:** ProxyNoCache ibm.com www-1.ibm.com sample.example.edu

The ProxyNoCache directive specifies a list of words, hosts, and domains (in any combination), separated by spaces. HTTP and non-passworded FTP documents from matched words, hosts or domains are not cached by the proxy server. The proxy module will also attempt to determine IP addresses of list items, that may be hostnames during startup, and cache them for a match test. If this directive is absent, all documents satisfying all other caching directives (for example: ProxyCacheOnly, CacheMaxFileSize, CacheMinFileSize, etc.) are cached. If this directive is present, documents from matched words, hosts or domains are not cached.

**Parameter:** *word* | *host* | *domain*

- A *word* can consist of any combination of keywords (for example, ProxyNoCache hello server good-bye).
- The *host* is a fully qualified DNS domain name that can be resolved to one or more IP address via the DNS domain name service. It represents a logical host (in contrast to domain, see above) and must be resolvable to at least one IP address (or often to a list of hosts with different IP addresses).
- The *domain* is a partially qualified DNS domain name, preceded by a period. It represents a list of hosts that logically belong to the same DNS domain or zone (that is, the suffixes of the hostnames are all ending in Domain).

### Example

```
ProxyNoCache ibm.com www-1.ibm.com sample.example.edu
```

The 'sample.example.edu' would also be matched if referenced by IP address. Note that 'example' is sufficient to match 'example.edu'.

- ProxyCacheOnly provides counter function. Documents matching a previous ProxyCacheOnly template in the configuration will be cached, regardless of whether they match a subsequent ProxyNoCache template. In other words, a ProxyCacheOnly directive may override a ProxyNoCache directive if configured prior to the ProxyNoCache directive.
- This directive is used only if CacheRoot is set.
- Setting ProxyRequests to *off* negates this directive.

**Note:** "ProxyNoCache \*" disables caching for all documents if not preceded by the ProxyCacheOnly directive, however garbage collection is not affected.

### ProxyNoConnect:

**Module:** mod\_proxy

**Syntax:** ProxyNoConnect *on* | *off*

**Default:** ProxyNoConnect *off*

**Context:** server config, virtual host

**Override:** none

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule proxy_connect_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_ftp_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_http_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

**Example:** ProxyNoConnect *off*

The ProxyNoConnect directive specifies if the proxy is to connect to remote content servers to retrieve documents. If the server is not allowed to connect to remote content servers, it can only serve documents from cache.

**Parameter:** *on* | *off*

- If *off* is specified, the server may serve documents from cache (if enabled) as well as issue outgoing requests to remote servers to retrieve servable documents (see Example 1, below).
- If set to *on* is specified, the proxy may only serve documents from cache (if enabled). It will not establish outgoing connections with remote servers. CacheRoot is required if *on* is specified (see Example 2, below).

#### Example 1

```
ProxyRequests on
ProxyNoConnect off
CacheRoot /QOpenSys/UserData/HTTPPA/CacheRoot/myproxy
```

In this example, the proxy may serve documents from cache as well as issue outgoing requests to remote servers.

#### Example 2

```
ProxyRequests on
ProxyNoConnect on
CacheRoot /QOpenSys/UserData/HTTPPA/CacheRoot
```

In this example, the proxy may only serve documents from cache. Documents will not be retrieved from remote servers since outgoing connections are not permitted. Since the server is not permitted to retrieve documents, items in cache must be managed by another application or process other than the server itself.



- CacheRoot is required if this directive is set to *on*.
- The ProxyNoConnect directive causes the AllowCONNECT directive to be ineffective. If ProxyNoConnect is present, and AllowCONNECT is also specified, then even if the AllowCONNECT allows a SSL connection to be made on a specific port, the ProxyNoConnect directive dictates that no connections are allowed.

### ProxyPass:

**Module:** mod\_proxy

**Syntax:** ProxyPass *path url* | ! [*key=value key=value* ]

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule proxy_connect_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_ftp_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_http_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
```

### Example:

```
ProxyPass /docs/confidential/ !
ProxyPass /docs/ http://pubserver.ibm.com/public/documentation/
```

This directive allows remote servers to be mapped into the space of the local server; the local server does not act as a proxy in the conventional sense, but appears to be a mirror of the remote server. *path* is the name of a local virtual path; *url* is a partial URL for the remote server and cannot include a query string.

The ProxyPass directive specifies information used either to identify and map requests into the space of remote servers, or to prevent requests from being mapped into the space of remote servers, when the reverse proxy function is enabled. Multiple ProxyPass directives are allowed. When enabled, the server does not act as a proxy in the conventional sense, but appears to be a mirror of remote servers by transforming requests that match specified (virtual) directory paths into proxy requests using a corresponding partial URL. If the reverse proxy function is not enabled, this directive has no affect (see “ProxyReverse” on page 453).

#### Parameter One: *path* | *url*

- The *path* parameter is the name of a local virtual path. When the directive is placed outside a location container, the first parameter accepts a directory name used to identify requests to be handled by the proxy function. The directory name does not need to specify an existing directory, it may be a name used only as a virtual directory for the server.
- The *url* parameter is a partial URL for the remote server. When the directive is placed inside a location container, the first parameter accepts a partial URL used to transform matched requests (for the location container) into proxy requests. When matched, the portion of the original request URL that matches the location container criteria is replaced with the specified partial URL. Mapped requests are then handled by the proxy function (see example two).

#### Parameter Two: *url* | !

- The *url* | ! parameter is used when the directive is placed outside a location container, the second parameter accepts a partial URL or the negation operator (!). Partial URLs are used to transform matched requests into proxy requests by replacing the portion of the original request URL that matches the path parameter (parameter one) with the specified partial URL (parameter two). Mapped requests are then handled by the proxy function. The negation operator is used to prevent requests that match the path parameter

(parameter one) from being mapped and handled by the proxy function, even though they may match a succeeding ProxyPass directive. Example one, below, shows both partial URLs and the negation operator being used for multiple ProxyPass directives.

- When the directive is placed inside a location container a second parameter cannot be specified.

The server functions as a reverse proxy by mapping requests for documents inside virtual directories (specified by the path parameter or location container criteria) into the space of remote servers (specified by the url parameter). It then retrieves the documents (via proxy), and serves them while making it appear to the client as if they originated from the local server.

The negation operator (!) is used to prevent specific virtual subdirectories to be mapped into the space of remote servers, while allowing higher level (parent) directories to be mapped. Order is important in these situations. ProxyPass directives using the negation operator to prevent specific virtual subdirectories from being mapped must be placed before those mapping higher level (parent) directories (see example one).

Suppose the local server has address `http://iseries.ibm.com/`:

#### Example 1

```
ProxyReverse on
ProxyPass /docs/v4r5m0/ http://pubserver.ibm.com/public/v4r5m0/
ProxyPass /docs/archives/confidential/ !
ProxyPass /docs/archives/private/ !
ProxyPass /docs/archives/ http://pubserver.ibm.com/archives/documents/example
```

For this example, since the reverse proxy function is enabled (ProxyReverse on), the first ProxyPass directive will cause a local request for `/docs/v4r5m0/manual.html` to be internally transformed into a request for `http://pubserver.ibm.com/public/v4r5m0/manual.html`. The proxy function will then be used to retrieve `/public/v4r5m0/manual.html` from `pubserver.ibm.com` and return the document to the requesting client. In this way, a virtual `/docs/v4r5m0/` directory on the local server (`as400.ibm.com`) appears as a mirror of the `/public/v4r5m0/` directory of the remote server (`pubserver.ibm.com`). A request for `/docs/archives/20020101.log` will be handled in a similar way, using the last ProxyPass directive (`ProxyPass /docs/archives/ http://pubserver.ibm.com/archives/documents/`). However, a request for `/docs/archives/confidential/secrets.txt` will not be handled by the proxy function since the second ProxyPass directive prohibits any request for documents within the `/docs/archives/confidential/` virtual subdirectory. Likewise, the third ProxyPass directive prohibits any request for documents within the `/docs/archives/private/` virtual subdirectory.

The following example shows the ProxyPass directive being used within a location container to obtain results similar to example 1.

#### Example Two

```
ProxyReverse on
<Location /docs/v4r5m0/>
  ProxyPass http://pubserver.ibm.com/public/v4r5m0/
</Location>
ProxyPass /docs/archives/confidential/ !
ProxyPass /docs/archives/private/ !
ProxyPass /docs/archives/ http://pubserver.ibm.com/archives/documents/
```

Notice the first ProxyPass directive is placed within a location container and specifies only one parameter. A local request for `/docs/v4r5m0/manual.html` is identified by matching the location container criteria (`/docs/v4r5m0/`), transformed into a request for `http://pubserver.ibm.com/public/v4r5m0/manual.html` by replacing the matched portion with the ProxyPass parameter, and handled by the proxy function in the same way described for example one.

- “ProxyPassReverse” on page 447 may be used to handle HTTP redirect responses from remote servers.
- Setting “ProxyReverse” on page 453 to *off* negates this directive.
- Setting “ProxyRequests” on page 452 to *off* does not negate this directive. It is available regardless of the forward proxy state.

As of Apache 2.2 or later, the ability to use pooled connections to a backend server is available. Using the key=value parameters it is possible to tune this connection pooling. The default for a Hard Maximum for the number of connections is the number of threads per process in the active MPM. In the Prefork MPM, this is always 1, while with the Worker MPM it is controlled by the ThreadsPerChild. Setting min will determine how many connections will always be open to the backend server. Upto the Soft Maximum or smax number of connections will be created on demand. Any connections above smax are subject to a time to live or ttl. Apache will never create more than the Hard Maximum or max connections to the backend server. For example:

```
ProxyPass /example http://backend.example.com smax=5 max=20 ttl=120 retry=300
```

Table 29. Values table

Parameter	Default	Description
min	0	Minumum number of connections that will always be open to the backend server.
max	1...n	Hard Maximum number of connections that will be allowed to the backend server. The default for a Hard Maximum for the number of connections is the number of threads per process in the active MPM. The Worker MPM it is controlled by the ThreadsPerChild. Apache will never create more than the Hard Maximum connections to the backend server.
smax	max	Upto the Soft Maximum number of connections will be created on demand. Any connections above smax are subject to a time to live or ttl.
ttl	-	Time To Live for the inactive connections above the smax connections in seconds. Apache will close all connections that have not been used inside that time period.
timeout	Timeout	Connection timeout in seconds. If not set the Apache will wait until the free connection is available. This directive is used for limiting the number of connections to the backend server together with max parameter.
acquire	-	If set this will be the maximum time to wait for a free connection in the connection pool. If there are no free connections in the pool the Apache will return SERVER_BUSY status to the client.

Table 29. Values table (continued)

Parameter	Default	Description
keepalive	Off	This parameter should be used when you have a firewall between your Apache and the backend server, which tends to drop inactive connections. This flag will tell the Operating System to send KEEP_ALIVE messages on inactive connections (interval depends on global OS settings, generally 120ms), and thus prevent the firewall to drop the connection. To enable keepalive set this property value to On.
retry	60	Connection pool worker retry timeout in seconds. If the connection pool worker to the backend server is in the error state, Apache will not forward any requests to that server until the timeout expires. This enables the ability to shut down the backend server for maintenance, and bring it back online later.
loadfactor	1	Worker load factor. Used with BalancerMember. It is a number between 1 and 100 and defines the normalized weighted load applied to the worker.
route	-	Route of the worker when used inside load balancer. The route is a value appended to session id.
redirect	-	Redirection Route of the worker. This value is usually set dynamically to enable safe removal of the node from the cluster. If set, all requests without a session id will be redirected to the BalancerMember that has route parameters equal to this value.

If the Proxy directive scheme starts with the balancer:// then a virtual worker that does not really communicate with the backend server will be created. Instead it is responsible for the management of several "real" workers. In that case the special set of parameters can be added to this virtual worker. Here is an example:

```
ProxyPass /special-area http://special.example.com/ smax=5 max=10
ProxyPass / balancer://mycluster stickysession=jsessionid nofailover=0n
<Proxy balancer://mycluster>
BalancerMember http://1.2.3.4:8009
BalancerMember http://1.2.3.5:8009 smax=10
# Less powerful server, don't send as many requests there
BalancerMember http://1.2.3.6:8009 smax=1 loadfactor=20
</Proxy>
```

When used inside a <Location> section, the first argument is omitted and the local directory is obtained from the <Location>. If you require a more flexible reverse-proxy configuration, see the RewriteRule directive with the [P] flag.

Table 30. Values table

Parameter	Default	Description
lbmethod	-	Balancer load-balance method. Select the load-balancing scheduler method to use. Either byrequests, to perform weighted request counting or bytraffic, to perform weighted traffic byte count balancing. Default is byrequests.
stickysession	-	Balancer sticky session name. The value is usually set to something like JSESSIONID or PHPSESSIONID, and it depends on the backend application server that support sessions.
nofailover	Off	If set to On the session will break if the worker is in error state or disabled. Set this value to On if backend servers do not support session replication.
timeout	0	Balancer timeout in seconds. If set this will be the maximum time to wait for a free worker. Default is not to wait.
maxattempts	1	Maximum number of failover attempts before giving up.

### ProxyPassReverse:

**Module:** mod\_proxy

**Syntax:** ProxyPassReverse *path* | *url*

**Default:** none

**Context:** server config, virtual host, directory

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule proxy_connect_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_ftp_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_http_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

**Example:** ProxyPassReverse /docs/ http://pubserver.ibm.com/public/documentation/

The ProxyPassReverse directive lets Apache adjust the URL in the Location, Content-Location and URI headers on HTTP redirect responses. This is essential when Apache is used as a reverse proxy to avoid by-passing the reverse proxy because of HTTP redirects on the backend servers which stay behind the reverse proxy.

The ProxyPassReverse directive may specify a directory path and a partial URL used to identify and adjust URLs in response headers returned to the client (via proxy). Multiple ProxyPassReverse directives are allowed.

Only the HTTP response headers specifically mentioned above will be rewritten. Apache will not rewrite other response headers, nor will it rewrite URL references inside HTML pages. This means that if the proxied content contains absolute URL references, they will by-pass the proxy.

**Parameter One:** *path* | *url*

- The *path* parameter is the name of a local virtual path. When the directive is placed outside a location container, the first parameter accepts a directory name used to adjust response header values. If URLs specified in response headers match the *url* parameter (parameter two), the portion that matches is replaced with the specified directory name. Adjusted headers are then returned to the client. The directory name does not need to specify an existing directory, it may be a name used only as a virtual directory for the server.
- The *url* parameter is a partial URL for the remote server. When the directive is placed inside a location container, the first parameter accepts a partial URL used to identify URLs in URI, Location, and Content-Location response headers returned to the server as requested by the proxy function. If any of these request headers match the specified partial URL, the portion that matches is replaced with the directory name specified for the location container. Adjust headers are then returned to the client.

**Parameter Two:** *url*

- The *url* parameter is a partial URL for the remote server. When the directive is placed outside a location container, the second parameter accepts a partial URL used to identify URLs in URI, Location, and Content-Location response headers returned to the server as requested by the proxy function.
- When the directive is placed inside a location container a second parameter cannot be specified.

This directive provides support to be used in applications when it is essential that clients are not directed to use URLs that bypass the proxy function. It is mainly intended to provide additional function for reverse proxy, however it may also be applied to forward proxy requests handled by the server.

Suppose the local server has address `http://iseries.ibm.com`:

**Example**

```
ProxyReverse on
ProxyPass /docs/v4r4m0/ http://pubserver.ibm.com/public/v4r4m0/
ProxyPass /docs/v4r5m0/ http://pubserver.ibm.com/public/v4r5m0/
ProxyPass /docs/v5r1m0/ http://pubserver.ibm.com/public/v5r1m0/
ProxyPassReverse /docs/ http://pubserver.ibm.com/public/
ProxyPass /docs/archives/ http://pubserver.ibm.com/archives/
```

For this example, since the reverse proxy function is enabled (ProxyReverse on), a request for `/docs/v4r4m0/api_reference.htm` will be internally transformed into a proxy request for `http://pubserver.ibm.com/public/v4r4m0/API_reference.htm` (the functionality the first ProxyPass directive provides here). The use of ProxyPassReverse adjusts URLs in URI, Location, and Content-Location response headers from `pubserver.ibm.com`. Therefore, when the server's request is subsequently redirected by `pubserver.ibm.com` with the following response:

```
301 "Permanently Moved"
Location: http://pubserver.ibm.com/public/archives/440/API_reference.htm
{other response headers}
```

```
{optional body text}
```

The server changes the matching portion of the URL in the Location header (`http://pubserver.ibm.com/public/`) to the virtual server path (`/docs/`) before sending the following (adjusted) response to the client:

```
301 "Permanently Moved"
Location: http://as400.ibm.com/docs/archives/440/API_reference.htm
{other response headers}

{optional body text}
```

In this way, any new request the client sends due to the redirect response (301 "Permanently Moved") is directed back to the proxy since the Location header is adjusted. The back end server and path name (<http://pubserver.ibm.com/public/>) remain hidden from the client.

- This directive is only useful when used in conjunction with the "ProxyPass" on page 443 directive.
- Setting "ProxyReverse" on page 453 to *off* negates this directive.
- Setting "ProxyRequests" on page 452 to *off* does not negate this directive. It is available regardless of the proxy state.

### ProxyPassReverseCookieDomain:

**Module:** mod\_proxy

**Syntax:** ProxyPassReverseCookieDomain *internal-domain public-domain*

**Default:** none

**Context:** Server, Virtual Host, Directory

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule proxy_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

**Example:** ProxyPassReverseCookieDomain internal.domain.com www.company.com

The ProxyPassReverseCookieDomain directive adjusts the Domain string in Set-Cookie headers from a reverse-proxied server. The usage of the ProxyPassReverseCookieDomain directive is similar to ProxyPassReverse, but instead of rewriting headers that are a URL it rewrites the domain string in Set-Cookie headers.

### ProxyPassReverseCookiePath:

**Module:** mod\_proxy

**Syntax:** ProxyPassReverseCookiePath *internal-path public-path*

**Default:** none

**Context:** Server, Virtual Host, Directory

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule proxy_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

The ProxyPassReverseCookiePath directive adjusts the Path string in Set-Cookie headers from a reverse-proxied server. The usage of the ProxyPassReverseCookiePath directive is similar to ProxyPassReverse, but instead of rewriting headers that are a URL, this rewrites the path string in Set-Cookie headers.

### ProxyPreserveHost:

**Module:** mod\_proxy

**Syntax:** ProxyPreserveHost on | off

**Default:** ProxyPreserveHost off

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule proxy_connect_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_ftp_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_http_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

**Example:** ProxyPreserveHost on

The ProxyPreserveHost directive specifies whether the server is to preserve Host: headers when handling requests using the reverse proxy function.

**Parameter:** *on* | *off*

- If *off* is specified (the default), the server generates Host: headers for requests handled by the reverse proxy function, using the hostname (and optionally a port number) specified for the ProxyPass or RewriteRule directives.
- If *on* is specified, the server uses Host: headers sent with requests, rather than generating Host: headers, and uses the hostname (and optional port) specified for the ProxyPass or RewriteRule directives only to route the request.

Suppose, for example, the local server has the address `http://as400.ibm.com/` with the following directive set up for reverse proxy:

**Example**

```
ProxyPass /docs/ http://pubserver.ibm.com:8080/public/documentation/
ProxyPreserveHost on
```

The server in this example is sent the following request:

```
GET /docs/manual.html HTTP/1.0
Host: virtual-host.ibm.com
{other request headers}
```

{optional body text}

The ProxyPass directive will cause the request to be internally transformed into a request for `http://pubserver.ibm.com:8080/public/documentation/manual.html`, and the ProxyPreserveHost directive will cause the Host: header to be preserved and passed by the proxy function, resulting in the following request sent to `pubserver.ibm.com`:

```
GET /public/documentation/manual.html HTTP/1.0
Host: virtual-host.ibm.com
{other request headers}
```

{optional body text}

If *off* were specified for ProxyPreserveHost, the Host: header would not be preserved. The server, in this case, would generate a Host: header, resulting in the following request:

```
GET /public/documentation/manual.html HTTP/1.0
Host: pubserver.ibm.com:8080
{other request headers}
```

{optional body text}

- “ProxyPassReverse” on page 447 may be used to handle HTTP redirect responses from remote servers.
- Setting “ProxyReverse” on page 453 to *off* negates this directive.
- Setting “ProxyRequests” on page 452 to *off* does not negate this directive. It is available regardless of the forward proxy state.



## ProxyReceiveBufferSize:

**Module:** mod\_proxy

**Syntax:** ProxyReceiveBufferSize *bytes*

**Default:** ProxyReceiveBufferSize 0

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule proxy_connect_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

```
LoadModule proxy_ftp_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

```
LoadModule proxy_http_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

```
LoadModule proxy_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

**Example:** ProxyReceiveBufferSize 2048 The ProxyReceiveBufferSize directive specifies an explicit network buffer size for outgoing HTTP and FTP connections (for increased throughput). This directive effectively overrides the server's default TCP/IP buffer size. Possible values include 0 (zero) and all positive integers greater than or equal to 512 (the maximum value is 2,147,483,647 bytes). The value 0 (zero) indicates the system's default buffer size should be used.

**Parameter:** *bytes*

- The *bytes* parameter has to be greater than '512' or set to '0' to indicate that the system's default buffer size should be used.

## ProxyRemote:

**Module:** mod\_proxy

**Syntax:** ProxyRemote *match remote-server*

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule proxy_connect_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

```
LoadModule proxy_ftp_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

```
LoadModule proxy_http_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

```
LoadModule proxy_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

**Example:** ProxyRemote ftp http://ftpproxy.mydomain.com:8080

The ProxyRemote directive defines remote proxies for the local server. Multiple ProxyRemote directives are allowed. When a client sends a request that matches a ProxyRemote directive, the local server connects to the remote proxy server specified in the directive, rather than to the server specified in the URL. The remote proxy server retrieves the requested document and returns it to the local server, who in turn returns it to the client. This is referred to as a "proxy chain" since more than one proxy is used.

Proxy chains are useful in cases where multiple caches are used, or when the local server doesn't support the protocol (or schema) specified in the URL and must chain the request to a proxy that does support the protocol. Proxy chains may also be useful in cases where certain requests must be chained to another proxy server in order to get through a firewall or route across a virtual private network.

**Parameter One:** *match*

- The *match* parameter is either the name of a URL scheme that the remote proxy server supports, a partial URL that can be used to distinguish requests that should be chained from requests that need not be chained, or '\*' to indicate the remote proxy server should be contacted (or chained) for all requests.

**Parameter Two:** *remote-server*

- The *remote-server* parameter is a partial URL for the remote server.

**Syntax:** `<remote-server>=<protocol>://<hostname>[:port]`

Where *<protocol>* is the protocol that should be used to communicate with the remote server. Only HTTP is supported by this module.

#### Example 1

```
ProxyRemote ftp http://ftpproxy.server.com:8080
```

#### Example 2

```
ProxyRemote http://server.com/ http://mirrorserver.com:8000
```

#### Example 3

```
ProxyRemote * http://server.com
```

In example 1, the server will forward (or chain) all FTP requests, encapsulated as yet another HTTP proxy request, to the server named `ftpproxy.server.com` (port 8080), which then handles the request and returns the document to the local server.

In example 2, the server will forward all requests that match the partial URL `http://server.com/` to the server named `mirrorserver.com` (port 8000).

In example 3, all requests will be forwarded to the server named `server.com`.

- “ProxyBlock” on page 433 may be used to block incoming requests prior to consideration for this directive.
- Requests matching a “NoProxy” on page 430 directive are not chained.
- This directive is commonly used in conjunction with “NoProxy” on page 430 and “ProxyDomain” on page 435 for directing proxy requests within intranets.
- Setting ProxyNoConnect to on negates this directive.

#### ProxyRemoteMatch:

**Module:** `mod_proxy`

**Syntax:** `ProxyRemoteMatch regex remote-server`

**Default:** none

**Context:** server, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule proxy_connect_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

```
LoadModule proxy_ftp_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

```
LoadModule proxy_http_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

```
LoadModule proxy_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

**Example:** `ProxyRemote ftp http://ftpproxy.mydomain.com:8080`

The ProxyRemoteMatch is identical to the ProxyRemote directive, except the first argument is a regular expression match against the requested URL.

#### ProxyRequests:

**Module:** `mod_proxy`

**Syntax:** `ProxyRequests on | off`

**Default:** ProxyRequests off

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule proxy_connect_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_ftp_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_http_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
```

**Example:** ProxyRequests on

The ProxyRequest directive allows or prevents the server from functioning as a forward proxy.

**Parameter:** *on* | *off*

- If set to *off*, the server does not function as a forward proxy (see Example 1, below).
- If set to *on*, the server functions as a forward proxy and accepts proxy requests. All other directives for the mod\_proxy module are in effect.

**Example 1**

```
ProxyRequests off
```

**Example 2**

```
ProxyRequests on
CacheRoot /QOpenSys/UserData/HTTPA/CacheRoot
```

**Example 3**

```
ProxyRequests on
ProxyNoConnect on
CacheRoot /QOpenSys/UserData/HTTPA/CacheRoot
```

**Example 4**

```
ProxyRequests on
CacheExpiryCheck off
CacheRoot /QOpenSys/UserData/HTTPA/CacheRoot
```

- If CacheRoot is set, the proxy also activates its caching function and may serve documents from cache (by default) as well as issue direct outgoing requests (by default) (see Example 2, above). Expiry checking for cached documents is performed (by default).
- If CacheRoot is set and “ProxyNoConnect” on page 442 is set to *on*, the proxy activates its caching function but will only serve documents from cache. It will not issue outgoing requests (see Example 3, above). Expiry checking for cached documents is performed.
- If CacheRoot is set and “CacheExpiryCheck” on page 264 is set to *off*, the proxy activates its caching function but will not check expiry times for cached documents. Expired documents may be served from cache (see Example 4, above).
- Setting “ProxyRequests” on page 452 to *off* negates all directives for the mod\_proxy module, except for the “ProxyPass” on page 443 and “ProxyPassReverse” on page 447 directives.

**ProxyReverse:**

**Module:** mod\_proxy

**Syntax:** ProxyReverse *on* | *off*

**Default:** ProxyReverse on

**Context:** server config, virtual host

**Override:** none

**Origin:** IBM

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule proxy_connect_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_ftp_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_http_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
```

**Example:** ProxyReverse off

The ProxyReverse directive specifies whether the server may function as a reverse proxy to handle requests. The reverse proxy function is enabled by default, however other directives must specify how the server identifies and maps requests for reverse proxy. If off is specified, the reverse proxy function is disabled, and directives that apply the function (ProxyPass and RewriteRule directives using the 'proxy' flag) are ineffective. See "ProxyPass" on page 443 and "RewriteRule" on page 463 for more details on reverse proxy.

**Parameter:** *on* | *off*

- If set to *off*, the server does not function as a reverse proxy (see Example 1).
- If set to *on*, the server functions as a reverse proxy to handle requests identified and mapped for reverse proxy

#### Example 1

```
ProxyReverse off
ProxyPass /docs/ http://pubserver.ibm.com/public/documentation/
```

For example one, the reverse proxy function is disabled. The ProxyPass directive is ineffective.

In the following example, the reverse proxy function is enabled. The server functions as a reverse proxy to handle requests that match the path specified for the ProxyPass directive.

#### Example 2

```
ProxyReverse on
ProxyPass /docs/ http://pubserver.ibm.com/public/documentation/
```

See "ProxyPass" on page 443 for more details.

### ProxyTimeout:

**Module:** mod\_proxy

**Syntax:** ProxyTimeout *period*

**Default:** none (the general server timeout value is used)

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule proxy_connect_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_ftp_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_http_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

**Example:** ProxyTimeout 300

The ProxyTimeout directive specifies the maximum number of minutes the server will wait for responses from remote servers when handling proxy requests. If not specified, the general server timeout value is used (see the "TimeOut" on page 330 directive).

**Parameter:** *period*

- The *period* parameter accepts an integer value between 1 and 2,147,483,648 to specify the maximum period of time the server should wait for responses from remote servers (in seconds).

If a response is not received in the specified number of seconds, the server will cancel the request and return response code 504 (Gateway timeout).

## Example

ProxyTimeout 120

For this example, the server will wait up to 120 seconds (or 2 minute) for responses from remote servers.

- “ProxyPassReverse” on page 447 may be used to handle HTTP redirect responses from remote servers.
- Setting “ProxyReverse” on page 453 to *off* negates this directive.
- Setting “ProxyRequests” on page 452 to *off* does not negate this directive. It is available regardless of the forward proxy state.

## ProxyVia:

**Module:** mod\_proxy

**Syntax:** ProxyVia *off* | *on* | *full* | *block*

**Default:** ProxyVia *off*

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule proxy_connect_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
```

```
LoadModule proxy_ftp_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
```

```
LoadModule proxy_http_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
```

```
LoadModule proxy_module /QSYS.LIB/QHTTSPVR.LIB/QZSRCORE.SRVPGM
```

**Example:** ProxyVia *on*

The ProxyVia directive controls the server's use of the Via: HTTP header. Its intended use is to control the flow of proxy requests along a chain of servers. See RFC2068 (HTTP/1.1) for an explanation of Via: header lines.

**Parameter:** *off* | *on* | *full* | *block*

- If set to *off* (the default value), no special processing is performed. The proxy does not include a Via: header line, however any existing Via: headers from other proxy servers are kept intact.
- If set to *on*, each request and reply will get a Via: header line added for the current host. The proxy includes its own, abbreviated Via: header line. Any additional Via: header lines from other proxy servers are kept intact.
- If set to *full*, each generated Via: header line will additionally have HTTP Server version shown on the Via: comment field. The proxy includes its own, full Via: header (containing the proxy's version description). Any additional Via: header lines from other proxy servers are kept intact.
- If set to *block*, every proxy request will have all its Via: header lines removed. No new Via: header will be generated. The proxy does not include a Via: header and removes all existing Via headers from other proxy servers.

A server may be a participant in a proxy chain even though it is not specifically configured to chain its own requests. For this reason, it may be necessary to control the server's use of the Via: HTTP header even though it is not specifically configured for proxy chaining (see “ProxyRemote” on page 451 for more details about proxy chains).

## Module mod\_rewrite

Module mod\_rewrite supports directives for the IBM HTTP Server for i Web server.

## Summary

This module allows you to control URL access to your HTTP Server.

For example, to prevent a particular user agent called Web crawler from accessing any pages on the server. To do this, include the following directives in your configuration:

```
RewriteEngine on
RewriteCond %{HTTP_USER_AGENT} ^Webcrawler
RewriteRule ^.*$ - [F,L]
```

The first line enables the rewrite engine. The second line provides a test that returns true if the HTTP\_USER\_AGENT string starts with the letters Web crawler. If the second line is true, then the third line takes any URL string and returns a forbidden message to the client.

## Directives

- “RewriteBase”
- “RewriteCond” on page 457
- “RewriteEngine” on page 460
- “RewriteLog” on page 460
- “RewriteLogLevel” on page 461
- “RewriteMap” on page 461
- “RewriteOptions” on page 463
- “RewriteRule” on page 463

## RewriteBase:

**Module:** mod\_rewrite

**Syntax:** RewriteBase *Base\_URL*

**Default:** RewriteBase physical directory path

**Context:** Directory, but not Location, .htaccess

**Override:** FileInfo

**Origin:** Apache

**Example:** RewriteBase /xyz

The RewriteBase directive explicitly sets the base URL for per-directory rewrites. As you will see below, RewriteRule can be used in per-directory config files (.htaccess). There it will act locally (for example, the local directory prefix is stripped at this stage of processing and your rewriting rules act only on the remainder). At the end it is automatically added back to the path.

When a substitution occurs for a new URL, this module has to re-inject the URL into the processing server. To be able to do this it needs to know what the corresponding URL-prefix or URL-base is. By default this prefix is the corresponding filepath itself. At most, Web sites URLs are not directly related to physical filename paths, so this assumption is usually incorrect. In this case, you have to use the RewriteBase directive to specify the correct URL-prefix.

**Note:** If your webserver's URLs are not directly related to physical file paths, you have to use RewriteBase in every .htaccess file where you want to use RewriteRule directives.

Assume the following per-directory configuration file (/abc/def is the physical path of /xyz, and the server has the 'Alias /xyz /ABC/def' established).

```
RewriteEngine On
RewriteBase /xyz
RewriteRule ^old\.html$ new.html
```

In the above example, a request to /xyz/old.html is correctly rewritten to the physical file /ABC/def/new.html.

## RewriteCond:

**Module:** mod\_rewrite

**Syntax:** RewriteCond *TestString CondPattern [flags]*

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** FileInfo

**Origin:** Apache

**Example:** RewriteCond %{HTTP\_USER\_AGENT} ^Mozilla.\*

The RewriteCond directive defines a rule condition. Precede a RewriteRule directive with one or more RewriteCond directives. The following rewriting rule is only used if its pattern matches the current state of the URI and if these additional conditions apply.

### Parameter One: *TestString*

- The *TestString* parameter can contain the following expanded constructs in addition to plain text:
  - **RewriteRule backreferences:** These are backreferences of the form.  
**\$N** (0 <= N <= 9) that provide access to the grouped sections (those in parenthesis) of the pattern from the corresponding RewriteRule directive (the one following the current RewriteCond directives).
  - **RewriteCond backreferences:** These are backreferences of the form.  
**%N** (0 <= N <= 9) that provide access to the grouped sections (those in parenthesis) of the pattern from the last matched RewriteCond directive in the current conditions
  - **RewriteMap expansions:** These are expansions of the form.  
**\${mapname:key|default}**  
See "RewriteMap" on page 461 for more details.
  - **Server-Variables:** These are variables of the form  
**%{ NAME\_OF\_VARIABLE }**  
Where NAME\_OF\_VARIABLE can be a string taken from the following list:

#### HTTP headers

- HTTP\_USER\_AGENT
- HTTP\_REFERER
- HTTP\_COOKIE
- HTTP\_FORWARDED
- HTTP\_HOST
- HTTP\_PROXY\_CONNECTION
- HTTP\_ACCEPT

#### Connection and Request

- REMOTE\_ADDR
- REMOTE\_HOST
- REMOTE\_USER
- REMOTE\_IDENT
- REQUEST\_METHOD
- SCRIPT\_FILENAME
- PATH\_INFO
- QUERY\_STRING
- AUTH\_TYPE

### Server Internals

- DOCUMENT\_ROOT
- SERVER\_ADMIN
- SERVER\_NAME
- SERVER\_ADDR
- SERVER\_PORT
- SERVER\_PROTOCOL
- SERVER\_SOFTWARE

### System

- TIME\_YEAR
- TIME\_MON
- TIME\_DAY
- TIME\_HOUR
- TIME\_MIN
- TIME\_SEC
- TIME\_WDAY
- TIME

### Special

- API\_VERSION
- THE\_REQUEST
- REQUEST\_URI
- REQUEST\_FILENAME
- IS\_SUBREQ

### Tip:

1. The variables SCRIPT\_FILENAME and REQUEST\_FILENAME contain the same value (the value of the filename field of the internal request\_rec structure of the server). The first name is just the commonly known CGI variable name while the second is the consistent counterpart to REQUEST\_URI (which contains the value of the URI field of request\_rec).
2. There is the special format: %{ENV:variable} where variable can be any environment variable. This is looked-up via internal structures and (if not found there) via getenv() from the server process.
3. There is the special format: %{HTTP:header} where header can be any HTTP MIME-header name. This is looked-up from the HTTP request. Example: %{HTTP:Proxy-Connection} is the value of the HTTP header ``Proxy-Connection``.
4. There is the special format %{LA-U:variable} for look-aheads that perform an internal (URL-based) sub-request to determine the final value of variable. Use this when you want to use a variable for rewriting (which is actually set later in an API phase and thus is not available at the current stage). For instance when you want to rewrite according to the REMOTE\_USER variable from within the per-server context (httpd.conf file) you have to use %{LA-U:REMOTE\_USER} because this variable is set by the authorization phases that come after the URL translation phase where mod\_rewrite operates. On the other hand, because mod\_rewrite implements its per-directory context (.htaccess file) via the Fixup phase of the API and because the authorization phases come before this phase, you just can use %{REMOTE\_USER} there.



5. There is the special format: %[LA-F:variable] that performs an internal (filename-based) sub-request to determine the final value of variable. Most of the time this is the same as LA-U above.

**Parameter Two: *CondPattern***

- The *CondPattern* parameter is the condition pattern (a regular expression) that is applied to the current instance of the TestString. TestString is evaluated and then matched against CondPattern.

CondPattern is a standard Extended Regular Expression with some additions:

1. You can prefix the pattern string with a '!' character (exclamation mark) to specify a non-matching pattern.
2. There are some special CondPattern variants. Instead of real regular expression strings you can also use one of the following:

**<CondPattern**

Treats the CondPattern as a plain string and compares it lexically to TestString. True if TestString is lexically lower than CondPattern.

**>CondPattern**

Treats the CondPattern as a plain string and compares it lexically to TestString. True if TestString is lexically greater than CondPattern.

**=CondPattern**

Treats the CondPattern as a plain string and compares it lexically to TestString. True if TestString is lexically equal to CondPattern (the two strings are exactly equal, character by character). If CondPattern is just "" (two quotation marks) this compares TestString to the empty string.

- d Treats the TestString as a pathname and tests if it exists and is a directory.
- f Treats the TestString as a pathname and tests if it exists and is a regular file.
- s Treats the TestString as a pathname and tests if it exists and is a regular file with size greater than zero.
- l Treats the TestString as a pathname and tests if it exists and is a symbolic link.
- F Checks if TestString is a valid file and accessible via all the server's currently-configured access controls for that path. This uses an internal subrequest to determine the check.
- U Checks if TestString is a valid URL and accessible via all the server's currently-configured access controls for that path. This uses an internal subrequest to determine the check.

**Note:** All of these tests can also be prefixed by an exclamation mark (!) to negate their meaning.

**Parameter Three: *flags***

- The *flags* parameter is appended to the CondPattern parameter. The *flags* parameter is a comma-separated list of the following flags:

**nocase | NC**

This makes the test case-insensitive (there is no difference between 'A-Z' and AZ both in the expanded TestString and the CondPattern). This flag is effective only for comparisons between TestString and CondPattern. It has no effect on filesystem and subrequest checks.

**ornext | OR**

Use this to combine rule conditions with a local OR instead of the implicit AND. Typical example:

```

RewriteCond %{REMOTE_HOST} ^host1.* [OR]
RewriteCond %{REMOTE_HOST} ^host2.* [OR]
RewriteCond %{REMOTE_HOST} ^host3.*
RewriteRule ...some special stuff for any of these hosts...

```

Without this flag you would have to write the cond/rule three times.

To rewrite the Homepage of a site according to the ``User-Agent:" header of the request, you can use the following:

```

RewriteCond %{HTTP_USER_AGENT} ^Mozilla.*
RewriteRule ^/$ /homepage.max.html [L]

```

```

RewriteCond %{HTTP_USER_AGENT} ^Lynx.*
RewriteRule ^/$ /homepage.min.html [L]

```

```

RewriteRule ^/$ /homepage.std.html [L]

```

If you use Netscape Navigator as your browser (which identifies itself as 'Mozilla'), then you get the max homepage, which includes Frames and so on. If you use the Lynx browser (which is Terminal-based), then you get the min homepage, which contains no images, no tables, and so on. If you use any other browser you get the standard homepage.

### RewriteEngine:

**Module:** mod\_rewrite

**Syntax:** RewriteEngine *on* | *off*

**Default:** RewriteEngine off

**Context:** server config, virtual host, directory, .htaccess

**Override:** FileInfo

**Origin:** Apache

**Example:** RewriteEngine on

The RewriteEngine directive enables or disables the runtime rewriting engine. You can use this directive to disable the module instead of commenting out all the RewriteRule directives.

**Parameter:** *on* | *off*

- If set to *on* runtime processing is enabled. If it is set to *off* runtime processing is disabled and this module does not runtime processing at all.

**Note:** By default, rewrite configurations are not inherited. This means that you need to have the RewriteEngine on for each virtual host in which you want to use it.

### RewriteLog:

**Module:** mod\_rewrite

**Syntax:** RewriteLog *filename*

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Example:** RewriteLog "/usr/local/var/apache/logs/rewrite.log"

The RewriteLog directive sets the name of the file to which the server logs any rewriting actions it performs. The directive should occur only once per server configuration.

**Parameter One:** *filename*

- The *filename* parameter is any valid filename that QTMHHTTP has authority to write to. If the name does not begin with a slash ('/') then it is assumed to be relative to the Server Root. For example,

```
RewriteLog "/usr/local/var/apache/logs/rewrite.log"
```

**Note:** To disable the logging of rewriting actions it is not recommended to set *Filename* to `/dev/null`, because although the rewriting engine does not then output to a logfile, it still creates the logfile output internally. This will slow down the server with no advantage to the administrator. To disable logging either remove or comment out the `RewriteLog` directive or use `RewriteLogLevel 0!` in your configuration.

### RewriteLogLevel:

**Module:** `mod_rewrite`

**Syntax:** `RewriteLogLevel Level`

**Default:** `RewriteLogLevel 0`

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Example:** `RewriteLogLevel 3`

The `RewriteLogLevel` directive sets the level of the rewriting logfile.

**Parameter:** *Level*

- The *Level* parameter sets the level of the rewriting logfile. The default level 0 means no logging, while 9 means that practically all actions are logged. For example,

```
RewriteLogLevel 3
```

To disable the logging of rewriting actions simply set *Level* to 0. This disables all rewrite action logs.

**Note:** Using a high value for *Level* will slow down your server dramatically. Use the rewriting logfile at a *Level* greater than 2 only for debugging purposes.

### RewriteMap:

**Module:** `mod_rewrite`

**Syntax:** `RewriteMap MapName MapType:MapSource`

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Example:** `RewriteMap servers rnd:/path/to/file/map.txt`

The `RewriteMap` directive defines a Rewriting Map that can be used inside rule substitution strings by the mapping-functions to insert or substitute fields through a key lookup. The source of this lookup can be of various types.

**Parameter:** *MapName*

- The *MapName* parameter is the name of the map and is used to specify a mapping-function for the substitution strings of a rewriting rule via one of the following constructs:

```

${ MapName : LookupKey }
${ MapName : LookupKey | DefaultValue }

```

When such a construct occurs the map *MapName* is consulted and the key *LookupKey* is looked-up. If the key is found, the map-function construct is substituted by *SubstValue*. If the

key is not found then it is substituted by DefaultValue or by the empty string if no DefaultValue was specified. The following combinations for MapType and MapSource can be used:

### Standard Plain Text

MapType: txt, MapSource: Path to a file

This is the standard rewriting map feature where the MapSource is a plain text file containing either blank lines, comment lines (starting with a '#' character) or pairs like the following (one per line): MatchingKey SubstitutionValue.

#### File example:

```
##
## map.txt -- rewriting map
##
Ralf.B.Jones      rbj # Operator
Mr.Joe.Average   joe # Mr. Average
```

#### Directive example:

```
RewriteMap real-to-user txt:/path/to/file/map.txt
```

### Randomized Plain Text

MapType: rnd, MapSource: Path to a file

This is identical to the Standard Plain Text variant above but with a special post-processing feature. After looking up a value it is parsed according to the contained horizontal bar ( | ) characters which mean "or". In other words, the horizontal bars indicate a set of alternatives from which the actual returned value is randomly chosen. This feature was designed for load balancing in a reverse proxy situation where the looked up values are server names.

#### File example:

```
##
## map.txt -- rewriting map
##
static  www1|www2|www3|www4
dynamic www5|www6
```

#### Directive example:

```
RewriteMap servers rnd:/path/to/file/map.txt
```

### Internal Function

MapType: int, MapSource: Internal Apache function

The following internal functions are valid:

#### **toupper**

Converts the looked up key to all upper case.

#### **tolower**

Converts the looked up key to all lower case.

**escape** Translates special characters in the looked up key to hex-encodings.

#### **unescape**

Translates hex-encodings in the looked up key back to special characters.

The RewriteMap directive can occur more than once. For each mapping function use one RewriteMap directive to declare its rewriting mapfile. While you cannot declare a map in a per-directory context, it is possible to use this map in a per-directory context.

**Note:** The prg and dbm MapTypes are not supported.

## RewriteOptions:

**Module:** mod\_rewrite

**Syntax:** RewriteOptions *Option*

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** FileInfo

**Origin:** Apache

**Example:** RewriteOptions inherit

The RewriteOptions directive sets some special options for the current per-server or per-directory configuration.

**Parameter:** *Option*

- The *Option* parameter strings can be one of the following:

### **inherit**

This forces the current configuration to inherit the configuration of the parent. In per-virtual-server context this means that the maps, conditions and rules of the main server are inherited. In per-directory context this means that conditions and rules of the parent directory's .htaccess configuration are inherited.

### **MaxRedirects=*number***

This forces a request to terminate after reaching a maximum number of redirects and responds with a 500 Internal Server Error. The *number* parameter value is the maximum number of redirects allowed. This prevents endless loops of internal redirects issued by per-directory RewriteRules. If additional internal redirects are required, increase the default to the desired value. For example, RewriteOptions MaxRedirects=13.

## RewriteRule:

**Module:** mod\_rewrite

**Syntax:** RewriteRule *pattern substitution [flags]*

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** FileInfo

**Origin:** Apache

**Example:** RewriteRule ^/ABC(\*) /def\$1 [PT]

The RewriteRule directive is the real rewriting workhorse. The directive can occur more than once. Each directive then defines one single rewriting rule. The definition order of these rules is important, because this order is used when applying the rules at run-time.

**Parameter One:** *pattern*

- The *pattern* parameter can be an extended regular expression which gets applied to the current URL. Here "current" means the value of the URL when this rule gets applied. This may not be the originally requested URL, because any number of rules may already have matched and made alterations to it. See "Environment variables set by HTTP Server" on page 493 for more information.
- Additionally in mod\_rewrite the not character (!) is a possible pattern prefix. This gives you the ability to negate a pattern; to say, for instance: "if the current URL does not match this pattern". This can be used for exceptional cases, where it is easier to match the negative pattern, or as a last default rule.

**Note:** When using the not character to negate a pattern you cannot have grouped wild card parts in the pattern. This is impossible because when the pattern does not match, there are no contents for the groups, and you cannot use \$N in the substitution string.

**Parameter Two:** *substitution*

- The *substitution* parameter is the string which is substituted for (or replaces) the original URL for which Pattern matched. Beside plain text you can use back-references \$N to the RewriteRule pattern, back-references %N to the last matched RewriteCond pattern, server-variables as in rule condition test-strings (%{VARNAME}) and mapping-function calls (\${mapname:key | default}).
- Back-references are \$N (N=0..9) identifiers which will be replaced by the contents of the Nth group of the matched Pattern. The server-variables are the same as for the TestString of a RewriteCond directive. The mapping-functions come from the RewriteMap directive and are explained there. These three types of variables are expanded in the order of the above list. As already mentioned above, all the rewriting rules are applied to the Substitution (in the order of definition in the config file). The URL is completely replaced by the Substitution and the rewriting process goes on until there are no more rules unless explicitly terminated by a L flag - see below.
- There is a special substitution string named '-' which means: NO substitution. It is useful to provide rewriting rules which only match some URLs but do no substitution, for example, in conjunction with the C (chain) flag to be able to have more than one pattern to be applied before a substitution occurs.

**Note:** You can even create URLs in the substitution string containing a query string part. Just use a question mark inside the substitution string to indicate that the following stuff should be re-injected into the QUERY\_STRING. When you want to erase an existing query string, end the substitution string with just the question mark. There is a special feature: When you prefix a substitution field with http://thishost[:thisport] then mod\_rewrite automatically strips it out. This auto reduction on implicit external redirect URLs is a useful and important feature when used in combination with a mapping-function which generates the hostname part. Have a look at the first example in the example section below to understand this.

Remember, an unconditional external redirect to your own server will not work with the prefix http://thishost because of this feature. To achieve such a self-redirect, you have to use the R-flag (see below).

**Parameter Three:** *flags*

- The *flags* parameter can additionally be set to special [flags] for Substitution by appending [flags] as the third argument to the RewriteRule directive. Flags is a comma separated list of the following flags:

**redirect | R [=code]**

Prefix Substitution with http://thishost[:thisport]/ (which makes the new URL a URI) to force a external redirection. If no code is given an HTTP response of 302 (MOVED TEMPORARILY) is used. If you want to use other response codes in the range 300-400 just specify them as a number or use one of the following symbolic names: temp (default), permanent, seeother. Use it for rules which should canonicalize the URL and give it back to the client, for example, translate ``/~" into ``/u/" or always append a slash to /u/user, etc.

**Note:** When you use this flag, make sure that the substitution field is a valid URL. If not, you are redirecting to an invalid location! And remember that this flag itself only prefixes the URL with http://thishost[:thisport]/, rewriting continues. Usually you also want to stop and do the redirection immediately. To stop the rewriting you also have to provide the 'L' flag.

**forbidden | F**

This forces the current URL to be forbidden, for example, it immediately sends back an HTTP response of 403 (FORBIDDEN). Use this flag in conjunction with appropriate RewriteConds to conditionally block some URLs.

**gone | G**

This forces the current URL to be gone, for example, it immediately sends back an HTTP response of 410 (GONE). Use this flag to mark pages which no longer exist as gone.

**proxy | P**

This flag forces the substitution part to be internally forced as a proxy request and immediately (for example, rewriting rule processing stops here) put through the proxy module. You have to make sure that the substitution string is a valid URI (for example, typically starting with http://hostname) which can be handled by HTTP Server proxy module. If not you get an error from the proxy module. Use this flag to achieve a more powerful implementation of the ProxyPass directive, to map some remote stuff into the name space of the local server.

**Note:** To use this functionality make sure you have the proxy module loaded into your HTTP Server configuration (for example, via LoadModule directive).

**last | L** Stop the rewriting process here and don't apply any more rewriting rules. (This corresponds to the Perl last command or the break command from the C language.) Use this flag to prevent the currently rewritten URL from being rewritten further by following rules. For example, use it to rewrite the rootpath URL ('/') to a real one, for example, '/e/www/'.

**next | N**

Re-run the rewriting process (starting again with the first rewriting rule). Here the URL to match is again not the original URL but the URL from the last rewriting rule. (This corresponds to the Perl next command or the continue command from the C language.) Use this flag to restart the rewriting process, for example, to immediately go to the top of the loop. But be careful not to create an infinite loop.

**chain | C**

This flag chains the current rule with the next rule (which itself can be chained with the following rule, etc.). This has the following effect: if a rule matches, then processing continues as usual, for example, the flag has no effect. If the rule does not match, then all following chained rules are skipped. For instance, use it to remove the ``.www" part inside a per-directory rule set when you let an external redirect happen (where the ``.www " part should not occur).

**type | T=MIME-type**

Force the MIME-type of the target file to be MIME-type. For instance, this can be used to simulate the mod\_alias directive ScriptAlias which internally forces all files inside the mapped directory to have a MIME type of ``application/x-httpd-cgi".

**nosubreq | NS**

This flag forces the rewriting engine to skip a rewriting rule if the current request is an internal sub-request. For instance, sub-requests occur internally in HTTP Server when mod\_include tries to find out information about possible directory default files (index.xxx). On sub-requests it is not always useful and even sometimes causes a failure if the complete set of rules are applied. Use this flag to exclude some rules. Whenever you prefix some URLs with CGI-scripts to force them to be processed by the CGI-script, the chance is high that you will run into problems (or even overhead) on sub-requests. In these cases, use this flag.

**nocase|NC**

This makes the Pattern case insensitive, for example, there is no difference between AZ and az when Pattern is matched against the current URL.

**noescape|NE**

This flag prevents `mod_rewrite` from applying the usual URI escaping rules to the result of a rewrite. Ordinarily, special characters (`%`, `$`, `'`, `;`) will be escaped into their hexcode equivalents (`%25`, `%24`, and `%3B`, respectively); this flag prevents this from happening. This flag allows percent symbols to appear in the output, as in RewriteRule `/foo/(.*) /bar?arg=P1\%3d$1 [R,NE]` which would turn `'/foo/zed'` into a safe request for `'/bar?arg=P1=zed'`.

**qsappend|QSA**

This flag forces the rewriting engine to append a query string part in the substitution string to the existing one instead of replacing it. Use this when you want to add more data to the query string via a rewrite rule.

**passthrough|PT**

This flag forces the rewriting engine to set the URI field of the internal request\_rec structure to the value of the filename field. This flag is used to be able to post-process the output of RewriteRule directives by Alias, ScriptAlias, Redirect, etc. - directives from other URI-to-filename translators. A trivial example to show the semantics: If you want to rewrite `/ABC` to `/def` via the rewriting engine of `mod_rewrite` and then `/def` to `/ghi` with `mod_alias`:

```
RewriteRule ^/ABC(.*) /def$1 [PT]
Alias /def /ghi
```

If you omit the PT flag then `mod_rewrite` will do its job fine, for example, it rewrites `uri=/ABC/...` to `filename=/def/...` as a full API-compliant URI-to-filename translator should do. Then `mod_alias` comes and tries to do a URI-to-filename transition which will not work.

**Note:** You have to use this flag if you want to intermix directives of different modules which contain URL-to-filename translators. The typical example is the use of `mod_alias` and `mod_rewrite`.

**skip|S=num**

This flag forces the rewriting engine to skip the next num rules in sequence when the current rule matches. Use this to make pseudo if-then-else constructs: The last rule of the then-clause becomes `skip=N` where N is the number of rules in the else-clause. (This is not the same as the `'chain|C'` flag.)

**env|E=VAR:VAL**

This forces an environment variable named VAR to be set to the value VAL, where VAL can contain regexp backreferences \$N and %N which will be expanded. You can use this flag more than once to set more than one variable. The variables can be later dereferenced in many situations, but usually from within SSI (via `<!--#echo var="VAR"-->`) or CGI (for example `$ENV{VAR}`). Additionally you can dereference it in a following RewriteCond pattern via  `%{ENV:VAR}`. Use this to strip but remember information from URLs.

**Note:** Never forget that Pattern is applied to a complete URL in per-server configuration files. But in per-directory configuration files, the per-directory prefix (which always is the same for a specific directory!) is automatically removed for the pattern matching and automatically added after the substitution has been done. This feature is essential for many sorts of rewriting, because without this prefix stripping you have to match the parent directory which is not always possible. There is one exception: If a substitution string starts with ```http://``` then the directory prefix will not be added and an external redirect or proxy throughput (if flag P is used) is forced. To enable the



rewriting engine for per-directory configuration files you need to set RewriteEngine On in these files and Option FollowSymLinks must be enabled. If the override of FollowSymLinks is disabled for a user's directory, then you cannot use the rewriting engine. This restriction is needed for security reasons.

### Possible substitution combinations and meanings:

Given rule	Resulting substitution
^/somepath(*) otherpath\$1	not supported
^/somepath(*) otherpath\$1 [R]	not supported
^/somepath(*) otherpath\$1 [P]	not supported
^/somepath(*) /otherpath\$1	/otherpath/pathinfo
^/somepath(*) /otherpath\$1 [R]	http://thishost/otherpath/pathinfo via external redirection
^/somepath(*) /otherpath\$1 [P]	not supported
^/somepath(*) http://thishost/otherpath\$1	/otherpath/pathinfo
^/somepath(*) http://thishost/otherpath\$1 [R]	http://thishost/otherpath/pathinfo via external redirection
^/somepath(*) http://thishost/otherpath\$1 [P]	not supported
^/somepath(*) http://otherhost/otherpath\$1	http://otherhost/otherpath/pathinfo via external redirection
^/somepath(*) http://otherhost/otherpath\$1 [R]	^/somepath(*) http://otherhost/otherpath\$1 [R]
^/somepath(*) http://otherhost/otherpath\$1 [P]	http://otherhost/otherpath/pathinfo via internal proxy
^localpath(*) otherpath\$1	/somepath/otherpath/pathinfo
^localpath(*) otherpath\$1 [R]	http://thishost/somepath/otherpath/pathinfo via external redirection
^localpath(*) otherpath\$1 [P]	not supported
^localpath(*) /otherpath\$1	/otherpath/pathinfo
^localpath(*) /otherpath\$1 [R]	http://thishost/otherpath/pathinfo via external redirection
^localpath(*) /otherpath\$1 [P]	not supported
^localpath(*) http://thishost/otherpath\$1	/otherpath/pathinfo
^localpath(*) http://thishost/otherpath\$1 [R]	http://thishost/otherpath/pathinfo via external redirection
^localpath(*) http://thishost/otherpath\$1 [P]	not supported
^localpath(*) http://otherhost/otherpath\$1	http://otherhost/otherpath/pathinfo via external redirection
^localpath(*) http://otherhost/otherpath\$1 [R]	http://otherhost/otherpath/pathinfo via external redirection (the [R] flag is redundant)
^localpath(*) http://otherhost/otherpath\$1 [P]	http://otherhost/otherpath/pathinfo via internal proxy

If you wanted to rewrite URLs of the form / Language /~ Realname /.../ File into /u/ Username /.../ File . Language, you would take the rewrite mapfile from above and save it under /path/to/file/map.txt. Then we only have to add the following lines to HTTP Server configuration file:

```
RewriteLog /path/to/file/rewrite.log
RewriteMap real-to-user txt:/path/to/file/map.txt
RewriteRule ^/([\^/]+)/~([\^/]+)/(.*)$ /u/${real-to-user:$2|nobody}/${3.$1}
```

## Module mod\_setenvif

Module mod\_setenvif supports directives for the IBM HTTP Server for i Web server.

### Summary

The `mod_setenvif` module allows you to set environment variables if different aspects of the request match regular expressions that you specify. These variables can be used by other parts of the server to make decisions about actions to be taken.

The directives are considered in the order they appear in the configuration. So more complex sequences can be used, such as this example, which sets Netscape if the browser is Mozilla but not MSIE.

```
BrowserMatch ^Mozilla netscape
BrowserMatch MSIE !netscape
```

### Directives

- “`BrowserMatch`”
- “`BrowserMatchNoCase`” on page 469
- “`SetEnvIf`” on page 469
- “`SetEnvIfNoCase`” on page 471

### BrowserMatch:

**Module:** `mod_setenvif`

**Syntax:** `BrowserMatch regex envvar[=value] [...]`

**Default:** none

**Context:** server config, virtual host, directory, `.htaccess`

**Override:** none

**Origin:** Apache

**Example:** `BrowserMatch ^Mozilla forms jpeg=yes browser=netscape`

`BrowserMatch` defines environment variables based on the User-Agent HTTP request header field. The first argument should be a POSIX.2 extended regular expression (similar to an `egrep`-style regex). The rest of the arguments give the names of variables to set, and optional values to which they should be set. These take the form of the following:

- `varname`
- `!varname`
- `varname=value`

See “Environment variables set by HTTP Server” on page 493 for more information.

In the first form, the value will be set to “1”. The second will remove the given variable if already defined, and the third will set the variable to the value given by `value`. If a User-Agent string matches more than one entry, they will be merged. Entries are processed in the order in which they appear, and later entries can override earlier ones. For example:

```
BrowserMatch ^Mozilla forms jpeg=yes browser=netscape
BrowserMatch ^Mozilla/[2-3]" tables agif frames javascript
BrowserMatch MSIE !javascript
```

In the above example, if the User-Agent field is Mozilla, the environment variables `forms`, `jpeg=yes` and `browser=netscape` will be set. If the request is Mozilla/2 or Mozilla/3, then in addition to the environment variables on the first `BrowserMatch` directive, the variables `tables`, `agif`, `frames` and `javascript` will be set.

**Note:** The regular expression string is case-sensitive. For case-insensitive matching, see “`BrowserMatchNoCase`” on page 469. `BrowserMatch` and “`BrowserMatchNoCase`” on page 469 are special cases of “`SetEnvIf`” on page 469 and “`SetEnvIfNoCase`” on page 471. The following two lines have the same effect:

```
BrowserMatchNoCase Robot is_a_robot
SetEnvIfNoCase User-Agent Robot is_a_robot
```

**Parameter One:** *regex*

- The *regex* parameter is a case-sensitive POSIX.2 extended regular expression. This gives the user the ability to select variants on the User-Agent field, such as using some wildcarding to group versions of a client browser. See “Environment variables set by HTTP Server” on page 493 for more information.

**Parameter Two:** *envvar[=value]*

- The *envvar[=value]* parameter gives the names of the variables to set and, optional, values to which they should be set. The case is preserved when lowercase characters are specified. Valid values include all EBCDIC characters. The value must be enclosed in quotation marks if it contains any non-alphanumeric character or blanks.

**BrowserMatchNoCase:**

**Module:** mod\_setenvif

**Syntax:** BrowserMatchNoCase *regex envvar[=value] [...]*

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** none

**Origin:** Apache

**Example:** BrowserMatchNoCase ibm platform=ibm

BrowserMatchNoCase is semantically identical to “BrowserMatch” on page 468. However, it provides for case-insensitive matching. For example:

```
BrowserMatchNoCase mac platform=ibm
BrowserMatchNoCase win platform=windows
```

“BrowserMatch” on page 468 and BrowserMatchNoCase are special cases of “SetEnvIf” and “SetEnvIfNoCase” on page 471. The following two lines have the same effect:

```
BrowserMatchNoCase Robot is_a_robot
SetEnvIfNoCase User-Agent Robot is_a_robot
```

**Parameter One:** *regex*

- The *regex* parameter is a case-insensitive POSIX.2 extended regular expression. This gives the user the ability to select variants on the User-Agent field, such as using some wildcarding to group version of a client browser. See “Environment variables set by HTTP Server” on page 493 for more information.

**Parameter Two:** *envvar[=value]*

- The *envvar[=value]* parameter gives the names of variables to set and, optionally, values to which they should be set. They can take the form of 'varname', '!varname' or 'varname=value'. The case is preserved when lowercase characters are specified. Valid values include all EBCDIC characters. The value must be enclosed in quotation marks if it contains any non-alphanumeric character or blanks.

**SetEnvIf:**

**Module:** mod\_setenvif

**Syntax:** SetEnvIf *attribute regex envvar[=value] [...]*

**Default:** none



**Context:** server config, virtual host, directory, .htaccess

**Override:** none

**Origin:** Apache

**Example:** SetEnvIf Request\_URI "\.gif\$" object\_is\_image=gif

SetEnvIf defines environment variables based on attributes of the request. These attributes can be the values of various HTTP request header fields or of other aspects of the request. See RFC2068 for more information.

**Note:** To view the RFC listed above, visit the RFC index search engine  located on the RFC editor  web site. Search for the RFC number you want to view. The search engine results display the corresponding RFC title, author, date, and status.

Attribute	Description
Remote_Host	The hostname (if available) of the client making the request.
Remote_Addr	The IP address of the client making the request.
Remote_User	The authenticated username (if available).
Request_Method	The name of the method being used (GET, POST).
Request_Protocol	The name of the method being used (GET, POST).
Request_URI	The portion of the URL following the scheme and host portion.

Some of the more commonly used request header field names include Host, User-Agent, and Referrer.

If the attribute name does not match any of the special keywords, or any of the request's header field names, it is tested as the name of an environment variable in the list of those associated with the request. This allows SetEnvIf directives to test against the result of prior matches.

Only those environment variables defined by earlier SetEnvIf[NoCase] directives are available for testing in this manner. Earlier means that they were defined in a broader context (such as server-wide) or previously in the current directive's context. For example:

```
SetEnvIf Request_URI "\.gif$" object_is_image=gif
SetEnvIf Request_URI "\.jpg$" object_is_image=jpg
SetEnvIf Request_URI "\.xpm$" object_is_image=xpm
:
SetEnvIf Referrer www\.mydomain\.com intra_site_referral
:
SetEnvIf object_is_image xpm XBIT_PROCESSING=1
```

The first three will set the environment variable object\_is\_image if the request was for an image file, and the fourth sets intra\_site\_referral if the referring page was somewhere on the www.mydomain.com Web site. The 5th statement of the example sets XBIT processing, if the environment variable object\_is\_image was set by the directive.

**Parameter One:** *attribute*

- The *attribute* parameter is the attribute of the request, such as an HTTP header value. The attribute can also be an environment variable that was set by an earlier SETENVIF directive.

**Parameter Two:** *regex*

- The *regex* parameter is a case-sensitive POSIX.2 extended regular expression. This gives the user the ability to select variants on the Attribute field, such as using some wildcarding to group related values, and use those to set the environment variables. See “Environment variables set by HTTP Server” on page 493 for more information.

**Parameter Three:** *envvar[=value]*

- The *envvar[=value]* gives the names of variables to set and, optionally, values to which they should be set. They take the form of 'varname', '!varname' or 'varname=value'. The case is preserved when lowercase characters are specified. Valid values include all EBCDIC characters. The value must be enclosed in quotation marks if it contains any non-alphanumeric character or blanks.

## SetEnvIfNoCase:

**Module:** mod\_setenvif

**Syntax:** SetEnvIfNoCase *attribute regex envvar[=value] [...]*

**Default:** none

**Context:** server config, virtual host, directory .htaccess

**Override:** none

**Origin:** Apache

**Example:** SetEnvIfNoCase Host IBM\.Org site=ibm

SetEnvIfNoCase is semantically identical to “SetEnvIf” on page 469, and differs only in that the regular expression matching is performed in a case-insensitive manner. For example:

```
SetEnvIfNoCase Host QIBM\.Org site=ibm
```

This will cause the site variable to be set to 'ibm' if the HTTP request header field Host: was included and contained QIBM.Org, qibm.org, or any other combination.

### Parameter One: *attribute*

- The *attribute* parameter is the attribute of the request, such as an HTTP Header value. The attribute can also be an environment variable that was set by an earlier setenvif directive.

### Parameter Two: *regex*

- The *regex* parameter is a case-sensitive POSIX.2 extended regular expression. This gives the user the ability to select variants on the Attribute field, such as using some wildcarding to group related values, and use those to set the environment variables. See “Environment variables set by HTTP Server” on page 493 for more information.

### Parameter Three: *envvar[=value]*

- The *envvar[=value]* parameter gives the names of variables to set and, optionally, values to which they should be set. They take the form of 'varname', '!varname' or 'varname=value'. The case is preserved when lowercase characters are specified. Valid values include all EBCDIC characters. The value must be enclosed in quotation marks if it contains any non-alphanumeric character or blanks.

## Module mod\_so

Module mod\_so supports directives for the IBM HTTP Server for i Web server.

### Summary

The module mod\_so provides for loading of executable code and modules into the HTTP Server at startup or restart time. On the IBM i server, the loaded code comes from a service program object with a .SRVPGM extension.

### Directive

- “LoadModule”

### LoadModule:

**Module:** mod\_so

**Syntax:** LoadModule *module filename*

**Default:** none

**Context:** server config

**Override:** none

**Origin:** Apache

**Example:** LoadModule ibm\_ldap\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVLDAP.SRVPGM

The LoadModule directive links in the object file filename and adds the module structure named module to the list of active modules.

**Parameter One:** *module*

- The *module* parameter is the name of the external variable of type module is the IBM i file.

**Parameter Two:** *filename*

- The *filename* parameter must be an IBM i service program.

The following example loads ibm\_ldap in QZSRVLDAP service program into the current HTTP Server:

```
LoadModule ibm_ldap_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVLDAP.SRVPGM
```

## Module mod\_userdir

Module mod\_userdir supports directives for the IBM HTTP Server for i Web server.

### Summary

The module mod\_userdir provides for automatic directory indexing.

### Directive

- "UserDir"

#### UserDir:

**Module:** mod\_userdir

**Syntax:** UserDir *directory* [*directory* ...] | *enabled* *username* [*username* ...] | *disabled* [*username*...]

**Default:** UserDir public\_html

**Context:** server config, virtual host

**Override:** mod\_userdir

**Origin:** Apache

**Example:** UserDir WWW

**Example:** UserDir enable lewis thomas

**Example:** UserDir disable sherman fazio

**Example:** UserDir disable

The UserDir directive sets the real directory in a user's home directory to use when a request for a document for a user is received. The user's home directory is based on the HOMEDIR setting of the user's IBM i user profile. Directory is one of the following:

**Parameter:** *directory*

- The name of a directory or a pattern such as those shown below.
- The keyword *disabled*. This turns off all username-to-directory translations except those explicitly named with the enabled keyword (see below).
- The keyword *disabled* followed by a space-delimited list of usernames. Usernames that appear in such a list will never have directory translation performed, even if they appear in an enabled clause.
- The keyword *enabled* followed by a space-delimited list of usernames. These usernames will have directory translation performed even if a global disable is in effect, but not if they also appear in a disabled clause. Note: the keyword enabled without a list of usernames is not valid.

**Note:** The UserDir directive is not inherited by virtual hosts when it is set in the global server configuration.

If neither the enabled nor the disabled keywords appear in the UserDir directive, the argument is treated as a filename pattern (or list of filename patterns), and is used to turn the name into a directory specification. For example, assume that the HOMEDIR parameter of the IBM i user profile "bob" is set to /home/bob. A request for `http://www.QIBM.com/~bob/one/two.html` will be translated to:

```
UserDir public_html -> /home/bob/public_html/one/two.html
UserDir /usr/web -> /usr/web/bob/one/two.html
UserDir /home/*/www -> /home/bob/www/one/two.html
```

The following directives will send redirects to the client:

```
UserDir http://www.QIBM.com/users -> http://www.QIBM.com/users/home/bob/one/two.html
UserDir http://www.QIBM.com/*/usr -> http://www.QIBM.com/home/bob/usr/one/two.html
```

**Note:** Use caution when using this directive; for instance, "UserDir ./" would map "/~root" to "/" - which is most likely undesirable. It is strongly recommended that your configuration include a "UserDir disabled root" declaration. If multiple UserDir directives without disable or enable keywords occur in a configuration, the last one is used.

See <Directory> and "Security tips for HTTP Server" on page 29 for more information.

## Module mod\_usertrack

Module mod\_usertrack supports directives for the IBM HTTP Server for i Web server.

### Summary

This module provides support for tracking users through the use of cookies.

**Note:** Netscape 4.x (Communicator) and above can use two or four digit dates. Netscape 3.x and below will only accept two digit dates. To ensure the expiration date is legible to the client's browser use two digit dates.

### Directives

- "CookieDomain"
- "CookieExpires" on page 474
- "CookieName" on page 474
- "CookieStyle" on page 474
- "CookieTracking" on page 475

### CookieDomain:

**Module:** mod\_usertrack

**Syntax:** CookieDomain *domain*

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** none

**Origin:** Apache

**Example:** CookieDomain .mydomain.com

The CookieDomain directive controls the setting of the domain to which the tracking cookie applies. If not present, no domain is included in the cookie header field. The domain string must begin with a dot, and must include at least one embedded dot. That is, .ibm.com is legal, but ibm.com and .com are not.

**Parameter:** *domain*

- A *domain* is a partially qualified DNS domain name, preceded by a period. It represents a group of hosts that logically belong to the same DNS domain or zone (that is, the suffixes of the hostnames are all ending in Domain).

## CookieExpires:

**Module:** mod\_usertrack

**Syntax:** CookieExpires *expiry-period*

**Default:** none

**Context:** server config, virtual host, directory, .htaccess

**Override:** none

**Origin:** Apache

**Example:** CookieExpires 120

The CookieExpires directive sets an expiry time on the cookie generated by the usertrack module. If this directive is not used, cookies last only for the current browser session.

**Parameter:** *expiry-period*

- The *expiry-period* specifies the time, in seconds, the cookie should remain.

## CookieName:

**Module:** mod\_usertrack

**Syntax:** CookieName *token*

**Default:** CookieName Apache

**Context:** server config, virtual host, directory, .htaccess

**Override:** FileInfo

**Origin:** Apache

**Example:** CookieName ABCDE19

The CookieName directive allows you to change the name of the cookie. The cookie name is used for tracking purposes. You must specify a valid cookie name; results are unpredictable if you use a name containing unusual characters. Valid characters include A-Z, a-z, 0-9, '\_' and '-'.

**Parameter:** *token*

- The *token* parameter allows you to change the name of the cookie.

## CookieStyle:

**Module:** mod\_usertrack

**Syntax:** CookieStyle *Netscape* | *Cookie* | *Cookie2* | *RFC2109* | *RFC2965*

**Default:** CookieStyle Netscape

**Context:** server config, virtual host, directory, .htaccess

**Override:** none

**Origin:** Apache

**Example:** CookieStyle Cookie

This CookieStyle directive controls the format of the cookie header field.

**Parameter:** *Netscape* | *Cookie* | *Cookie2* | *RFC2109* | *RFC2965*

- *Netscape* is the original, but now deprecated, syntax. This is the default, and the syntax HTTP Server has historically used.
- *Cookie* or *RFC2109* is the syntax that superseded the *Netscape* syntax.
- *Cookie2* or *RFC2965* is the most current cookie syntax.

**Note:** Not all clients can understand all of these formats. You should use the most current one that is generally acceptable to your users' browsers.



## CookieTracking:

**Module:** mod\_usertrack

**Syntax:** CookieTracking *on* | *off*

**Default:** Compiling mod\_usertrack will not activate cookies by default.

**Context:** server config, virtual host, directory, .htaccess

**Override:** FileInfo

**Origin:** Apache

**Example:** CookieTracking on

The CookieTracking directive allows you to send a user-tracking cookie for all new requests. This directive can be used to turn this behavior on or off on a per-server or per-directory basis.

**Parameter:** *on* | *off*

- With CookieTracking *on*, the server starts sending a user-tracking cookie for all new requests.
- With CookieTracking *off*, the server does not send a user-tracking cookie for all new requests.

## Module mod\_vhost\_alias

Module mod\_vhost\_alias supports directives for the IBM HTTP Server for i Web server.

### Summary

The module mod\_vhost\_alias provides support for dynamically configured mass virtual hosting.

### Virtual hosting

The term Virtual Host refers to the practice of maintaining more than one server on one machine or server instance, as differentiated by their apparent hostname (or server instance name). For example, it is often desirable for companies sharing a web server to have their own domains, with web servers accessible as `www.company1.com` and `www.company2.com`, without requiring the user to know extra path information.

HTTP Server supports two types of virtual hosting, they are IP-based Virtual Host and Name-based Virtual Host. As the term IP-based indicates, the server must have a different IP address for each IP-based virtual host. This can be achieved by the machine having several physical network connections, or by use of virtual interfaces that are supported by most modern operating systems.

While the approach with IP-based Virtual Hosts works well, it is not the most elegant solution, because a dedicated IP address is needed for every virtual host and is hard to implement on some machines. The HTTP/1.1 protocol contains a method for the server to identify what name it is being addressed as.

The benefits of using the name-based virtual host support is a practically unlimited number of servers, ease of configuration and use, and no additional hardware or software requirements. The main disadvantage is that the client must support this part of the protocol. The latest versions of most browsers (e.g. HTTP 1.1) do, but there are still old browsers (e.g. HTTP 1.0) in use that do not. This can cause problems, although a possible solution is addressed below.

### Using non-IP virtual hosts

The notable difference between IP-based and name-based virtual host configuration is the NameVirtualHost directive that specifies an IP address that should be used as a target for name-based virtual hosts. For example, suppose that both `www.domain.tld` and `www.otherdomain.tld` point at the IP address `111.22.33.44`. Simply add to one of the configuration files (most likely `httpd.conf` or `srm.conf`) code similar to the following:

```
NameVirtualHost 111.22.33.44
```

```
<VirtualHost 111.22.33.44>  
ServerName www.domain.tld  
DocumentRoot /www/domain  
</VirtualHost>
```

```
<VirtualHost 111.22.33.44>  
ServerName www.otherdomain.tld  
DocumentRoot /www/otherdomain  
</VirtualHost>
```

Of course, any additional directives can (and should) be placed into the <VirtualHost> section. To make this work, make sure that the names www.domain.tld and www.otherdomain.tld are pointing to the IP address 111.22.33.44

**Note:** When you specify an IP address in a NameVirtualHost directive, requests to that IP address are only served by matching <VirtualHost>s. The main server is never served from the specified IP address. If you start to use virtual hosts you should stop using the main server as an independent server and use it as a place for configuration directives that are common for all your virtual hosts. In other words, you should add a <VirtualHost> section for every server (hostname) you want to maintain on your server.

Additionally, many servers may want to be accessible by more than one name. For example, the example server might want to be accessible as domain.tld, or www2.domain.tld, assuming the IP addresses pointed to the same server. In fact, one might want it so that all addresses at domain.tld were picked up by the server. This is possible with the ServerAlias directive, placed inside the <VirtualHost> section. For example:

```
ServerAlias domain.tld *.domain.tld
```

**Note:** You can use \* and ? as wild-card characters.

You might also need ServerAlias if you are serving local users who do not always include the domain name. For example, if local users are familiar with typing "www" or "www.example" then you will need to add ServerAlias www www.example. It isn't possible for the server to know what domain the client uses for their name resolution because the client doesn't provide that information in the request. The ServerAlias directive provides a means for different hostnames to point to the same virtual host.

## Dynamic virtual hosting

A virtual host is defined by two pieces of information: its IP address, and the contents of the Host: header in the HTTP request. The dynamic mass virtual hosting technique is based on automatically inserting this information into the pathname of the file that is used to satisfy the request. This is done most easily using mod\_vhost\_alias.

A couple of things need to be 'faked', that being specific parameters with incorrect parameter values, to make the dynamic virtual host look like a normal one. The most important is the server name which is used by HTTP Server to generate a self referencing URLs. It is configured with the ServerName directive, and it is available to CGIs via the SERVER\_NAME environment variable. The actual value used at run time is controlled by the UseCanonicalName setting. With UseCanonicalName off the server name comes from the contents of the Host: header in the request. With UseCanonicalName DNS it comes from a reverse DNS lookup of the virtual host's IP address. The former setting is used for name-based dynamic virtual hosting, and the latter is used for IP-based hosting. If HTTP Server cannot work out the server name because there is no Host: header or the DNS lookup fails then the value configured with ServerName is used instead.

The other thing to 'fake' is the document root (configured with DocumentRoot and available to CGIs via the DOCUMENT\_ROOT environment variable). This setting is used by the core module when mapping

URIs to filenames, but when the server is configured to do dynamic virtual hosting that job is taken over by the `mod_vhost_alias` module. If any CGIs or SSI documents make use of the `DOCUMENT_ROOT` environment variable they will therefore get a misleading value; there is not any way to change `DOCUMENT_ROOT` dynamically.

### Motivation for dynamic virtual hosting

The techniques described here are of interest if your `httpd.conf` contains many `<VirtualHost>` sections that are substantially the same. For example:

```
NameVirtualHost 10.22.33.44
<VirtualHost 10.22.33.44>
ServerName www.customer-1.com
DocumentRoot /www/hosts/www.customer-1.com/docs
ScriptAlias /cgi-bin/ /www/hosts/www.customer-1.com/cgi-bin
</VirtualHost>
<VirtualHost 10.22.33.44>
ServerName www.customer-2.com
DocumentRoot /www/hosts/www.customer-2.com/docs
ScriptAlias /cgi-bin/ /www/hosts/www.customer-2.com/cgi-bin
</VirtualHost>
# comment line
<VirtualHost 10.22.33.44>
ServerName www.customer-N.com
DocumentRoot /www/hosts/www.customer-N.com/docs
ScriptAlias /cgi-bin/ /www/hosts/www.customer-N.com/cgi-bin
</VirtualHost>
```

The basic idea is to replace all of the static `<VirtualHost>` configuration with a mechanism that works it out dynamically. This has a number of advantages:

1. Your configuration file is smaller so HTTP Server starts faster and uses less memory.
2. Adding virtual hosts is simply a matter of creating the appropriate directories in the filesystem and entries in the DNS - you do not need to configure or restart HTTP Server.

The main disadvantage is that you cannot have a different log file for each virtual host; however if you have very many virtual hosts then doing this is dubious anyway because it eats file descriptors. It is better to log to a pipe or a fifo and arrange for the process at the other end to distribute the logs to the customers (it can also accumulate statistics).

A request for `http://www.example.isp.com/directory/file.html` will be satisfied by the file:

```
/usr/local/apache/vhosts/isp.com/e/x/a/example/directory/file.html.
```

A more even spread of files can be achieved by hashing from the end of the name, for example:

```
VirtualDocumentRoot /usr/local/apache/vhosts/%3+/%2.-1/%2.-2/%2.-3/%2
```

The example request would come from `/usr/local/apache/vhosts/isp.com/e/1/p/example/directory/file.html`. Alternatively you might use:

```
VirtualDocumentRoot /usr/local/apache/vhosts/%3+/%2.1/%2.2/%2.3/%2.4+
```

The example request would come from `/usr/local/apache/vhosts/isp.com/e/x/a/mple/directory/file.html`.

### Simple dynamic virtual hosts

This extract from `httpd.conf` implements the virtual host arrangement outlined in the Motivation section above, but in a generic fashion using `mod_vhost_alias`.

```
# get the server name from the Host: header
UseCanonicalName off
# this log format can be split per-virtual-host based on the first field
```

```

LogFormat "%V %h %l %u %t \"%r\" %s %b" vcommon
CustomLog logs/access_log vcommon
# include the server name in the filenames used to satisfy requests
VirtualDocumentRoot /www/hosts/%0/docs
VirtualScriptAlias /www/hosts/%0/cgi-bin

```

This configuration can be changed into an IP-based virtual hosting solution by just turning `UseCanonicalName` off into `UseCanonicalName` DNS. The server name that is inserted into the filename is then derived from the IP address of the virtual host.

### A virtually hosted homepages system

This is an adjustment of the above system tailored for an ISP's homepages server. Using a slightly more complicated configuration we can select substrings of the server name to use in the filename so that e.g. the documents for `www.user.isp.com` are found in `/home/user/`. It uses a single `cgi-bin` directory instead of one per virtual host.

```

# all the preliminary stuff is the same as above, then
# include part of the server name in the filenames
VirtualDocumentRoot /www/hosts/%2/docs
# single cgi-bin directory
ScriptAlias /cgi-bin/ /www/std-cgi/

```

### Use more than one virtual hosting system on the same server

With more complicated setups you can use HTTP Server's normal `<VirtualHost>` directives to control the scope of the various virtual hosting configurations. For example, you could have one IP address for homepages customers and another for commercial customers with the following setup. This can of course be combined with conventional `<VirtualHost>` configuration sections.

```

UseCanonicalName off

LogFormat "%V %h %l %u %t \"%r\" %s %b" vcommon

<Directory /www/commercial>
Options FollowSymLinks
AllowOverride All
</Directory>
<Directory /www/homepages>
Options FollowSymLinks
AllowOverride None
</Directory>
<VirtualHost 10.22.33.44>
ServerName www.commercial.isp.com
CustomLog logs/access_log.commercial vcommon
VirtualDocumentRoot /www/commercial/%0/docs
VirtualScriptAlias /www/commercial/%0/cgi-bin
</VirtualHost>
<VirtualHost 10.22.33.45>
ServerName www.homepages.isp.com
CustomLog logs/access_log.homepages vcommon
VirtualDocumentRoot /www/homepages/%0/docs
ScriptAlias /cgi-bin/ /www/std-cgi/
</VirtualHost>

```

### Directory name interpolation

All the directives in this module interpolate (insert) a string into a pathname. The interpolated string may either be the server name (see "UseCanonicalName" on page 331 for more information) or the IP address of the virtual host on the server in dotted-quad format. The interpolation is controlled by specifiers inspired by UNIX `printf` which have a number of formats:

Specifier	Description
%%	Insert a % sign
%P	Insert the port number of the virtual host
%N.M	Insert (part of) the interpolated string

**N** and **M** are used to specify substrings of the interpolated string. **N** selects from the period separated components of the interpolated string, and **M** selects characters within whatever **N** has selected. **M** is optional and defaults to zero if it is not present. The period (.) must be present if and only if **M** is present. The interpretation is as follows:

N.M interpretation	Description
0	The whole name.
1	The first part.
2	The second part.
-1	The last part.
-2	The next to last part.
2+	The second and all subsequent parts.
-2+	The next to last part and all preceding parts.
1+ and -1+	The same as 0 (zero).

If **N** or **M** is greater than the number of parts available a single underscore is interpolated.

For a simple name-based virtual hosts you might use the following directives in your server configuration file:

```
UseCanonicalName off
```

```
VirtualDocumentRoot
```

```
/usr/local/www.example.isp.com/vhosts/%0
```

A request for `http://www.example.com/directory/file.html` will be satisfied by the file `/usr/local/www.example.isp.com/vhosts/www.example.com/directory/file.html`.

For a very large number of virtual hosts it is a good idea to arrange the files to reduce the size of the vhosts directory. To do this you might use the following in your configuration file:

```
UseCanonicalName off
```

```
VirtualDocumentRoot
```

```
/usr/local/www.example.isp.com/vhosts/%3+/%2.1/%2.2/%2.3/%2
```

A request for `http://www.example.isp.com/directory/file.html` will be satisfied by the file `/usr/local/www.example.isp.com/isp.com/e/x/a/example/directory/file.html`.

A more even spread of files can be achieved by hashing from the end of the name, for example:

```
VirtualDocumentRoot
```

```
/usr/web/www.example.isp.com/vhosts/%3+/%2.-1/%2.-2/%2.-3/%2
```

The example request would come from `/usr/web/www.example.isp.com/vhosts/isp.com/e/l/p/example/directory/file.html`. Alternatively you might use:

```
VirtualDocumentRoot
```

```
/usr/local/www.example.isp.com/vhosts/%3+/%2.1/%2.2/%2.3/%2.4+
```

The example request would come from `/usr/web/www.example.isp.com/vhosts/isp.com/e/x/a/mple/directory/file.html`. For IP-based virtual hosting you might use the following in your configuration file:

UseCanonicalName DNS

```
VirtualDocumentRootIP /usr/local/www.example.isp.com/vhost/%1/%2/%3/%4/docs
VirtualScriptAliasIP
/usr/local/www.example.isp.com/vhost/%1/%2/%3/%4/cgi-bin
```

A request for `http://www.example.isp.com/directory/file.html` would be satisfied by the file `/usr/local/www.example.isp.com/10/20/30/40/docs/directory/file.html` if the IP address of `www.example.com` were 10.20.30.40. A request for `http://www.example.isp.com/cgi-bin/script.pl` would be satisfied by executing the program `/usr/local/www.example.isp.com/10/20/30/40/cgi-bin/script.pl`.

The `LogFormat` directives `%V` and `%A` are useful in conjunction with this module. See “`LogFormat`” on page 411 for more information.

## Directives

- “`VirtualDocumentRoot`”
- “`VirtualDocumentRootIP`” on page 481
- “`VirtualScriptAlias`” on page 482
- “`VirtualScriptAliasIP`” on page 483

## VirtualDocumentRoot:

**Module:** `mod_vhost_alias`

**Syntax:** `VirtualDocumentRoot interpolated-directory`

**Default:** `none`

**Context:** `server config, virtual host`

**Override:** `none`

**Origin:** `Apache`

**Usage Considerations:** A `LoadModule` is required in the config file prior to using the directive. The statement should be as follows: `LoadModule vhost_alias_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM`

**Example:** See below.

The `VirtualDocumentRoot` directive allows you to determine where the server will find your documents based on the value of the server name. The result of expanding `interpolated-directory` is used as the root of the document tree in a similar manner to the `DocumentRoot` disabled. See “`DocumentRoot`” on page 293 for more information.

If `interpolated-directory` is `none` then `VirtualDocumentRoot` is disabled. This directive cannot be used in the same context as `VirtualDocumentRootIP`. See “`VirtualDocumentRootIP`” on page 481 for more information.

**Parameter:** `interpolated-directory`

- The `interpolated-directory` parameter the full path to a directory.

For example, a simple dynamic virtual host:

```
# LocalModule directive required
LoadModule vhost_alias_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
# get the server name from the Host: header
UseCanonicalName off
# this log format can be split per-virtual-host based on the first field
LogFormat "%V %h %l %u %t \"%r\" %s %b" vcommon
CustomLog logs/access_log vcommon
# include the server name in the filenames used to satisfy requests
VirtualDocumentRoot /www/web/%0/docs
```

The next example is an adjustment of the above, system tailored for an ISP's homepage server. Using a slightly more complicated configuration we can select substrings of the server name to use in the filename so that e.g. the documents for www.user.isp.com are found in /home/user/. It uses a single cgi-bin directory instead of one per virtual host:

```
# LocalModule directive required
LoadModule vhost_alias_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
# get the server name from the Host: header
UseCanonicalName off
# include part of the server name in the filenames
VirtualDocumentRoot /usr/web/hosts/%2/docs
# single cgi-bin directory
ScriptAlias /cgi-bin/ /usr/web/std-cgi/
```

**Note:** This configuration can be changed into an IP-based virtual hosting solution by just turning UseCanonicalName off into UseCanonicalName DNS. The server name that is inserted into the filename is then derived from the IP address of the virtual host. See "UseCanonicalName" on page 331 for more information.

### VirtualDocumentRootIP:

**Module:** mod\_vhost\_alias

**Syntax:** VirtualDocumentRootIP *interpolated-directory*

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the config file prior to using the directive. The statement should be as follows: LoadModule vhost\_alias\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** See VirtualDocumentRoot.

**Parameter:** *interpolated-directory*

- The *interpolated-directory* parameter the full path to a directory.

More complicated setups can use the server's normal <VirtualHost> directives to control the scope of the various virtual hosting configurations. For example, you could have one IP address for home page customers and another for commercial customers with the following directives. This can of course be combined with conventional <VirtualHost> configuration sections.

```
UseCanonicalName off

LogFormat "%V %h %l %u %t \"%r\" %s %b" vcommon

<Directory /usr/web/commercial>
Options FollowSymLinks
AllowOverride All
</Directory>

<Directory /usr/web/homepages>
Options FollowSymLinks
AllowOverride None
</Directory>
# LocalModule directive required
LoadModule vhost_alias_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

<VirtualHost 10.22.33.44>
ServerName www.commercial.isp.com
CustomLog logs/access_log.commercial vcommon
VirtualDocumentRoot /usr/web/commercial/%0/docs
VirtualScriptAlias /usr/web/commercial/%0/cgi-bin
</VirtualHost>
```

```

<VirtualHost 10.22.33.45>
ServerName www.homepages.isp.com
CustomLog logs/access_log.homepages vcommon
VirtualDocumentRoot /usr/web/homepages/%0/docs
ScriptAlias /cgi-bin/ /usr/web/std-cgi/
</VirtualHost>

```

### More efficient IP-based virtual hosting:

In the first example note that it is easy to turn it into an IP-based virtual hosting setup. Unfortunately that configuration is not very efficient because it requires a DNS lookup for every request. This can be avoided by laying out the filesystem according to the IP addresses themselves rather than the corresponding names and changing the logging similarly. HTTP Server will not usually need to work out the server name and a DNS lookup. For example:

```

# Get the server name from the reverse DNS of the IP address
UseCanonicalName DNS
# LocalModule directive required
LoadModule vhost_alias_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

# include the IP address in the logs so they may be split
LogFormat "%A %h %l %u %t \"%r\" %s %b" vcommon
CustomLog logs/access_log vcommon
# include the IP address in the filenames
VirtualDocumentRootIP /usr/web/hosts/%0/docs
VirtualScriptAliasIP /usr/web/hosts/%0/cgi-bin

```

The VirtualDocumentRootIP directive is like the VirtualDocumentRoot directive, except that it uses the IP address of the server end of the connection instead of the server name. See “VirtualDocumentRoot” on page 480 for more information.

### VirtualScriptAlias:

**Module:** mod\_vhost\_alias

**Syntax:** VirtualScriptAlias *interpolated-directory*

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the config file prior to using the directive. The statement should be as follows: LoadModule vhost\_alias\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** See below.

**Parameter:** *interpolated-directory*

- The *interpolated-directory* parameter the full path to a directory.

### Using more than one virtual hosting system on the same server instance:

More complicated setups use the server's normal <VirtualHost> directives to control the scope of the various virtual hosting configurations. For example, you could have one IP address for homepages customers and another for commercial customers with the following directives. This can of course be combined with conventional <VirtualHost> configuration sections.

```

UseCanonicalName off

LogFormat "%V %h %l %u %t \"%r\" %s %b" vcommon

<Directory/usr/web/commercial>
Options FollowSymLinks
AllowOverride All

```



```

</Directory>
<Directory /usr/web/homepages>
Options FollowSymLinks
AllowOverride None
</Directory>

# LocalModule directive required
LoadModule vhost_alias_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
<VirtualHost 10.22.33.44>
ServerName www.commercial.isp.com
CustomLog logs/access_log.commercial vcommon
VirtualDocumentRoot /usr/web/commercial/%0/docs
VirtualScriptAlias /usr/web/commercial/%0/cgi-bin
</VirtualHost>

<VirtualHost 10.22.33.45>
ServerName www.homepages.isp.com
CustomLog logs/access_log.homepages vcommon
VirtualDocumentRoot /usr/web/homepages/%0/docs
ScriptAlias /cgi-bin/ /usr/web/std-cgi/
</VirtualHost>

```

**Note:** The VirtualScriptAlias directive allows you to specify the directory path where the server will find CGI scripts in a similar manner to “VirtualDocumentRoot” on page 480 does for other documents. In this case the target directory of the CGI scripts must be named "cgi-bin". For example:

```
VirtualScriptAlias /user/web/commercial/%0/cgi-bin
```

### VirtualScriptAliasIP:

**Module:** mod\_vhost\_alias

**Syntax:** VirtualScriptAliasIP *interpolated-directory*

**Default:** none

**Context:** server config, virtual host

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the config file prior to using the directive. The statement should be as follows: LoadModule vhost\_alias\_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM

**Example:** See “VirtualScriptAlias” on page 482.

The VirtualScriptAliasIP directive is like the “VirtualScriptAlias” on page 482 directive, except that it uses the IP address of the server end of the connection instead of the server name.

**Parameter:** *interpolated-directory*

- The *interpolated-directory* parameter the full path to a directory.

### Module mod\_version

Module mod\_version supports directives for the IBM HTTP Server for i Web server.

#### Summary

The module mod\_version is designed for use in test suites and large networks which have to deal with different httpd versions and different configurations. It provides a new container that enables flexible version checking, including numeric comparisons and regular expressions. This module is helpful when using the same configuration across different apache versions and IBM i releases.

## <IfVersion>:

**Module:** mod\_version

**Syntax:** <IfVersion> [[!]operator] version> ... </IfVersion>

**Default:** none

**Context:** server, virtual host, directory, .htaccess

**Override:** All

**Origin:** Apache

The <IfVersion> section encloses configuration directives which are executed only if the Apache HTTP version matches the desired criteria. For normal (numeric) comparisons the version argument has the format major[.minor[.patch]], e.g. 2.0.56 or 2.2. minor and patch are optional. If these numbers are omitted, they are assumed to be zero.

The following numerical operators are possible:

*Table 31. Mod\_version operators*

Operator	Description
= or ==	Apache HTTP version is equal
>	Apache HTTP version is greater than
>=	Apache HTTP version is greater or equal
<	Apache HTTP version is less than
<=	Apache HTTP version is less or equal

### Example

```
<IfVersion >= 2.0>
# this happens only in versions greater or
# equal 2.0
</IfVersion>
```

**Note:** If the operator is omitted, it is assumed to be =.

## Module mod\_proxy\_balancer

Module mod\_proxy\_balancer supports directives for the IBM HTTP Server for i Web server.

### Summary

The Proxy balancer module has been added to provide load balancing services for the main mod\_proxy proxy module. The load balancer enables requests to be shared among workers via two methods, request counting and weighted traffic counting. Request counting just counts the number of requests and distributes requests across workers until they have each served an equal number of requests.

If you are using proxy load balancing with a Proxy directive scheme that starts with balancer://, you will need to specify the working members of the cluster. The balancer://xxx specification refers to a virtual worker that gets created. It does not really communicate with the backend server but is responsible for the management of several "real" workers. A special set of parameters can be added to this virtual worker. The "real" worker is a member of the load balancer, usually a remote host serving one of the supported protocols. Review ProxyPass directive for all of the allowed parameters.

### Directives

- "BalancerMember" on page 485
- "ProxySet" on page 485

## BalancerMember:

**Module:** mod\_proxy\_balancer

**Syntax:** BalancerMember [*path*] !|url [*key=value key=value ...*]

**Default:** none

**Context:** server, virtual host, directory.

**Override:** none

**Origin:** Apache

### Example 1

```
ProxyPass /special-area http://special.example.com/ smax=5 max=10
ProxyPass / balancer://mycluster stickysession=jsessionid nofailover
<Proxy balancer://mycluster>
BalancerMember http://1.2.3.4:8009
BalancerMember http://1.2.3.5:8009 smax=10
# Less powerful server, don't send as many requests there
BalancerMember http://1.2.3.6:8009 smax=1 loadfactor=20
</Proxy>
```

### Example 2

```
<Proxy balancer://mycluster2>
BalancerMember http://196.128.0.1:4000
BalancerMember http://196.128.0.1:4001
</Proxy>
```

Then you proxy the location or virtual host to the cluster:

```
<VirtualHost *:80>
ProxyPass / balancer://mycluster2/
ProxyPassReverse / balancer://mycluster2/
</VirtualHost>
```

**Note:** The slash must occur after the ProxyPass directive.

## ProxySet:

**Module:** mod\_proxy\_balancer

**Syntax:** ProxySet *parms*

**Default:** none

**Context:** Server, Virtual Host, Directory

**Override:** none

**Origin:** Apache

**Usage Considerations:** A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows:

```
LoadModule proxy_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_ftp_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule proxy_http_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
LoadModule balancer_proxy_module /QSYS.LIB/QHTTPSVR.LIB/QZSRCORE.SRVPGM
```

### Example:

```
RewriteRule ^/mbp/(.*) balancer://ourcluster/$1 [P]
<Proxy balancer://myCluster>
.....
ProxySet stickysession=OUR_COOKIE timeout=20 nofailover=On
</Proxy>
```

The ProxySet is similar to the ProxyPass directive. If the ProxySet directive scheme starts with the balancer:// then a virtual worker that does not really communicate with the backend server will be created. Instead it is responsible for the management of several "real" workers. In that case the special set

of parameters can be added to this virtual worker. These same parameters can also be added to the ProxySet directive when balancer:// is specified. The following table displays the parameters for all proxy workers in a group or cluster.

Table 32. ProxySet parameters

Header	Default	Description
lbmethod	-	Balancer load-balance method. Select the load-balancing scheduler method to use. Either byrequests, to perform weighted request counting or bytraffic, to perform weighted traffic byte count balancing. Default is byrequests
stickysession	-	Balancer sticky session name. The value is usually set to something like JSESSIONID or PHPSESSIONID, and it depends on the backend application server that support sessions.
nofailover	Off	If set to On the session will break if the worker is in error state or disabled. Set this value to On if backend servers do not support session replication.
timeout	0	Balancer timeout in seconds. If set this will be the maximum time to wait for a free worker. Default is not to wait.
maxattempts	1	Maximum number of failover attempts before giving up.

## Module mod\_filter

Module mod\_filters supports directives for the IBM HTTP Server for i Web server.

### Summary

The filter module provides context-sensitive configuration of output content filters. This module by gives server administrators a great deal of flexibility in configuring the filter chain. In fact, filters can be inserted based on any Request Header, Response Header or Environment Variable. The filter module has been extended to allow filters to be executed based on conditional criteria. This changes the old model under which documents were merely filtered unconditionally according to the configuration of the *AddOutputFilter* directive or the minor flexibility offered by *AddOutputFilterByType*. Instead of adding specific filters to specific file types you can create a proper filter chain output is processed by each filter in the chain. This requires a declaration of the available filter types, and if necessary, the source requirements (file type) and the filters to apply.

The filter module works by introducing indirection into the filter chain. Instead of inserting filters in the chain, we insert a filter harness which in turn dispatches conditionally to a filter provider. Any content filter may be used as a provider to mod\_filter; no change to existing filter modules is required (although it may be possible to simplify them). There can be multiple providers for one filter, but no more than one provider will run for any single request.

A filter chain comprises any number of instances of the filter harness, each of which may have any number of providers. A special case is that of a single provider with unconditional dispatch, which is equivalent to inserting the provider filter directly into the chain.

## Directives

- “FilterChain”
- “FilterDeclare” on page 488
- “FilterProvider” on page 488

## FilterChain:

**Module:** mod\_filter

**Syntax:** FilterChain [+=@!] filter-name ...

**Default:** none

**Context:** Server, Virtual Host, Directory, .htaccess

**Override:** Options

**Origin:** Apache

## Examples

The following example demonstrates using the mod\_filter in place of the AddOutputFilterByType directive.

### Server side Includes (SSI)

```
FilterDeclare SSI
FilterProvider SSI INCLUDES resp=Content-Type $text/html
FilterChain SSI
```

The following example demonstrates using the mod\_filter in place of the AddOutputFilterByType directive, but dispatching is done on the handler.

### Server side Includes (SSI)

```
FilterProvider SSI INCLUDES Handler server-parsed
FilterChain SSI
```

The following example demonstrates Insert INFLATE filter only if gzip is not in the Accept-Encoding header. This filter runs with ftype CONTENT\_SET.

### Emulating mod\_gzip with mod\_deflate

```
FilterDeclare gzip CONTENT_SET
FilterProvider gzip inflate req=Accept-Encoding !$gzip
FilterChain gzip
```

The following are the three stages used to configuring a filter chain with mod\_filter.

### Declare Filters

The FilterDeclare directive declares a filter, assigning it a name and filter type. Required only if the filter is not the default type AP\_FTYPE\_RESOURCE.

### Register Providers

The FilterProvider directive registers a provider with a filter. The filter may have been declared with FilterDeclare; if not, FilterProvider will implicitly declare it with the default type AP\_FTYPE\_RESOURCE. The provider must have been registered with ap\_register\_output\_filter by some module. The remaining arguments to FilterProvider are a dispatch criterion and a match string. The former may be an HTTP request or response header, an environment variable, or the Handler used by this request. The latter is matched to it for each request, to determine whether this provider will be used to implement the filter for this request.

### Configure the Chain

The above directives build components of a smart filter chain, but do not configure it to run. The FilterChain directive builds a filter chain from smart filters declared, offering the flexibility to insert filters at the beginning or end of the chain, remove a filter, or clear the chain.

FilterChain takes any number of arguments, each optionally preceded with a single-character control that determines what to do. The following information configures an actual filter chain from declared filters.

**+filter-name**

Adds filter-name to the end of the filter chain.

**@filter-name**

Inserts filter-name at the start of the filter chain

**-filter-name**

Removes filter-name from the filter chain

**=filter-name**

Empty the filter chain and insert filter-name

!

Empty the filter chain

**filter-name**

Equivalent to +filter-name

**FilterDeclare:**

**Module:** mod\_filter

**Syntax:** FilterDeclare *filter-name* [*type*]

**Default:** none

**Context:** Server, Virtual Host, Directory, .htaccess

**Override:** Options

**Origin:** Apache

This directive declares an output filter together with a header or environment variable that will determine runtime configuration. The first argument is a filter-name for use in FilterProvider and FilterChain directives. The final (optional) argument is the type of filter, and takes values of ap\_filter\_type - namely RESOURCE (the default), CONTENT\_SET, TRANSCODE, PROTOCOL, CONNECTION, or NETWORK.

**FilterProvider:**

**Module:** mod\_filter

**Syntax:** FilterProvider *filter-name provider-name* [*req* | *resp* | *env*]=*dispatch match*

**Default:** none

**Context:** Server, Virtual Host, Directory, .htaccess

**Override:** Options

**Origin:** Apache

**Example:** CacheTimeMargin 300

The FilterProvider directive registers a provider for the smart filter. The provider will be called only if the match declared here matches the value of the header or environment variable declared as dispatch.

The provider-name is registered by loading a module that registers the name with ap\_register\_output\_filter.

The dispatch argument is a string with optional req=, resp= or env= prefix causing it to dispatch on the request header, response header, or environment variable named. In the absence of a prefix, it defaults to a response header. A special case is the word handler, which causes mod\_filter to dispatch on the content handler.

The match argument specifies a match that will be applied to the filter's dispatch criterion. The match may be a string match (exact match or substring), a regex, an integer (greater, less-than or equals), or unconditional. The first characters of the match argument determines the following:

1. If the first character is an exclamation mark (!), this reverses the rule, so the provider will be used if and only if the match fails.
2. If the first character is not a leading ! it interprets the following character:

Table 33. Character interpretation

Characters	Description
(none)	exact match
\$	substring match
/	regex match (delimited by a second /)
=	integer equality
<	integer less-than
<=	integer less-than or equal
>	integer greater-than
>=	integer greater-than or equal
*	Unconditional match

## Module mod\_ident

Module mod\_ident supports directives for the IBM HTTP Server for i Web server.

### IdentityCheckTimeout:

**Module:** mod\_ident

**Syntax:** IdentityCheckTimeout *seconds*

**Default:** IdentityCheckTimeout 30

**Context:** Server, Virtual Host, Directory

**Override:** none

**Origin:** Apache

The IdentityCheckTimeout directive determines the timeout duration for ident requests. The default value of 30 seconds is recommended by RFC 1413, mainly because of possible network latency. However, you may want to adjust the timeout value according to your local network speed.

## Log file format tokens

This topic provides information about tokens used to define log file formats.

Table 34. Tokens that define log file formats

Token	Description
%. . . a	The remote client IP address. Example: 192.168.1.3
%. . . A	The local client IP address. Example: 192.168.1.3
%. . . b	The number of bytes transmitted, excluding HTTP headers in common log format. Example: - = no bytes transmitted
%. . . B	The number of bytes transmitted, excluding HTTP headers in extended log format. Example: 0 = no bytes transmitted
%. . . {var}e	The contents of the environment variable named var.
%. . . f	The requested file name. Example: /www/index.htm

Table 34. Tokens that define log file formats (continued)

Token	Description
%.h	The remote host name or IP address. Example: ha1.ibm.com or 192.168.1.3
%.H	The requested protocol.
%...{var}i	The contents of the HTTP header line named var. Example: %{User-agent}i = Mozilla/4.5 [en] (WinNT; U)
%.l	The remote logname.
%.m	The request method.
%...{var}n	The contents of the note named var.
%...{var}o	The contents of the header lines named var in the reply.
%.p	The canonical Port of the server serving the request. Example: 80
%.P	The process ID that serviced the request. Example: 837
%.q	The query string (or search argument) prepended with a "?". Example: ?name=ha1
%.r	The first line of the request. Example: GET / HTTP/1.0
%.s	The server response status. Example: 200
%.t	The time in common log format. Example: [21/Mar/2000:14:08:03 -0600]
%...{strftime}t	The time in strftime format.
%.T	The time (in seconds) taken to serve the request. Example: 1
%.u	The name of the authenticated remote user. Example: ha1
%.U	The requested URL path. Example: /
%.v	The canonical server name of the server serving the request.
%.V	The server name according to the UseCanonicalName setting.

**Note:**

- Logformat %D is not supported.
- The "..." can be replaced with a condition for inclusion or it can be omitted. The character < determines if the original value is logged. The greater than character (>) determines if the redirected value is logged. The condition may be preceded by a ! to reverse the condition. For example:

Condition	Description
%>s	Logs the returned status.
%{User-agent}i	Logs User-agent on all requests.
%400,501{User-agent}i	Logs User-agent only when a 400 error (Bad Request) or a 501 error (Not Implemented) is encountered.
%!200,304,302{Referer}i	Logs Referer on all requests which did not return some sort of normal status.

**Related information:**

“Log formats for HTTP Server” on page 27

This topic provides information about log formats and log files.

“Setting up logs on HTTP Server” on page 112

Set up logs to record events and other information for your IBM HTTP Server for i instance using the IBM Web Administration for i interface.



## Regular expression notation for HTTP Server

This topic provides a general overview of regular expression notation for the IBM HTTP Server for i Web server.

A regular expression notation specifies a pattern of character strings. One or more regular expressions can be used to create a matching pattern. Certain characters (sometimes called wildcards) have special meanings. The following table describes the pattern matching scheme.

### Regular expression pattern matching

Pattern	Description
string	string with no special characters matches the values that contain the string.
[set]	Match a single character specified by the set of single characters within the square brackets.
[a-z]	Match a character in the range specified within the square brackets.
[^abc]	Match any single character not specified in the set of single characters within the square brackets.
{n}	Match exactly n times.
{n,}	Match at least n times.
{n,m}	Match at least n times, but no more than m times.
^	Match the start of the string.
\$	Match the end of the string.
.	Match any character (except Newline).
*	Match zero or more of preceding character.
+	Match one or more of preceding character.
?	Match one or zero of preceding character.
string1 string2	Match string1 or string2.
\	Signifies an escape character. When preceding any of the characters that have special meaning, the escape character removes any special meaning from the character. For example, the backslash is useful to remove special meaning from a period in an IP address.
(group)	Group a character in a regular expression. If a match is found the first group can be accessed using \$1. The second group can be accessed using \$2 and so on.
\w	Match an alphanumeric character.
\s	Match a white-space character.
\t	Tab character.
\n	Newline character.
\r	Return character.
\f	Form feed character.
\v	Vertical tab character.
\a	Bell character.
\e	Escape character.
\0nn	Octal character, for example \076.
\xnn	Hex character, for example \xff.
\cn	Control character, for example \c[.
\l	Lowercase next character.

Pattern	Description
\L	Lowercase until \E.
\u	Uppercase next character.
\U	Uppercase until \E.
\E	End modification.
\Q	Quote until \E.

### Examples of regular expression pattern matching

Pattern	Examples of strings that match
ibm	ibm01, myibm, aibmbc
^ibm\$	ibm
^ibm0[0-4][0-9]\$	ibm000 through ibm049
ibm[3-8]	ibm3, myibm4, aibm5b
^ibm	ibm01, ibm
ibm\$	myibm, ibm, 3ibm
ibm...	ibm123, myibmabc, aibm09bcd
ibm*1	ibm1, myibm1, aibm1abc, ibmkkkkk12
^ibm0..	ibm001, ibm099, ibm0abcd
^ibm0..\$	ibm001, ibm099
10.2.1.9	10.2.1.9, 10.2.139.6, 10.231.98.6
^10\.2\.1\.9\$	10.2.1.9
^10\.2\.1\.1[0-5]\$	10.2.1.10, 10.2.1.11, 10.2.1.12, 10.2.1.13, 10.2.1.14, 10.2.1.15
^192\.168\.*\.*\$	(All addresses on class B subnet 192.168.0.0)
^192\.168\.10\.*\$	(All addresses on class C subnet 192.168.10.0)

## CL commands for HTTP Server

Manage an IBM HTTP Server for i instance using command line (CL) commands for a 5250 session.

The following table summarizes the CL commands associated with the HTTP Server.

Command	Description
CFGTCPHHTTP	Configure TCP/IP HTTP - display a list of HTTP Server related commands.
ENDCHTSVR	End Clustered Hash Table Server - end a clustered hash table (CHT) server job on one or more nodes of a cluster.
ENDTCP	End TCP/IP - stop TCP/IP.
ENDTCPSPVR	End TCP/IP Server - stop an HTTP Server.
STRCHTSVR	Start Clustered Hash Table Server - start a job for a clustered hash table (CHT) server instance on each of the specified node systems.
STARTCP	Start TCP/IP - autostart an HTTP Server.
STARTCPSPVR	Start TCP/IP Server - start an HTTP server.
TRCTCPAPP	Trace TCP/IP Application - capture trace information for the HTTP Server.

The following table summarizes the CL commands associated with the highly available Web server cluster function. See "Highly available HTTP Server" on page 43 for more information.

Command	Description
ADDCLUNODE	Add Cluster Node Entry - add a node to the membership list of an existing cluster.
CHGCLUNODE	Change Cluster Node Entry - change cluster membership information for a cluster node entry.
CHGCRGPRI	Change Cluster Resource Group Primary - performs an administrative switchover of the cluster resource group by changing the current roles of nodes in the recovery domain.
CRTCLU	Create Cluster - create a new cluster of one or more nodes.
DSPCLUINF	Display Cluster Information - display or print information about a cluster.
ENDCLUNOD	End Cluster Node - end Cluster Resource Services on one or all the nodes in the membership list of an existing cluster.
RMVCLUNODE	Remove Cluster Node Entry - remove a node from a cluster.
STRCLUNOD	Start Cluster Node - start Cluster Resource Services on a node in the cluster.

## Environment variables set by HTTP Server

The IBM HTTP Server for i supports the standard environment variables in addition to environment variables that are unique to the IBM i server.

When using application programming interfaces to retrieve the value of an environment variable, you need to handle the case in which there is no value for the environment variable. For example, when a CGI program is trying to do a `getenv(ΔCONTENT_LENGTHΔ)` and the request method is GET, the value returned is NULL. The reason NULL is returned for the value is because CONTENT\_LENGTH is only defined for POST request methods (to describe the length of standard input).

The following table lists the environment variables supported by HTTP Server. The environment variables have been divided into two groups: Non-SSL and SSL.

### Notes:

1. All headers sent by a client (such as Set-Cookie) are prefixed by "HTTP\_". To access the value of a header, prefix the header name with "HTTP\_".
2. In the following table, long variable names are shortened by the insertion of a blank character within the name. This is done for display purposes only.

*Table 35. Environment variables that may be set by the HTTP Server*

Variable Name	Type	Description
AUTH_TYPE	Non-SSL	If the server supports client authentication and the script is a protected script, this environment variable contains the method that is used to authenticate the client.  Example: Cert_Or_Basic
CGI_ASCII_CCSID	Non-SSL	Contains the ASCII CCSID the server used when converting CGI input data. If the server did not perform any conversion, (for Example, in %%BINARY%% mode), the server sets this value to the DefaultNetCCSID configuration directive value.  Example: 819
CGI_EBCDIC_CCSID	Non-SSL	Contains the EBCDIC CCSID under which the current CGI job is running (DefaultFsCCSID or CGIJobCCSID configuration directive). It also represents the job CCSID that is used during server conversion (if any) of CGI input data.  Example: 37

Table 35. Environment variables that may be set by the HTTP Server (continued)

Variable Name	Type	Description
CGI_JOB_LOCALE	Non-SSL	Allows a locale to be set globally or for a specific CGI job. After the locale is set, region specific information such as date or time format can be accessed. Some ILE C/C++ run-time functions such as ctime() and localtime() are locale sensitive.  Example: CGIJobLocale /QSYS.LIB/LOCALELIB.LIB/EN_US.LOCALE
CGI_MODE	Non-SSL	Contains the CGI conversion mode the server is using for this request. The program can use this information to determine what conversion, if any, was performed by the server on CGI input data and what format that data is currently in.  Example: EBCDIC
CGI_OUTPUT_MODE	Non-SSL	Determines which output conversion mode the server is using.  Example: EBCDIC
CONTENT_LENGTH	Non-SSL	When the method of POST is used to send information, this variable contains the number of characters. Servers typically do not send an end-of-file flag when they forward the information by using stdin. If needed, you can use the CONTENT_LENGTH value to determine the end of the input string.  Example: 7034
CONTENT_TYPE	Non-SSL	When information is sent with the method of POST, this variable contains the type of data included. You can create your own content type in the server configuration file and map it to a viewer.  Example: Application/x-www-form-urlencoded
DATE_GMT	Non-SSL	The current date and time in Greenwich Mean Time.  Example: 2000/12/31:03:15:20
DATE_LOCAL	Non-SSL	The current date and time in the local time zone.  Example: 2000/08/14:15:40:10
DOCUMENT_NAME	Non-SSL	The file name of the document requested by the user.  Example: /www/myserver/htdocs/html/hello.html
DOCUMENT_PATH_INFO	Non-SSL	Contains the additional path information as sent by the Web browser for SSL.  Example: /wizard
DOCUMENT_ROOT	Non-SSL	Sets the directory from which the HTTP Server will serve files. The server appends the path from the requested URL to the document root and makes the path to the document.  Example: /www/myserver/htdocs
DOCUMENT_URI	Non-SSL	The URI of the document requested by the user.  Example: /html/hello.html <b>Note:</b> The DOCUMENT_URI and DOCUMENT_URL environment variables are identical
DOCUMENT_URL	Non-SSL	The URL of the document requested by the user.  Example: /html/hello.html <b>Note:</b> The DOCUMENT_URI and DOCUMENT_URL environment variables are identical.
FSCP	Non-SSL	The EBCDIC CCSID used to translate the data.  Example: 37
GATEWAY_INTERFACE	Non-SSL	Contains the version of CGI that the server is using.  Example: CGI/1.1

Table 35. Environment variables that may be set by the HTTP Server (continued)

Variable Name	Type	Description
HTTP_ACCEPT	Non-SSL	Contains the list of MIME types the browser accepts.  Example: image/gif,image/x-xbitmap,image/jpeg,image/png,*/* <b>Note:</b> The header lines received from the client, if any, are placed into the environment variable with the prefix HTTP_* followed by the header name. The header returns the environment variable.
HTTP_ACCEPT_CHARSET	Non-SSL	Contains the list of character sets the browser accepts.  Example: iso-8859-1*,utf-8 <b>Note:</b> The header lines received from the client, if any, are placed into the environment variable with the prefix HTTP_* followed by the header name. The header returns the environment variable.
HTTP_ACCEPT_ENCODING	Non-SSL	Contains the list of encoding protocols the browser accepts.  Example: gzip <b>Note:</b> The header lines received from the client, if any, are placed into the environment variable with the prefix HTTP_* followed by the header name. The header returns the environment variable.
HTTP_ACCEPT_LANGUAGE	Non-SSL	Contains the list of languages the browser accepts.  Example: de,fr,en <b>Note:</b> The header lines received from the client, if any, are placed into the environment variable with the prefix HTTP_* followed by the header name. The header will return the environment variable.
HTTP_CONNECTION	Non-SSL	The header lines received from the client, if any, are placed into the environment variable with the prefix HTTP_* followed by the header name. The header returns to the environment variable.  Example: Keep-Alive
HTTP_COOKIE	Non-SSL	User-defined cookie for the response.  Example: w3ibmTest=true
HTTP_HOST	Non-SSL	Contains the HTTP host URL.  Example: IBM.COM <b>Note:</b> The header lines received from the client, if any, are placed into the environment variable with the prefix HTTP_* followed by the header name. The header returns the environment variable.
HTTP_USER_AGENT	Non-SSL	Contains the name of your browser (web client). It includes the name and version of the browser, requests that are made through a proxy, and other information.  Example: Mozilla/4.72 [en ](WinNT;U) <b>Note:</b> The header lines received from the client, if any, are placed into the environment variable with the prefix HTTP_* followed by the header name. The header returns the environment variable.
IBM_CCSID_VALUE	Non-SSL	The CCSID under which the current server job is running.  Example: 37
NETCP	Non-SSL	The default ASCII CCSID used to translate the data.  Example: 819
PATH_INFO	Non-SSL	Contains the additional path information as sent by the web browser.  Example: /wizard
PATH_TRANSLATED	Non-SSL	Contains the decoded or translated version of the path information that is contained in PATH_INFO, which takes the path and does any virtual-to-physical mapping to it.  Example: /wwwhome/wizard

Table 35. Environment variables that may be set by the HTTP Server (continued)

Variable Name	Type	Description
QIBM_CGI_LIBRARY_LIST	Non-SSL	This variable is used to set the CGI jobs' library list. The variable can be set using the SetEnv directive. See the SetEnv directive for more information.
QUERY_STRING	Non-SSL	When information is sent using a method of GET, this variable contains the information in a query that follows the "?". The string is coded in the standard URL format of changing spaces to "+" and encoding special characters with "%xx" hexadecimal encoding. The CGI program must decode this information.  Example: NAME=Eugene+T%2E+Fox=etfox%7Cibm.net=xyz <b>Note:</b> The supported maximum size of QUERY_STRING is 8K for HTTP Server.
QZHBHA_MODEL	Non-SSL	Model of the highly available Web server.  Example: PRIMARYBACKUP
QZHBIS_FIRST_REQUEST	Non-SSL	This environment variable indicates to a CGI program if this is a subsequent request of some session. The Web server sets this variable to 1 if this is not a subsequent request of any session (this is potentially the first request of a new session). The Web server sets this variable to 0 if this is a subsequent request of some session.  Example: 0
QZHBIS_CLUSTER_ENABLED	Non-SSL	This environment variable indicates to the CGI program that the CGI program is allowed to be cluster-enabled if the request does not belong to any existing session (QZHBIS_FIRST_REQUEST is set to 1). This environment variable indicates to the CGI program that the CGI program is cluster-enabled (QZHBIS_FIRST_REQUEST set to "0"). When the Web server receives a first request to a CGI, it decides if the CGI program is allowed to be cluster-enabled. If the CGI program is allowed to be cluster-enabled, the Web server sets the QZHBIS_CLUSTER_ENABLED environment variable to 1; otherwise the Web server does not define the QZHBIS_CLUSTER_ENABLED environment variable. When the Web server receives a subsequent request to a CGI, it looks to see if the session is cluster-enabled. If the session is cluster-enabled, the Web server sets the QZHBIS_CLUSTER_ENABLED environment variable to 1; otherwise the Web server does not define the QZHBIS_CLUSTER_ENABLED environment variable.  Example: 1
QZHBNEXT_SESSION_HANDLE	Non-SSL	This environment variable contains a new session handle for a CGI program to use. If the CGI program is cluster-disabled, it may ignore this session handle. The Web server generates a session handle and sets the QZHBNEXT_SESSION_HANDLE environment variable to this value. If the CGI program decides to be cluster-enabled, it must use the passed session handle in the URLs of subsequent requests; otherwise, the Web server will not associate subsequent requests with this session.  Example: 8B739003AB741824899F0004AC009021
QZHBRECOVERY	Non-SSL	Contains whether the highly available Web server has gone through a recovery (primary to backup or backup to primary). If this environment variable is present, recovery has occurred. If it is not present, then recovery has not occurred
REDIRECT_QUERY_STRING	Non-SSL	Contains QUERY_STRING from a re-directed request.  Example: NAME=Eugene+T%2E+Fox=etfox%7Cibm.net=xyz

Table 35. Environment variables that may be set by the HTTP Server (continued)

Variable Name	Type	Description
REDIRECT_QUERY_URL	Non-SSL	This environment variable is used in the primary/backup models only. This environment variable is used to indicate to a cluster-enabled CGI program that it should perform a recovery operation (for example, restore its state). The Web server passes a session handle to the CGI program through the QZHBRECOVERY environment variable. The Web server passes the CGI's state to the CGI program. If there is no recovery, this environment variable is undefined. In the primary/backup model, the high availability CGI is also treated as a persistent CGI. The high availability CGI state information can also be retained in the CGI job. The next request for the next step in the CGI is automatically run in the same job. Therefore, the CGI program can skip reading its state unless this environment variable is defined.  Example: 4D868803AB731824899F0004AC009021
REFERER	Non-SSL	Contains the referrer.  Example: http://www.myserver.com/cgi-bin/
HTTP_REFERER	Non-SSL	Contains the referrer.  Example: http://www.myserver.com/cgi-bin/
REFERER_URL	Non-SSL	Contains the referrer URL.  Example: http://WWW.MYSERVER.COM:8080/perlSetEnv/
REMOTE_ADDR	Non-SSL	Contains the IP address of the remote host (web browser) that is making the request, if available.  Example: 10.10.2.3
REMOTE_PORT	Non-SSL	Contains the remote user port number.  Example: 3630
REMOTE_IDENT	Non-SSL	Contains the user ID of the remote user.  Example: MyIdentityx
REMOTE_USER	Non-SSL	If you have a protected script and the server supports client authentication, this environment variable contains the user name that is passed for authentication.  Example: SMITH
REQUEST_METHOD	Non-SSL	Contains the method (as specified with the METHOD attribute in an HTML form) that is used to send the request.  Example: GET
REQUEST_URI	Non-SSL	Specifies URI to be requested.  Example: /cgi-bin/hello.pgm
RULE_FILE	Non-SSL	Specifies rule file to be used.  Example: /www/myserver/conf/httpd.conf
SCRIPT_FILENAME	Non-SSL	The file name of the document requested by the user.  Example: /QSYS.LIB/CGI.LIB/HELLO.PGM
SCRIPT_NAME	Non-SSL	A virtual path to the program being run. Use this for self-referring URLs.  Example: /cgi-bin/hello.pgm
SERVER_ADDR	Non-SSL	Contains the address of the server.  Example: 10.10.2.3
SERVER_ADMIN	Non-SSL	Contains information about the server administrator.  Example: [no address given ]

Table 35. Environment variables that may be set by the HTTP Server (continued)

Variable Name	Type	Description
SERVER_NAME	Non-SSL	Contains the server host name or IP address of the server. Example: 10.9.8.7
SERVER_PORT	Non-SSL	Contains the port number to which the client request was sent. Example: 2001
SERVER_PROTOCOL	Non-SSL	Contains the name and version of the information protocol that is used to make the request. Example: HTTP/1.0
SERVER_SIGNATURE	Non-SSL	Allows configuration of a trailing footer line under server generated documents like error messages, mod_proxy ftp directory listings, and mod_info output. Enabling the footer line allows the user to tell which chained servers in a proxy chain produced a returned error message. Example: On
SERVER_SOFTWARE	Non-SSL	Contains the name and version of the information server software that is answering the request. Example: IBM-HTTP-SERVER/1.0
SSI_DIR	Non-SSL	The path of the current file relative to SSI_ROOT. If the current file is in SSI_ROOT, this value is "/". Example: ssi_child_dir/
SSI_FILE	Non-SSL	The file name of the current file. Example: ssi_parent.shtml
SSI_INCLUDE	Non-SSL	The value that is used in the include command that retrieved this file. This is not defined for the topmost file. Example: ssi_child_dir/ssi_child.shtml
SSI_PARENT	Non-SSL	The path and file name of the include, relative to SSI_ROOT. Example: ssi_parent.shtml
SSI_ROOT	Non-SSL	The path of the topmost file. All include requests must be in this directory or a child of this directory. Example: #echo var=SSI_DIR -> <b>Note:</b> You can use echo to display a value set by the set or global directives.
UNIQUE_ID	Non-SSL	Provides a unique magic token and acts as the identifier across all requests under very specific conditions. Example: aK8YOAKFBZkAABsuEC4AAACB
HTTPS	SSL	Returns <b>ON</b> if the system has completed an SSL handshake. It returns <b>OFF</b> if the exchange of signals to set up communications between two modems has failed. Example: OFF
HTTPS_CIPHER	SSL	This is the cipher that is used to negotiate with the client on the SSL handshake. Example: SSL_RSA_WITH_RC4_128_MD5
HTTPS_CLIENT_CERT	SSL	The entire certificate passed to the server from the client browser when SSL client authentication is enabled. The format of the certificate is a BASE64 encoded string that represents the DER format of the X.509 certificate. As an environment variable the BASE64 encoded string has been converted to EBCDIC and must be converted back to ASCII before it can be used for typical digital certificate API's. Example: MIIC0DCCAbigAwIBAgIHOL2Yx...



Table 35. Environment variables that may be set by the HTTP Server (continued)

Variable Name	Type	Description
HTTPS_CLIENT_CERT_COMMON_NAME	SSL	The common name from the client certificate's distinguished name. Example: SMITH
HTTPS_CLIENT_CERT_COUNTRY	SSL	The region code from the client certificate's distinguished name. Example: US
HTTPS_CLIENT_CERT_DN	SSL	The client certificate's distinguished name. Example: :cn=CAPTAIN,ou=downtown,o=fire fighters,l=Minot,st=North Dakota,c=US
HTTPS_CLIENT_CERT_EMAIL	SSL	The email of the client owning the certificate. Example: me@mycompany.com
HTTPS_CLIENT_CERT_ISSUER_COMMON_NAME	SSL	The common name of the certificate authority that issued the client's certificate. Example: SMITH
HTTPS_CLIENT_CERT_ISSUER_COUNTRY	SSL	The region code of the certificate authority that issued the client's certificate. Example: US
HTTPS_CLIENT_CERT_ISSUER_DN	SSL	The distinguished name of the certificate authority that issued the client's certificate. Example: :cn=testsystem.ibm.com CA,ou=Test Organization Unit,o=System test,l=Rochester,st=Minnesota,c=US
HTTPS_CLIENT_CERT_ISSUER_EMAIL	SSL	The e-mail address of the certificate authority that issued the client's certificate. Example: me@mydomain.net
HTTPS_CLIENT_CERT_ISSUER_LOCALITY	SSL	The locality or city of the certificate authority that issued the client's certificate. Example: New York
HTTPS_CLIENT_CERT_ISSUER_ORG_UNIT	SSL	The organizational unit of the certificate authority that issued the client's certificate. Example: bird watchers
HTTPS_CLIENT_CERT_ISSUER_ORGANIZATION	SSL	The organization name of the certificate authority that issued the client's certificate. Example: dove
HTTPS_CLIENT_CERT_ISSUER_POSTAL_CODE	SSL	The postal code of the certificate authority that issued the client's certificate. Example: 12344-6789
HTTPS_CLIENT_CERT_ISSUER_STATE_OR_PROVINCE	SSL	The state or province of the certificate authority that issued the client's certificate. Example: North Dakota
HTTPS_CLIENT_CERT_LEN	SSL	The length of the certificate passed in HTTPS_CLIENT_CERT. Example: 968
HTTPS_CLIENT_CERT_LOCALITY	SSL	The locality or city of the client certificate's distinguished name. Example: New York
HTTPS_CLIENT_CERT_ORG_UNIT	SSL	The organization unit name from the client certificate's distinguished name. Example: Pack234
HTTPS_CLIENT_CERT_ORGANIZATION	SSL	The organization name from the client certificate's distinguished name. Example: Scouts

Table 35. Environment variables that may be set by the HTTP Server (continued)

Variable Name	Type	Description
HTTPS_CLIENT_CERT_POSTAL_CODE	SSL	The postal code assigned by the issuing certificate authority. Example: 80525
HTTPS_CLIENT_CERT_SERIAL_NUM	SSL	The serial number assigned by the issuing certificate authority. Example: 3F:E4:83:81:02:D5:58
HTTPS_CLIENT_CERT_STATE_OR_PROVINCE	SSL	The state or province from the client certificate's distinguished name. Example: Alberta
HTTPS_CLIENT_ISSUER_EMAIL	SSL	Contains the email address of the Certificate Authority that issued the certificate. Example: jones@mydomain.net
HTTPS_KEYSIZE	SSL	If a valid security product is installed and the SSLMode directive is SSLMode=ON, this will be set to the size of the bulk encryption key used in the SSL session. Example: [ 128 ]
HTTPS_SESSION_ID	SSL	Set to NULL by default when used with HTTP Server.
HTTPS_SESSION_ID_NEW	SSL	If the value is <b>TRUE</b> , it indicates that a full handshake was performed for this SSL session. If the value is <b>FALSE</b> , it indicates that an abbreviated handshake was performed for this SSL session. Example: True
SSL_CIPHER	SSL	This is the cipher that is used to negotiate with the client on the SSL handshake. Example: SSL_RSA_WITH_RC4_128_MD5
SSL_CLIENT_C	SSL	The region code from the client certificate's distinguished name. Example: USA
SSL_CLIENT_CERTBODY	SSL	The entire certificate passed to the server from the client browser when SSL Client authentication is enabled. The format of the certificate is a BASE64 encoded string that represents the DER format of the X.509 certificate. As an environment variable the BASE64 encoded string has been converted to EBCDIC and must be converted back to ASCII before it can be used for typical digital certificate APIs. Example: MIIC0DCC big IB gIHOL2Yx...
SSL_CLIENT_CERTBODYLEN	SSL	The length of the certificate passed in SSL_CLIENT_CERT. Example: 828
SSL_CLIENT_CERT_EMAIL	SSL	The email of the client owning the certificate. Example: me@mycompany.com
SSL_CLIENT_CN	SSL	The common name from the client certificate's distinguished name. Example: SMITH
SSL_CLIENT_DN	SSL	The client's distinguished name. Example: :cn=CAPTAIN,ou=downtown,o=fire fighters,l=Minot,st=North Dakota,c=US HTTPS_CLIENT_CERT_DN :cn=CAPTAIN,ou=downtown,o=fire fighters,l=Minot,st=North Dakota,c=US
SSL_CLIENT_ICN	SSL	The common name of the certificate authority that issued the client's certificate. Example: SMITH
SSL_CLIENT_IC	SSL	The region code of the certificate authority that issued the client's certificate. Example: CA

Table 35. Environment variables that may be set by the HTTP Server (continued)

Variable Name	Type	Description
SSL_CLIENT_IDN	SSL	The distinguished name of the certificate authority that issued the client's certificate.  Example: :cn=testsystem.ibm.com CA,ou=Test Organization Unit,o=System test,l=Rochester,st=Minnesota,c=US
SSL_CLIENT_EMAIL	SSL	The e-mail of the certificate authority that issued the client's certificate.  Example: me@mycompany.com
SSL_CLIENT_IL	SSL	The locality of the certificate authority that issued the client's certificate.  Example: New York
SSL_CLIENT_IO	SSL	The organization name of the certificate authority that issued the client's certificate.  Example: bird watchers
SSL_CLIENT_I OU	SSL	The organizational unit of the certificate authority that issued the client's certificate.  Example: bird watchers
SSL_CLIENT_IPC	SSL	The postal code of the certificate authority that issued the client's certificate.  Example: 55901
SSL_CLIENT_IST	SSL	The state or province of the certificate authority that issued the client's certificate.  Example: MNA
SSL_CLIENT_L	SSL	The locality or city of the client certificate's distinguished name.  Example: New York
SSL_CLIENT_NEWSESSIONID	SSL	If the value is <b>TRUE</b> , it indicates that a full handshake was performed for this SSL session. If the value is <b>FALSE</b> , it indicates that an abbreviated handshake was performed for this SSL session.  Example: True
SSL_CLIENT_O	SSL	The organization name from the client certificate's distinguished name.  Example: bird watchers
SSL_CLIENT_OU	SSL	The organizational unit name from the client certificate's distinguished name.  Example: bird watchers
SSL_CLIENT_PC	SSL	The postal code from the client certificate's distinguished name.  Example: 58401
SSL_CLIENT_SERIALNUM	SSL	The serial number assigned by the issuing certificate authority.  Example: 3F:E4:83:81:02:D5:58
SSL_CLIENT_SESSIONID	SSL	If the value is <b>TRUE</b> , it indicates that a full handshake was performed for this SSL session. If the value is <b>FALSE</b> , it indicates that an abbreviated handshake was performed for this SSL session.  Example: True
SSL_CLIENT_ST	SSL	The state or province from the client certificate's distinguished name.  Example: North Dakota
SSL_PROTOCOL_VERSION	SSL	The SSL protocol version negotiated on the SSL handshake with the client.  Example: SSLV3

Table 35. Environment variables that may be set by the HTTP Server (continued)

Variable Name	Type	Description
SSL_SERVER_C	SSL	The region where the server is located in. Example: Denmark
SSL_SERVER_CN	SSL	The common name from the server certificate's distinguished name. Example: WWW.MYDOMAIN.COM
SSL_SERVER_DN	SSL	The server's distinguished name. Example: :cn=TESTSYSTEM.IBM.COM,ou=MyTestOrganizationUnit, o=Software test, l=Rochester,st=Minnesota,c=US
SSL_SERVER_EMAIL	SSL	The e-mail address of the server certificate. Example: me@mydomain.net
SSL_SERVER_L	SSL	The locality of the server certificate's distinguished name. Example: New York
SSL_SERVER_OU	SSL	The organization unit name from the server certificate's distinguished name. Example: bird watchers
SSL_SERVER_O	SSL	The organization name from the server certificate's distinguished name. Example: bird watchers
SSL_SERVER_ST	SSL	The state or province from the server certificate's distinguished name. Example: North Dakota
HTTP_AS_AUTH_PROFILETKN	SSL, Non-SSL	A 32-bit value used to identify or authenticate the user. See the ProfileToken directive for more information.

**Related information:**

“The CGI Process” on page 182

The basic principle of Common Gateway Interface (CGI) is that a Web server passes client request information to CGI programs in system environment variables (and in some cases through standard input or command line arguments) and all standard output of CGI programs is returned to Web clients.

“CGI APIs” on page 137

This topic provides information about IBM HTTP Server for i APIs for CGI applications.

## Server-side include commands for HTTP Server

This topic provides information about server-side include (SSI) commands for the IBM HTTP Server for i Web server.

HTTP Server SSI commands have the following format:

```
<--#command parameter="value -->
```

**Note:** There is a space before -->.

The following describes the SSI commands for HTTP Server.

### echo

This command prints one of the SSI or API variables. Dates are printed using config timefmt. The attributes are:

**var** Specifies an environment variable name or CGI environment variable name.

See the Environment variables on HTTP Server topic for a list of environment variables.

For example:

```
<!--#echo var="DATE_GMT" -->
```

### **encoding**

Specifies how the server encodes special characters contained in the variable. If set to none, no encoding is done. If set to url, then URL encoding (or %-encoding) is performed. If set to the default of entity, then entity encoding is performed.

For example:

```
<!--#echo encoding="none" -->
```

### **exec**

This command calls a CGI program. The attributes are:

**cgi** Specifies the relative path and file name. For example:

```
<!--#exec cgi="/cgi-bin/counter.pgm" -->
```

### **fsize**

This command prints the size of the specified file according to **config sizefmt**. The attributes are:

**file** Specifies the relative path and file name. For example:

```
<!--#fsize virtual="/include/include.htm" -->
```

### **virtual**

Specifies the relative path and file name using URL encoding. For example:

```
<!--#fsize virtual="/include/include.htm" -->
```

### **lastmod**

This command prints the last modification date of the specified file according to **config timefmt**. The attributes are:

**file** Specifies the relative path and file name. For example:

```
<!--#lastmod file="/include/include.htm" -->
```

### **virtual**

Specifies the relative path and file name using URL encoding. For example:

```
<!--#lastmod virtual="/include/include.htm" -->
```

### **global**

This command is the same as the set command.

### **include**

This command inserts the text of another file. Included files can be nested. The attributes are:

**file** Specifies the relative path and file name. For example:

```
<!--#include file="/include/include.htm" -->
```

### **virtual**

Specifies the relative path and file name using URL encoding. For example:

```
<!--#include virtual="/include/include.htm" -->
```

### **printenv**

This command prints all existing environment variables and their values. There are no attributes. For example:

```
<!--#printenv -->
```

## set

This command sets the value of an environment variable. The attributes are:

**var** Specifies an environment variable name.

See "Environment variables set by HTTP Server" on page 493 for a list of environment variables.

**value** Specifies the value to assign to the environment variable name. For example:

```
<!--#set var="var1" value="yes" -->
```

If you want to insert a special character in a string, precede it with a \. For example:

```
<!--#set var="var1" value="\$Date_GMT" -->
```

## Conditional commands

There are four conditional or flow control commands. The **if** command tests a value. If the value is true, then processing continues with the next line. If the value is not true then processing continues with an **elif**, **else**, or **endif** command. The **elif** and **else** commands are optional. The **if** and **elif** commands have a parameter of **expr**. The **expr** parameter contains the test condition. An **endif** command is required for every if command. For example:

```
<!--#if expr="$USER_AGENT = /MSIE/" -->
<P>You are using Internet Explorer.</P>
<!--#elif expr="$USER_AGENT = /Mozilla/" -->
<P>You are using Netscape.</P>
<!--#else -->
<P>You are not using Internet Explorer or Netscape.</P>
<!--#endif -->
```

The **expr** parameter can have one of the following forms:

Condition	Comments
string	True if the string is not empty
string1 = string2 (equal)	Compare string1 with string2. If string2 has the form /string/, then it is compared as a regular expression. See "Regular expression notation for HTTP Server" on page 491 for more information.
string1 != string2 (not equal)	
string1 < string2 (less than)	
string1 <= string2 (less than or equal to)	
string1 > string2 (greater than)	
string1 >= string2 (greater than or equal to)	
( test_condition )	True if test_condition is true.
!test_condition	True if test_condition is false.
Test_condition1 && test_condition2	True if both test_condition1 and test_condition2 are true.
Test_condition1    test_condition2	True if either test_condition1 or test_condition2 are true.

## Variable substitution

Values can be supplied in the following ways:

- Test can be supplied within a quoted string. For example:

```
<!--#config timefmt="%b%d%y" -->
```

- A literal dollar sign can be supplied in a string using a backslash. For example:  

```
<!--#ifexpr="$a=\$test" -->
```
- A variable reference can be supplied within a character sequence using braces. For example:  

```
<!--#set var="ABC" value="{REMOTE_HOST}_{REQUEST_METHOD}" -->
```

 If REMOTE\_HOST is equal to X and REQUEST\_METHOD is equal to Y, then \$ABC is equal to X\_Y.

## Additional notes

Server-side includes look for the variable, echoes where the variable is found, and proceeds with the function. You can have multiple variable references. When server-side includes encounter a variable reference inside a server-side include directive, it attempts to resolve it on the server side. The following example escapes the & so that server-side includes do not recognize it as a variable. In the second line of the example, the variable "&index" is a server-side variable and is used to construct the variable name "var1". The variable &ecirc; is a client side variable, so the & is escaped to create the value ":fr&ecirc;d" or "fred" with a circumflex over the e.

```
<!--#set var="index" value="1" -->
<!--#set var+"var&index;" value+"fr\&ecirc;d" -->
<!--#echo var="var1" -->
```

The following characters can be escaped. Escape variables must be preceded with a backslash (\).

Escape variable	Meaning
\a	Alert (bell)
\b	Backslash
\f	Form feed (new page)
\n	New line
\r	Carriage return
\t	Vertical tab
\v	Vertical tab
\'	Single quote mark
\''	Double quote mark
\?	Question mark
\\	Backslash
\-	Hyphen
\.	Period
\&	Ampersand

## Time formats for HTTP Server

This topic provides information about time formats for server-side includes for the IBM HTTP Server for i Web server.

The following table contains formatting values used to specify time with server-side includes. See "Log formats for HTTP Server" on page 27 and "Server-side include commands for HTTP Server" on page 502 for proper use of the server-side include time format.

Table 36. Time formats for SSI includes

Value	Description	Example
%%	Replace with %.	%

Table 36. Time formats for SSI includes (continued)

Value	Description	Example
%a	Replace with the abbreviated weekday name.	Mon
%A	Replace with the full weekday name.	Monday
%b	Replace with the abbreviated month name.	Apr
%B	Replace with the full month name.	April
%c	Replace with the date and time.	
%C	Replace with the century number (year divided by 100 and truncated).	
%d	Replace with the day of the month (01-31).	20
%D	Insert the date as %m/%d/%y.	04/20/00
%e	Insert the month of the year as a decimal number (01-12)	03
%E[cCzY]	If the alternative date and time format is not available, the %E descriptions are mapped to their unextended counterparts. For example %E is mapped to %C.	
%Ec	Replace with the alternative data and time representation.	
%EC	Replace with the name of the base year in the alternative representation.	
%Ex	Replace with the alternative data representation.	
%EX	Replace with the alternative time representation.	
%Ey	Replace with the offset from %EC (year only) in the alternative representation.	
%EY	Replace with the full alternative year representation.	
%h	Replace with the abbreviated month name. This is the same as %b.	Apr
%H	Replace with the hour (23-hour clock) as a decimal number (00-23).	22
%I	Replace with the hour (12-hour clock) as a decimal number(00-12).	04
%j	Replace with the day of the year (001-366).	222
%m	Replace with the month (01-12).	04
%M	Replace with the minute (00-59).	24
%n	Replace with a new line.	
%O[deHlMMSUwWy]	If the alternative date and time format is not available the %E descriptors are mapped to their extended counterparts. For example %Od is mapped to %d.	
%Od	Replace with the day of the month using the alternative numeric symbols. Fill as needed with leading zeros if there is any alternative symbol for zero otherwise with leading spaces.	
%Oe	Replace with the day of the month using the alternative numeric symbols filled as needed with leading spaces.	
%OH	Replace with the hour (24 hour clock) using the alternative numeric symbols.	
%OI	Replace with the hour (12 hour clock) using the alternative numeric symbols.	
%Om	Replace with the month using the alternative numeric symbols.	
%OM	Replace with the minutes using the alternative numeric symbols.	
%OS	Replace with the seconds using the alternative numeric symbols.	







Table 36. Time formats for SSI includes (continued)

Value	Description	Example
%0U	Replace with the week number of the year (Sunday as the first day of the week, rules corresponding to %U) using alternative numeric symbols.	
%0w	Replace with the weekday (Sunday=0) using the alternative numeric symbols.	
%0W	Replace with the week number of the year (Monday as the first day of the week) using the alternative numeric symbols.	
%0y	Replace with the year (offset from %C) in the alternative representation and using the alternative numeric symbols.	
%p	Replace with the local equivalent of AM or PM.	
%r	Replace with the string equivalent to %I:%M:%S %p.	
%R	Replace with the time in 24 hour notation (%H:%M).	
%S	Replace with seconds (00-61).	
%t	Replace with a tab.	
%T	Replace with a string equivalent to %H:%M:%S.	16:31:04
%u	Replace with a weekday as a decimal number (1 to 7) with a 1 representing Monday.	3
%U	Replace with the week number of the year (00-53) where Sunday is the first day of the week.	24
%V	Replace with the week number of the year (01-53) where Monday is the first day of the week.	5
%w	Replace with the weekday (0-6) where Sunday is 0.	0
%W	Replace with the week number of the year (00-53) where Monday is the first day of the week.	13
%x	Replace with the appropriate date representation.	
%X	Replace with the appropriate time representation.	
%y	Replace with the year with the century.	02
%Y	Replace with the year with the current century.	2002



## Related information for HTTP Server

IBM Redbooks® publications and Web sites contain information that relates to the IBM HTTP Server for i topic collection. You can view or print any of the PDF files.

### IBM Redbooks

- AS/400 HTTP Server Performance and Capacity Planning 
- IBM HTTP for iSeries: Features of the HTTP Server (original and powered by Apache) 
- IBM HTTP Server (powered by Apache): An Integrated Solution for IBM eServer iSeries Servers 
- Who Knew You Could Do That with RPG IV? A Sorcerer's Guide to System Access and More 


## Web sites

- IBM HTTP Server for i product page 
- Apache HTTP Server Project 

---

## Legal notices for Apache Software Foundation on HTTP Server

The Apache Software Foundation has specific licensing agreements for the ASF Apache Web Server.

- Apache license 

---

## Appendix. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

- | Intellectual Property Licensing
- | Legal and Intellectual Property Law
- | IBM Japan, Ltd.
- | 3-2-12, Roppongi, Minato-ku, Tokyo 106-8711

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

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

Software Interoperability Coordinator, Department YBWA  
3605 Highway 52 N  
Rochester, MN 55901  
U.S.A.

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

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, IBM License Agreement for Machine Code, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

#### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

- | © (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. ©
- | Copyright IBM Corp. \_enter the year or years\_.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

---

## Programming interface information

This IBM HTTP Server for i publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of IBM i.

---

## Trademarks

| IBM, the IBM logo, and [ibm.com](http://ibm.com) are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

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

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

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

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

---

## Terms and conditions

Permissions for the use of these publications is granted subject to the following terms and conditions.

**Personal Use:** You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of these publications, or any portion thereof, without the express consent of IBM.

**Commercial Use:** You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.







Printed in USA