



iSeries

IBM HTTP Server

Version 5 Release 3





iSeries

IBM HTTP Server

Version 5 Release 3

Note

Before using this information and the product it supports, read the information in "Notices," on page 791.

Fourth Edition (August 2005)

This edition applies to version 5, release 3, modification 0 of IBM HTTP Server (product number 5722-DG1) 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 International Business Machines Corporation 1997, 2005. All rights reserved.

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

Contents

IBM HTTP Server for iSeries 1

What's new for IBM HTTP Server for iSeries V5R3	. 1
Printable PDF	. 2
Concepts of functions of HTTP Server	. 2
IBM Web Administration for iSeries	. 2
Fundamental directive, context, and server area concepts on HTTP Server (powered by Apache)	. 7
Content negotiation for HTTP Server (powered by Apache)	. 11
Highly available Web server cluster on HTTP Server	. 17
Real time server statistics	. 21
Web Publishing with the PUT Method	. 25
File compression for HTTP Server (powered by Apache)	. 26
Fast Response Cache Accelerator (FRCA) for HTTP Server (powered by Apache)	. 27
Log formats for HTTP Server (powered by Apache)	. 29
Proxy server types and uses for HTTP Server (powered by Apache)	. 31
Web crawling on HTTP Server	. 33
Webserver search engine on HTTP Server	. 34
Security tips for HTTP Server	. 38
Kerberos for HTTP Server	. 39
User profiles and required authorities for HTTP Server	. 40
Validation list on HTTP Server	. 43
About Tomcat	. 43
Triggered cache manager for HTTP Server	. 46
Trigger messages for triggered cache manager on HTTP Server (powered by Apache)	. 49
WebDAV for HTTP Server (powered by Apache)	. 64
Virtual hosts on HTTP Server	. 65
WebSphere Application Server for iSeries	. 66
WebSphere Portal Express for iSeries	. 67
Scenarios for HTTP Server	. 68
JKL Toy Company creates an HTTP Server (powered by Apache)	. 69
JKL Toy Company adds a new directory to HTTP Server (powered by Apache)	. 71
JKL Toy Company adds user directories for HTTP Server (powered by Apache)	. 74
JKL Toy Company enables cookie tracking on HTTP Server (powered by Apache)	. 78
JKL Toy Company creates virtual hosts on HTTP Server (powered by Apache)	. 81
JKL Toy Company adds password protection for HTTP Server (powered by Apache)	. 84
JKL Toy Company adds dynamic content with server-side includes for HTTP Server (powered by Apache)	. 88
JKL Toy company enables Secure Sockets Layer (SSL) protection on HTTP Server (powered by Apache)	. 91

JKL Toy Company enables single signon for HTTP Server (powered by Apache)	. 96
JKL Toy Company configures an in-process JSP with ASF Tomcat on HTTP Server (powered by Apache)	. 112
JKL Toy Company configures an in-process servlet with ASF Tomcat on HTTP Server (powered by Apache)	. 114
JKL Toy Company configures an in-process WAR file with ASF Tomcat on HTTP Server (powered by Apache)	. 117
JKL Toy Company monitors Web server activity with logs on HTTP Server (powered by Apache)	. 120
Tasks	. 123
Getting started with the IBM Web Administration for iSeries interface	. 123
Install and test the HTTP Server	. 125
HTTP Server tasks	. 127
Compression tasks	. 143
Fast Response Cache Accelerator tasks	. 145
Log and log file tasks	. 147
Proxy tasks	. 151
Search tasks	. 154
Security tasks	. 173
Tomcat tasks	. 178
Triggered cache manager tasks	. 185
WebDAV tasks	. 195
WebSphere tasks	. 196
Virtual host tasks	. 215
Programming	. 218
Application Programming Interface	. 218
Directives for HTTP Server	. 456
Common Gateway Interface	. 723
HTTP Server (powered by Apache) and Apache portable runtime application programming interfaces	. 751
Regular expression notation for HTTP Server	. 752
Set up third party modules for HTTP Server (powered by Apache)	. 754
Handler for HTTP Server (powered by Apache)	. 755
Net.Data programs for the HTTP Server	. 756
Java servlets and JSPs for the HTTP Server (powered by Apache)	. 757
Troubleshoot	. 758
Symptom: Out-of-process ASF Tomcat server does not start or appears to start, but then stops	. 758
Symptom: Error 404 on HTTP Server	. 759
Symptom: HTTP Server has a slow response	. 759
Symptom: Error 500 on HTTP Server	. 759
Symptom: Cannot read or write to QUSRSYS/QATMHINSTC or QUSRSYS/QATMHASFT	. 760
Symptom: HTTP Server on port 80 does not start	. 760
Symptom: Web browser problems with HTTP Server	. 761

Symptom: ADMIN server will not start	762
Symptom: HTTP Server will not start or functions will not work	762
Symptom: Unknown server type when working with HTTP Servers in ADMIN	763
Symptom: All servers show status 'Stopped'	763
Symptom: Cannot access ADMIN or some functions do not work	763
Symptom: User Profile does not have *IOSYSCFG	763
Symptom: Tomcat options not shown in the IBM Web Administration for iSeries interface	763
Symptom: Error 500 when Tomcat settings in the IBM Web Administration for iSeries interface is accessed	763
Symptom: Cannot create new HTTP Server instance	764
Symptom: Net.Data error	764
Symptom: Error occurred opening file	764
Symptom: Databases fail to deploy when configuring with the IBM Web Administration for iSeries interface	764

Symptom: WebSphere Portal authentication performance problems	764
Reference documentation for HTTP Server	765
CL commands for HTTP Server	765
Environment variables on HTTP Server	766
Lotus Domino plug-in for HTTP Server (powered by Apache).	782
Server-side include commands for HTTP Server	782
Supported OS/400 file systems for Web content served by HTTP Server	785
Time formats for HTTP Server.	786
Related information on HTTP Server	788
Legal notices for Apache Software Foundation on HTTP Server	790

Appendix. Notices	791
Trademarks	792
Terms and conditions for downloading and printing information	793

IBM HTTP Server for iSeries

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Welcome to the V5R3 IBM® HTTP Server for iSeries™ documentation. This information describes concepts and tasks related to the V5R3 IBM Web Administration for iSeries interface.

See the IBM HTTP Server for iSeries homepage  for additional product information.

What's new for IBM HTTP Server for iSeries V5R3

This topic highlights the product features that are new in version V5R3 and monthly updates.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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

“IBM Web Administration for iSeries” on page 2

The IBM Web Administration for iSeries interface combines forms, tools, and wizards to create a simplified environment to set up and manage many different servers and server types on your iSeries server.

“Web Publishing with the PUT Method” on page 25

The standard way of uploading files to a Web server using HTTP is through the use of the PUT method. HTTP Server (powered by Apache) 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 (powered by Apache) through the module mod_dav. Another is to provide your own CGI program and configure it for use with HTTP Server (powered by Apache). This article discusses both options, as well as the PUT method in general.

“JKL Toy Company enables single signon for HTTP Server (powered by Apache)” on page 96

This scenario describes how to enable single signon for your HTTP Server (powered by Apache). Follow JKL Toy Company's (a fictitious company) step-by-step examples to enable single signon for your HTTP Server.

“Real time server statistics” on page 21

Real time server statistics provide information on HTTP Server (powered by Apache) performance. Server statistics can be viewed with the **Real Time Server Statistics** tool available through the IBM Web Administration for iSeries interface.

“WebSphere Portal wizard worksheet” on page 198

Use the work sheet to help you quickly set up and run the WebSphere® Portal server. The worksheet provides space for you to enter your own values.

To find other information about what is new or changed this release, see the Memo to Users.

Printable PDF

This topic provides PDF files for this information that you can print or download.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

You can view or download these related topics:


See “Related information on HTTP Server” on page 788 for additional information.

Saving PDF files

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

1. Right-click the PDF in your browser (right-click the link above).
2. Click **Save Target As...**
3. Navigate to the directory in which you would like to save the PDF.
4. Click **Save**.

Downloading Adobe Acrobat Reader

If you need Adobe Acrobat Reader to view or print these PDFs, you can download a copy from the Adobe Web site (www.adobe.com/products/acrobat/readstep.html) .

Concepts of functions of HTTP Server

This topic provides concepts of functions on HTTP Server and IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

IBM Web Administration for iSeries

This topic provides information about the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

The IBM Web Administration for iSeries interface combines forms, tools, and wizards to create a simplified environment to set up and manage many different servers and server types on your iSeries server.

One of the key differences between IBM HTTP Server for iSeries and other Web server products is the graphical user interface (GUI) provided for setting up and managing your servers. The IBM Web Administration for iSeries interface is rich in function, examples, error-checking, and ease-of-use. Administration is made significantly easier through the use of forms, wizards, tasks and tools.

With the HTTP Server for iSeries, 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 IBM Web Administration for iSeries interface enforces supported context for the directives.

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

1. Start a 5250 session on your iSeries server where IBM HTTP Server for iSeries is installed.
2. Type STRTCPSVR SERVER(*HTTP) HTTPSVR(*ADMIN).
3. Start a Web browser.
4. Type `http://[your_iSeries]:2001` in the URL field to start the iSeries Tasks Web page, where [your_iSeries] is the name of your iSeries server.

Example: `http://ibm.com:2001`


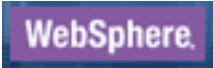


5. Click **IBM HTTP Server for iSeries**.

Note: The IBM Web Administration for iSeries interface can also be started by iSeries Navigator.

Header images

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

Table 1. Header images

Header Image	Description
	Image hyperlink to the iSeries 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 iSeries Web page. This Web page contains additional information on PTFs and support, developer documentation, and other topics.

Tabs and subtabs

Navigation of the IBM Web Administration for iSeries 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 2. 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 IBM Web Administration for iSeries interface. Simple "getting started" tasks and wizards are also available.

Table 2. Main task tabs (continued)

Tab Name	Description
Manage	The <i>Manage</i> tab contains the manage tasks for your servers. The HTTP Server, Application Server, and ASF Tomcat Server subtabs are available under the Manage tab. You can manage all servers, or choose a specific server to manage on your iSeries 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 iSeries server, Internet Users and Groups management, and Search Setup.
Related Links	The <i>Related Links</i> tab contains hyperlinks to useful information related to features, functions, and uses of the IBM Web Administration for iSeries 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 3. 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 iSeries server. This form also provide 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 iSeries server. Use these forms to set up and manage your HTTP Server quickly and easily.
Application Servers	The <i>Application Servers</i> subtab opens forms for managing WebSphere Application Servers, and WebSphere Application Servers - Express, and WebSphere Portal servers currently configured on your iSeries server. Server properties forms, the WebSphere Application Server Administrative Console, and problem determination logs can be viewed here. Note: This subtab is only available if WebSphere Application Server, WebSphere Application Server - Express, or WebSphere Portal is installed.
ASF Tomcat Servers	The <i>ASF Tomcat Servers</i> subtab opens forms for managing the ASF Jakarta Tomcat servers currently configured on your iSeries server. ASF Jakarta Tomcat is an industry standard Java™ Servlet and JavaServer Pages engine. The iSeries implementation of ASF Jakarta Tomcat servlet engine provides a lightweight servlet engine that supports servlets, JavaServer pages (JSPs), and Web application archive (WAR) files. The out-of-process ASF Jakarta Tomcat runs in a separate process (this separate process can be on a different system than the one HTTP Server (powered by Apache) is on) and communicates to the HTTP Server (powered by Apache) through a TCP/IP sockets connection. Running out-of-process provides for more flexibility in the number and type of processes that you can configure.

Table 4. Advanced subtabs

Subtab Name	Description
Settings	The <i>Settings</i> subtab opens forms for managing your global server settings. Global server settings are values that apply to each IBM HTTP Server (powered by Apache) configuration. The values provided here can be overridden individually within each IBM HTTP Server (powered by Apache) configuration file.

Table 4. Advanced subtabs (continued)

Subtab Name	Description
Internet Users and Groups	<p>The <i>Internet Users and Groups</i> subtab opens 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 iSeries 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 iSeries 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>
Search Setup	<p>The <i>Search Setup</i> subtab opens forms for managing your Web server search engine. Use the Web server search engine to set up your web site for full text searches on HTML and text files. Enhance your search results with field support for META tags, a mapping rules file to map internal file names to an external path, and thesaurus support.</p>
TCM	<p>The <i>TCM</i> subtab opens forms for managing your <i>Triggered Cache Manager</i> (TCM) servers. TCM servers may be used in conjunction with Web servers and Web document caching agents to keep Internet (and Intranet) web sites running at peak performance. If your Web site contains dynamically produced web pages, or perhaps rapidly changing static pages, it may benefit you by having a Web document caching agent that is managed by a TCM server.</p> <p>Note: This subtab only appears if you have Option 1 of the DG1 product installed.</p>

Lists

The IBM Web Administration for iSeries 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 5. Lists

List Name	Description
Server	<p>The <i>Server list</i> contains the name of every server currently configured on your iSeries server. This includes HTTP Servers, 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 HTTP Servers is selected, the servers list will show only HTTP Servers, not WebSphere Application Servers.</p>
Server area	<p>The <i>Server area</i> list contains the different container areas, such as <VirtualHost> or <Directory> for the HTTP Server (powered by Apache) configuration file.</p>

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 6. Tasks, Wizards, and Property Forms






Name	Description
Task	<i>Tasks</i> are guided property forms that guide you through advanced configuration steps. Tasks are not a wizard. Tasks group property forms together for advanced configuration tasks.
Wizards	<i>Wizards</i> provide instructional steps that 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 IBM Web Administration for iSeries interface 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 IBM Web Administration for iSeries interface shows you the current status of your servers. The status of the server is displayed with the following icons.







Table 7. Server states

Server State	Description
 Stopped	Stopped. The server is currently stopped. The server is no longer available. The IP address and port number are not in use.
 Running	Running. The server is currently running. The IP address and port number are in use.
 Stopping	Stopping. The server is attempting to stop. the IP address and port number are still in use.
 Creating	Creating. The server is being configured and created. The IP address and port number are not in use.
 Loading...	Loading. The IBM Web Administration for iSeries interface is loading the selected form, wizard, or Web browser frame.

Server buttons

The IBM Web Administration for iSeries 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 8. Buttons

Button	Description
 	Start. Click this button to start the server you are currently working with. The button is gray and unavailable when the server is running.
 	Stop. Click this button to stop the server you are currently working with. The button is gray and unavailable when the server has stopped.
	Restart. Click this button to restart the server you are currently working with. The restart button stops the server and then attempts to restart it.
	Refresh. Click this button to refresh the IBM Web Administration for iSeries interface display. Some changes made by property forms may not be readily displayed. The refresh button clears the IBM Web Administration for iSeries interface display and updates the current server status.

Fundamental directive, context, and server area concepts on HTTP Server (powered by Apache)

This topic provides information on the fundamental directive, context, and server area concepts on HTTP Server (powered by Apache).

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

The HTTP Server 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 (powered by Apache) should wait on to handle incoming requests.

With the HTTP Server (powered by Apache), the directives are extensive, functional, and built around the concept of context.

Directive categories of HTTP Server (powered by Apache)

The HTTP Server (powered by Apache) 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 (powered by Apache). For example, the Listen directive assigns the port the HTTP Server (powered by Apache) should use to handle incoming requests.

Container directives

Used to group directives together within the HTTP Server (powered by Apache). 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 456 for more information on the specific context a directive may or may not be used.

Supported directive context for HTTP Server (powered by Apache)

Understanding the context concept is necessary to increase the productivity and usefulness of your HTTP Server (powered by Apache). The IBM Web Administration for iSeries interface assist in managing context areas of your server. By selecting a different area of the server area, you are changing the context you are managing.

The following are the types of context the HTTP Server (powered by Apache) directives 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 container directive (`<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 456 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.

Container types of HTTP Server (powered by Apache)

The directives used to configure HTTP Server (powered by Apache) 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>

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 (powered by Apache) 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 (<Directory> or <DirectoryMatch>)
File container	directory (<File> or <FileMatch>)
Location container	directory (<Location> or <LocationMatch>)
Proxy container	directory (<Proxy> or <ProxyMatch>)
Virtual host container	virtual host (<VirtualHost>)
Limit except container	<Limit> or <LimitExcept> Note: The context depends on the location of the <Limit> and <LimitExcept> container. It will inherit the context of the area it is in, meaning, if you work with a <Limit> or <LimitExcept> container within a directory container, the <Limit> or <LimitExcept> will be assigned the same values as the <Directory> container in which it is located.

See “Directives for HTTP Server” on page 456 for more information on all the supported HTTP Server (powered by Apache) directives and the context the directive 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 (powered by Apache) 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.

Example:

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

In the above 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.

In reverse, *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.

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.

Content negotiation for HTTP Server (powered by Apache)

This topic provides information on content negotiation, type-map files, MultiViews, negotiation methods, dimensions of negotiation, negotiation algorithm, media types, and wildcards.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 (powered by Apache) 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 (powered by Apache) 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 (powered by Apache) 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 (powered by Apache) 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
```

See the comments in the sample config file for more details.

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 (powered by Apache) 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 (powered by Apache) 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 (powered by Apache) 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 (powered by Apache) 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 (powered by Apache) algorithm** is used in the normal case. The HTTP Server (powered by Apache) algorithm is explained in more detail below. When this algorithm is used, the HTTP Server (powered by Apache) can sometimes 'fiddle' the quality factor of a particular dimension to achieve a better result. The ways the HTTP Server (powered by Apache) 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 (powered by Apache) 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 (powered by Apache) 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 (powered by Apache) sometimes changes the quality values from what would be expected by a strict interpretation of the HTTP Server (powered by Apache) 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 (powered by Apache) 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 en
- 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 (powered by Apache) 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.

Notes on 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
ijkl.html.en	ijkl ijkl.html	-
ijkl.en.html	ijkl	ijkl.html
ijkl.html.en.gz	ijkl ijkl.html	ijkl.gz ijkl.html.gz

Filename	Valid hyperlink	Invalid hyperlink
jkl.en.html.gz	jkl	jkl.html jkl.html.gz jkl.gz
jkl.gz.html.en	jkl jkl.gz jkl.gz.html	jkl.html
jkl.html.gz.en	jkl jkl.html jkl.html.gz	jkl.gz

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 jkl.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, jkl.html.en).

Notes on 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 (powered by Apache) normally marks all responses that are returned after content negotiation as non-cacheable by HTTP/1.0 clients. The HTTP Server (powered by Apache) 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.

Highly available Web server cluster on HTTP Server

This topic provides information on highly available Web server clusters.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 iSeries 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. For more information on clusters, see Clusters in the iSeries Information Center.

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.

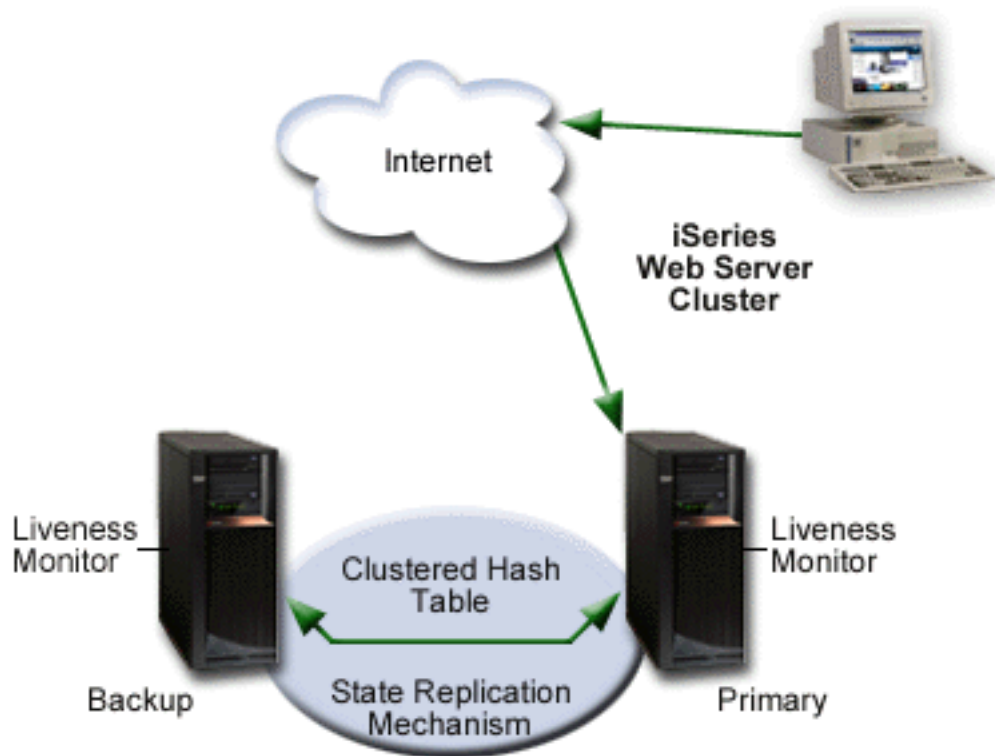
There are three Web server cluster models that are supported:

- Primary/backup with takeover IP model
- Primary/backup with a network dispatcher model
- Peer model

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.



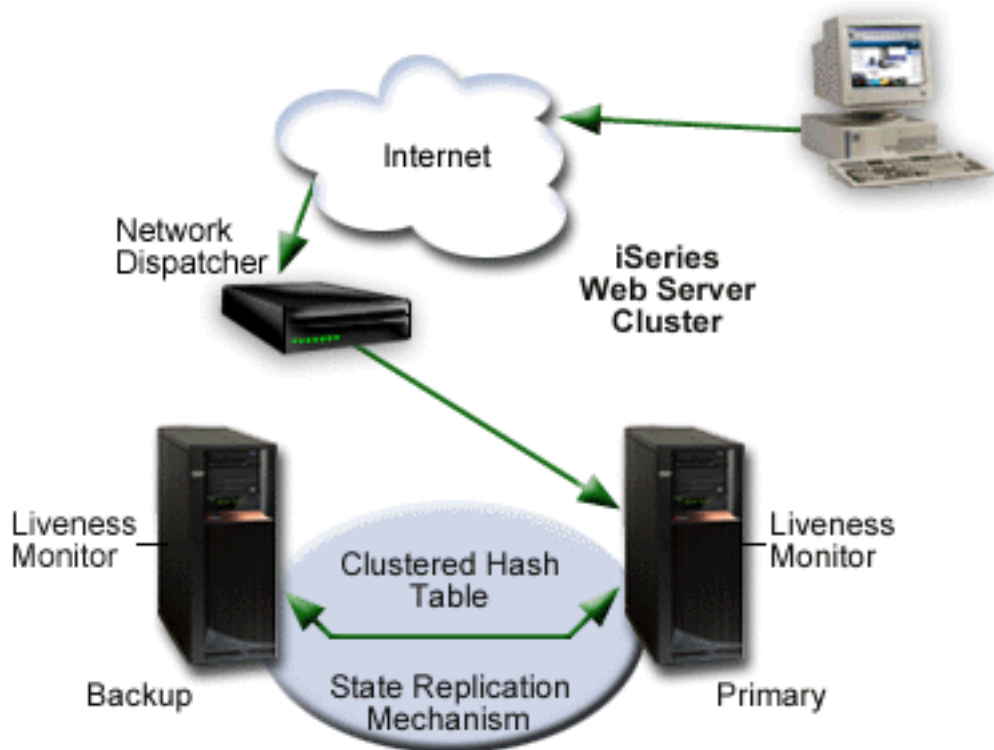
When the primary node fails (failover), or is brought down by the administrator, the failover/switchover process begins. The following steps are performed during failover/switchover:

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 iSeries 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.

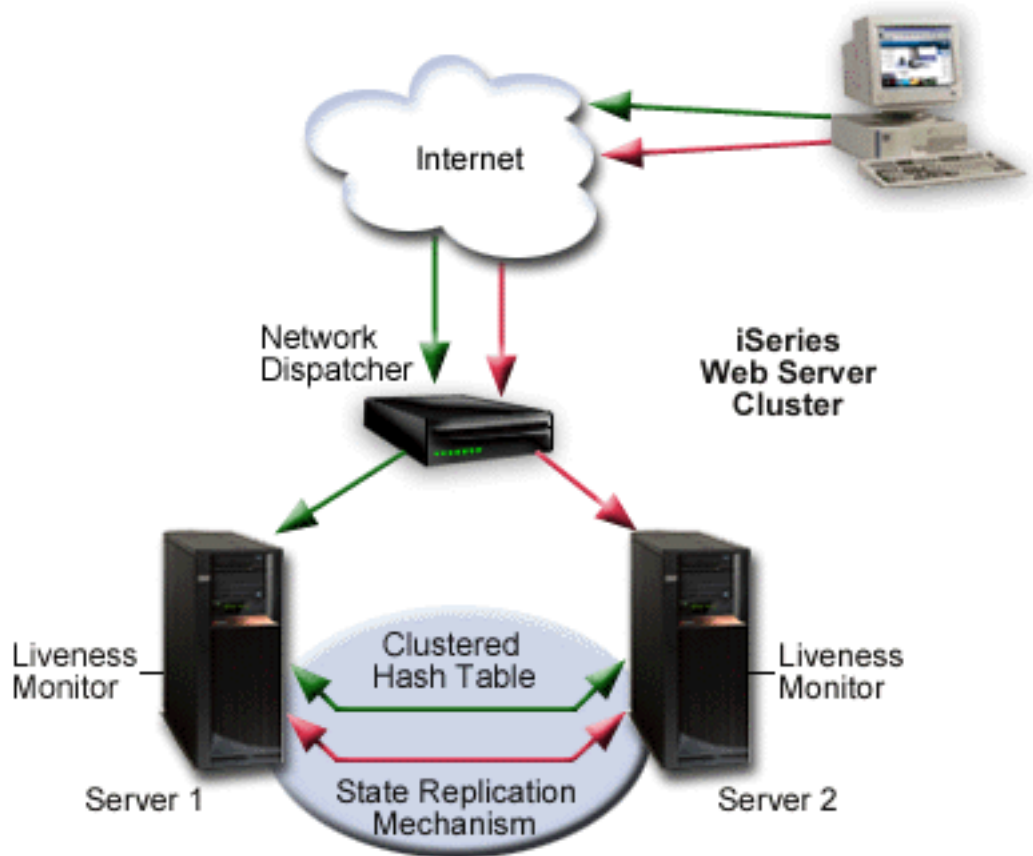
Note: A node can join a recovery domain as primary only if the cluster resource group is in inactive mode.

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.



For more information on Clusters, see Clustering troubleshooting. For instructions on how to set up a highly available Web server, see “Set up and administration of a highly available Web server cluster on HTTP Server (powered by Apache)” on page 130, or Set up a highly available Web server using clustering CL commands for HTTP Server.

Real time server statistics

This topic provides information about the real time server statistics support for HTTP Server (powered by Apache).

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Real time server statistics provide information on HTTP Server (powered by Apache) performance. Server statistics can be viewed with the **Real Time Server Statistics** tool available through the IBM Web Administration for iSeries interface. Only statistics for running HTTP Servers (powered by Apache) 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 Intervals** drop-down menu.

Note: Statistical information is cumulative. If a value is greater than 264⁻¹ 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 (powered by Apache) 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 (powered by Apache) 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 (powered by Apache)" on page 31 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 "Set up CGI programs for HTTP Server (powered by Apache)" on page 744 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 "Set up SSL for the administration (ADMIN) server for HTTP Server" on page 176 for more information.

WebSphere

This column displays the number of completed server transactions that used an associated application server since HTTP Server (powered by Apache) 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 (powered by Apache).

Tomcat

This column displays the number of completed server transactions that used an in-process or

out-of-process ASF Tomcat server since HTTP Server (powered by Apache) was started. See “About Tomcat” on page 43 for more information.

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 (powered by Apache). See “Set up third party modules for HTTP Server (powered by Apache)” on page 754 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 (powered by Apache)” on page 27 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 (powered by Apache)” on page 27 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 to the server.

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. Connections 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 Publishing with the PUT Method

This topic provides information on Web publishing with the PUT method.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

The standard way of uploading files to a Web server using HTTP is through the use of the PUT method. HTTP Server (powered by Apache) 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 (powered by Apache) through the module `mod_dav`. Another is to provide your own CGI program and configure it for use with HTTP Server (powered by Apache). This article 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 (powered by Apache) 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 (powered by Apache) 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 (powered by Apache) 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 (powered by Apache)” on page 64 and “Set up WebDAV for HTTP Server (powered by Apache)” on page 196.

CGI programs

Alternatively, you can provide your own CGI program to handle incoming PUT requests, and configure it for use with HTTP Server (powered by Apache). 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 (powered by Apache) using the Script directive. For more information on the Script directive, see “Module mod_actions” on page 472.


File compression for HTTP Server (powered by Apache)

This topic provides information on file compression and how output from your server is compressed before being sent to the client over the network.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Compressed output is transferred to requesting client browsers at a higher rate of speed than noncompressed output. 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 for HTTP Server (powered by Apache)” on page 578. The DEFLATE filter is always inserted after RESOURCE filters like PHP or SSI. It never touches internal subrequests. See Apache HTTP Server Version 2.0 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 enabling 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 *
```

Fast Response Cache Accelerator (FRCA) for HTTP Server (powered by Apache)

This topic provides information on the Fast Response Cache Accelerator.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Fast Response Cache Acceleration is only available for V5R2 HTTP Server and later releases.

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. You can configure FRCA by using the IBM Web Administration for iSeries interface. See "Set up Fast Response Cache Accelerator (FRCA) for HTTP Server (powered by Apache)" on page 145.

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.

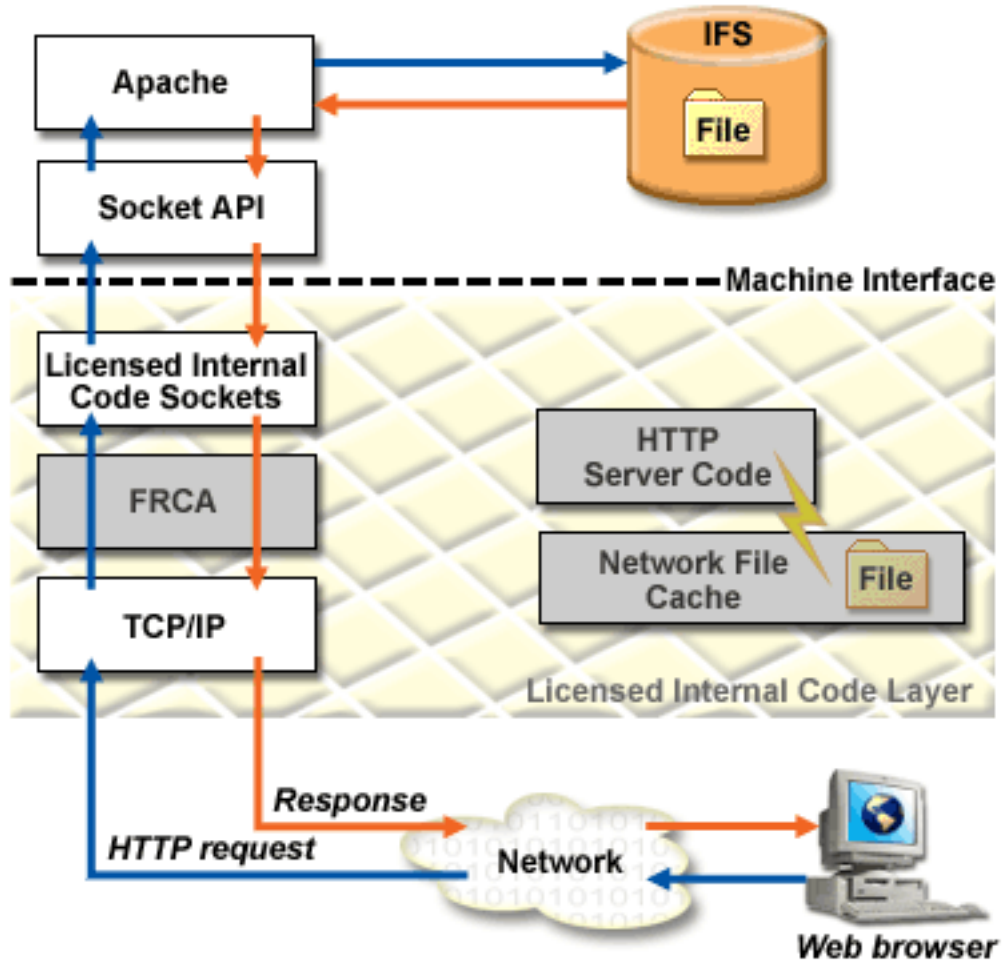
Note: The HTTP Server (powered by Apache) does not check for authorization on content served from FRCA. Use FRCA to cache content that does not need to be secured or accessed through specific validation.

FRCA improves HTTP Server (powered by Apache) performance for both static and dynamic content.

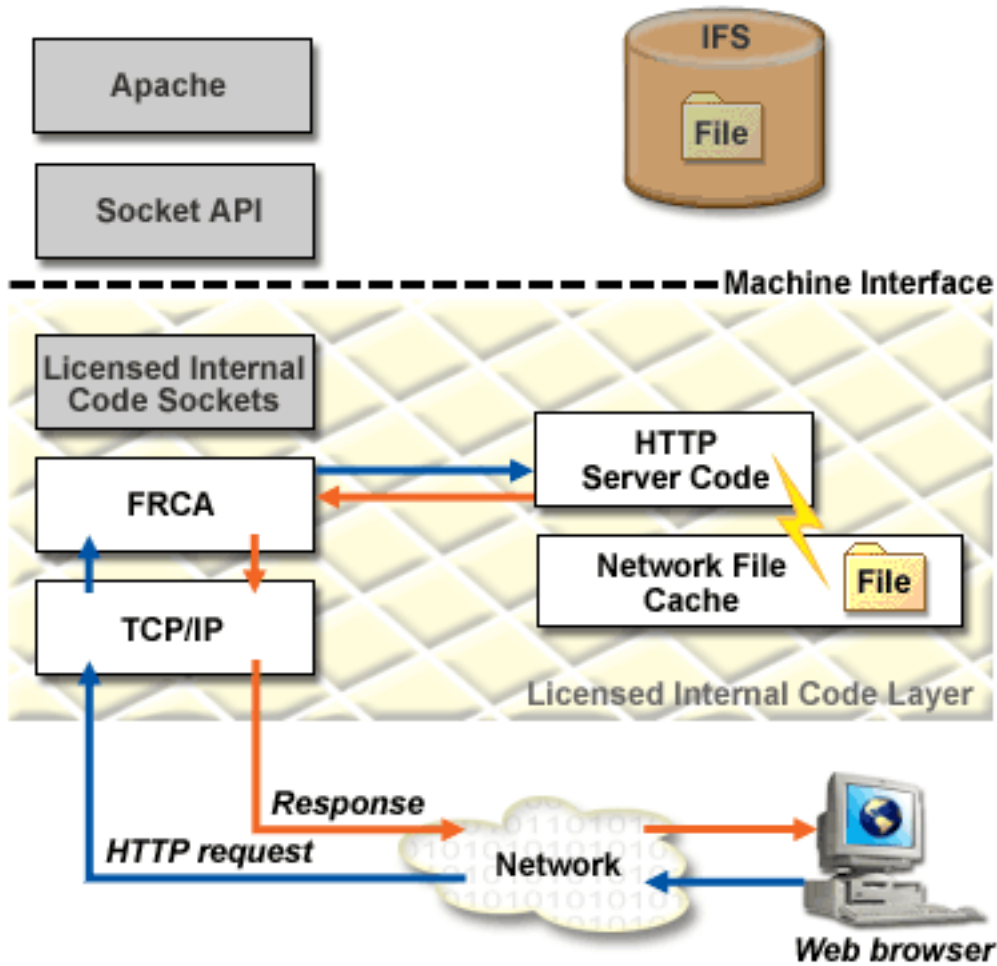
Static content, or content that comes from a file, is stored in Network File Cache and is then served to Licensed Internal Code of HTTP Server (powered by Apache) which essentially 'short circuits' the normal request processing path so the requested information will reach the user faster.

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



Log formats for HTTP Server (powered by Apache)

This topic provides information about log formats and log files.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Use the following information to understand log formats. For information on how to configure logs, see "Set up logs on HTTP Server (powered by Apache)" on page 148.


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 the W3C common log format web site at <http://www.w3.org/Daemon/User/Config/Logging.html> .

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 the W3C extended log format web site at <http://www.w3.org/TR/WD-logfile> .

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 iSeries 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.

Tokens used to 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
%.h	The remote host name or IP address. Example: hal.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=hal
%.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: hal

Token	Description
%. . . 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.

Proxy server types and uses for HTTP Server (powered by Apache)

This topic provides information about proxy server types and uses.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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.

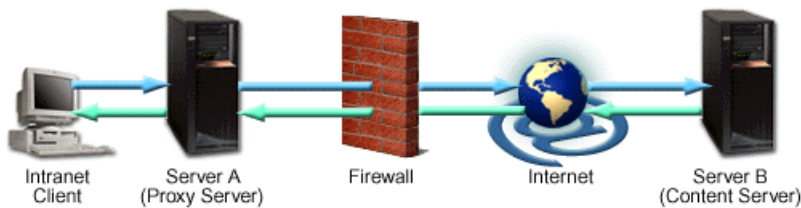
HTTP Server (powered by Apache) has proxy server capabilities built in. Activating these services is simply a matter of configuration. This topic explains three common proxy concepts: forward proxy, reverse proxy, and proxy chaining.

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 proxy 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 above 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).

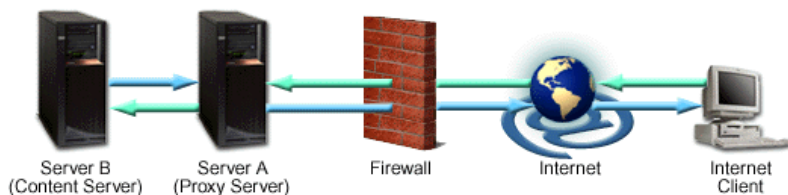


For information on how to configure a forward proxy, see “Set up forward proxy for HTTP Server (powered by Apache)” on page 151.

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

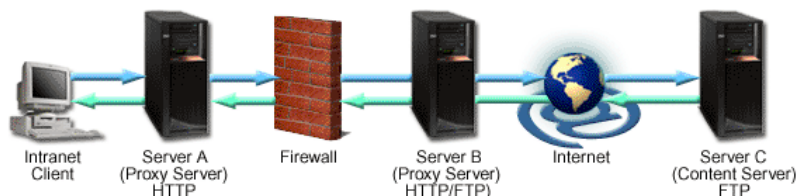
network traffic and load on the content server (proxy caching is optional and not necessary for proxy to function on your HTTP Server). In this example, all information originates from one content server (Server B).

For information on how to configure a reverse proxy, see “Set up reverse proxy for HTTP Server (powered by Apache)” on page 152.

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 (powered by Apache)” on page 153.

Web crawling on HTTP Server

This topic provides information about Web crawling and Web crawlers.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

A Web crawler is a program that finds a URL on another Web server. A "crawl" is the Web crawler program following links within Web pages and downloading HTML and text pages it finds. The Web crawler downloads files to your local directory, and creates a document list. The document list and the files can then be used to create a search index. The search results will link to the actual URL that was found during the crawl.

Attention: The Web crawler downloads text and HTML files to your iSeries. The iSeries checks if sufficient memory is available for a successful Web crawl, but it will not check for available storage.

To crawl a Web site, you must specify attributes such as the document storage directory, the URL to crawl, and so on. Alternately, you may start a crawl using a URL and options object that you have already created using other forms. A URL object contains a list of URLs. An options object contains crawling attributes, such as the proxy server to use for each crawling session.

Some sites cannot be entered without some sort of authentication, such as a userid and password, or certificate authentication. The web crawler has the capacity to handle either case as long as you do the required set up.

For a site requiring a userid and password, you must create a validation list object, entering the URL, userid, and password. See "Set up validation lists for the Webserver search engine on HTTP Server" on page 172 for more information. Then be sure to enter the validation list object when you start crawling. See the digital server certificate information on how to obtain certificate authentication. The digital certificate manager can be used to obtain a new, or register an existing, certificate for any secure server instance of the IBM HTTP Server.

Building a document list by crawling Web sites always runs as a background task and will take several minutes, at a minimum, to run, depending on the maximum time you selected for the session to run, as well as other attributes you have specified.

See "Build the document list by crawling a URL" on page 163 for information on how to use the Web crawler with the IBM Web Administration for iSeries interface.

Webserver search engine on HTTP Server

This topic provides information about the Webserver search engine and national language considerations.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

The Webserver search engine allows you to perform full text searches on HTML and text files. You can control what options are available to the user and how the search results are displayed through customized Net.Data® macros. You can enhance search results by using the thesaurus support. For information on configuring the search engine with the HTTP Server (powered by Apache), see "Set up the Webserver search engine on HTTP Server (powered by Apache)" on page 169.

How it works

Before you can search, you must have an index. The index is a set of files that contain the contents of the documents (in a searchable form) that are to be searched. The search index is used by the search engine rather than searching all of the actual documents.

A search index is created based upon a document list. A document list contains a list of fully qualified path names of all the documents that you want to index.

Documents satisfying a search request are returned by default in their order of ranking. A document's ranking specifies the relevance with respect to the specified search conditions. The following factors determine a document's ranking:

- Frequency of search terms in the document - As the search words appear more frequently in the document, the ranking gets higher.
- Position of search terms in the document - As the search words appear closer to the beginning of the document, the ranking gets higher.
- Frequency of search terms in the whole set of documents - As the search words appear less frequently within the documents in the entire index, the ranking for documents that have search words gets higher.

It is possible that a document with one search term appearing toward the beginning of the document can have a higher ranking than a document with multiple search terms appearing near the end of the document. The search function assumes that words indicating the subject or topic of the document usually appear near the beginning of the document. The highest ranking a document can have is 100%. A document can achieve a ranking of 100% if relatively few of the documents in the index contain the search terms. If many documents in the index contain the search terms, it is likely that none of the documents would achieve a ranking of 100%.

You can provide the following search functions through the customized Net.Data macros:

- Exact search - 100% of the letters match. For example street returns street, Street, and STREET.
- Fuzzy search - 60% of the letters match. For example street returns street, streets, treat, and Tree.
- Wild card search - an asterisk (*) is replaced by zero or more letters and a question mark (?) is replaced by one letter. For example jump* returns jump, jumps, Jumping, and jumper.
- Proximity search - two or more words in the same sentence.
- English word stemming - for example, knife returns knife and knives.
- Case sensitive search - for example, Street returns Street, not street.
- Boolean search (simple) - for example, A and B and C.
- Boolean Search (advanced) - for example, (A and (B or C) not D).
- Document ranking - documents are automatically sorted according to ranking.
- Thesaurus support - finds synonyms or related terms of a search word.
- Search within results - search within returned search results only.
- Simple and Advanced search

You can enhance search results through the use of the thesaurus support. A thesaurus contains words that are synonyms or related terms of a search word. For example, searching for Ping-Pong without thesaurus support results only in documents containing the string Ping-Pong. Using thesaurus support that includes synonyms for Ping-Pong, such as table tennis, results in documents containing either the string Ping-Pong or table tennis.

The URL mapping rules file, built from your selected HTTP Server, is used to set the URL for each document found on a search. It can specify the server port number (or instance) to use and can also map resulting file path names to external path names.

Sample files

Several files are shipped with the product for your use to customize your own Web search function:

File	Description
/QIBM/ProdData/HTTP/Public/HTTPSVR/sample_search.ndm	Sample Net.Data macro that you can customize.
QIBM/ProdData/HTTP/Public/HTTPSVR/thesaurus_sample_search.ndm	Sample Net.Data macro with thesaurus support that you can customize.

File	Description
/QIBM/ProdData/HTTP/Public/HTTPSVR/sample_search.html	Sample search HTML file.
/QIBM/ProdData/HTTP/Public/HTTPSVR/HTML/	Directory of sample HTML files that you can use to build a test search index.
/QIBM/ProdData/HTTP/Public/HTTPSVR/sample_thesaurus.txt	Sample thesaurus definition file.

National language considerations

Documents that you are indexing can be encoded in most ASCII codepages and EBCDIC CCSIDs. Because the search engine does not support all CCSIDs, your documents might be converted to one of the supported CCSIDs during the indexing process. To see the CCSID used to index your documents, view the status of the search index.

Wildcard characters in search strings are not allowed for double byte languages. A wildcard search is implied for double byte languages. Both the name of the index and index directory name must be specified in a single byte characters. The contents of documents are often converted to one of the index CCSIDs listed below.

Documents in languages from the included character sets can all be contained in the same index, as long as the documents are indexed separately. For example, an index can contain English and French documents. Create the index including just the English documents, then update the index with the French documents. If you attempt to index Italian and Russian documents in the same index, an error will occur since the two languages cannot be converted to a common index CCSID. In this case you would need to create two separate indexes. The following table describes the supported CCSIDs for indexes.

Index CCSID	Code page name	Included character sets (CCSIDs)
500	Latin 1	International Albanian, Belgian English, Belgian French, Canadian French MNCS, Danish, Dutch, Dutch MNCS, English International, English US, Finnish, French (France), French MNCS, German (Germany), German MNCS, Icelandic, Italian, Latin 1/Open Systems, Norwegian, Portuguese (Brazil), Portuguese (Portugal), Swedish
838	Thai	Thai
870	Latin 2	Croatian, Czech, Hungarian, Polish, Romanian, Serbian (Latin), Slovak, Slovenia
1025	Cyrillic	Bulgarian, Macedonian, Russian, Serbian (Cyrillic)
1026	Latin 5	Turkish
875	Greek	Greek
424	Hebrew	Hebrew
420	Arabic	Arabic
1112	Baltic	Latvian, Lithuanian
1122	Estonian	Estonian
935	Simplified Chinese (GB)	Simplified Chinese (GB)
1388	Simplified Chinese (GBK)	Simplified Chinese (GBK)
937	Traditional Chinese	Traditional Chinese
5026 (930)	Japanese Katakana	Japanese Katakana
5035 (939)	Japanese Latin	Japanese Latin
1364 (933)	Korean	Korean

Browser and CL command interface for the Webserver search engine and Web crawler

This table shows the browser and CL command interface to all of the search engine and web crawling tasks.

Task	Browser form	CL command	First release available
Create an index	Create search index	CFGHTTPSCH OPTION(*CRTIDX)	V4R4
Update an index	Update search index	CFGHTTPSCH OPTION(*ADDDOC) CFGHTTPSCH OPTION(*RMVDOC)	V4R4
Merge an index	Merge search index	CFGHTTPSCH OPTION(*MRGIDX)	V4R4
Delete an index	Delete search index	CFGHTTPSCH OPTION(*DLTIDX) V4R4 View the status of an index View status of search index: CFGHTTPSCH OPTION(*PRTIDXSTS)	V4R4
View the status of an index	View status of search index	CFGHTTPSCH OPTION(*PRTIDXSTS) See spoolfile QPZHASRCH	V4R4 V5R1
Create a document list	Build a document list	CFGHTTPSCH OPTION(*CRTDOCL) - local	V4R4
Start the web crawler		STRHTTPCRL OPTION(*CRTDOCL) - web crawler	V5R1
Add documents to a document list	Build a document list	CFGHTTPSCH OPTION(*UPDDOCL) Use for local documents. STRHTTPCRL OPTION(*UPDDOCL) Use for documents found with the web crawler.	V4R4 V5R1
Stop a web crawling session.	Work with document list status	ENDHTTPCRL	V5R1
Pause a web crawling session.	Work with document list status	ENDHTTPCRL	V5R1
Resume a web crawling session.	Work with document list status	RSMHTTPCRL	V5R1
Register a document list created before V4R5	Register document list	CFGHTTPSCH OPTION(*REGDOCL)	V5R1
Delete a document list	Delete document list	CFGHTTPSCH OPTION(*DLTDOCL)	V4R4
Display information about a document list	Work with document list status	CFGHTTPSCH OPTION(*PRTDOCLSTS) See spoolfile QPZHASRCH	V4R4 V5R1
Create a URL mapping rules file	Build URL mapping rules file	CFGHTTPSCH OPTION(*CRTMAPF)	V4R4
Append a URL mapping rules file	Build URL mapping rules file	CFGHTTPSCH OPTION(*UPDMAPF)	V4R4
Build a thesaurus dictionary	Build thesaurus dictionary	CFGHTTPSCH OPTION(*CRTTHSDCT)	V4R4
Test a thesaurus dictionary	Test thesaurus dictionary	None.	V5R1

Task	Browser form	CL command	First release available
Retrieve a thesaurus definition from a dictionary	Retrieve thesaurus definition	CFGHTTPSCH OPTION(*RTVTHSDFNF)	V4R4
Delete a thesaurus dictionary	Delete thesaurus dictionary	CFGHTTPSCH OPTION(*DLTTHSDCT)	V4R4
Create a list of URLs to crawl	Build URL object	CFGHTTPSCH OPTION(*CRTURLOBJ)	V5R1
Update a list of URLs to crawl	Build URL object	CFGHTTPSCH OPTION(*UPDURLOBJ)	V5R1
Delete a list of URLs to crawl	Delete URL object	CFGHTTPSCH OPTION(*DLTURLOBJ)	V5R1
Create an object containing crawling attributes	Build options object	CFGHTTPSCH OPTION(*CRTOPTOBJ)	V5R1
Update an object containing crawling attributes	Build options object	CFGHTTPSCH OPTION(*UPDOPTOBJ)	V5R1
Build an object with userid and passwords for authentication	Build validation list	CFGHTTPSCH OPTION(*CRTVLDL)	V5R1
Add userids and passwords for authentication.	Build validation list	CFGHTTPSCH OPTION(*ADDVLDLDTA)	V5R1
Remove userids and passwords for authentication.	Build validation list	CFGHTTPSCH OPTION(*RMVLDLDTA)	V5R1
Search an index	Search index	None	V4R4

Security tips for HTTP Server

This topic provides tips to secure your HTTP Server.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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

- “Permissions on ServerRoot directories for HTTP Server”
- “Stopping users from overriding system wide settings for HTTP Server (powered by Apache)” on page 39
- “Protect server files by default for HTTP Server (powered by Apache)” on page 39
- “Server Side Includes for HTTP Server (powered by Apache)” on page 39
- “CGI for HTTP Server” on page 39

Permissions on ServerRoot directories for HTTP Server

In typical operation, the HTTP Server is started under the iSeries 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 40 for additional information.

Stopping users from overriding system wide settings for HTTP Server (powered by Apache)

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.

Protect server files by default for HTTP Server (powered by Apache)

HTTP Server (powered by Apache) 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 filesystem 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 (powered by Apache)

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.

CGI for HTTP Server

You should consider limiting CGI program execution by using the ScriptAlias directive. This gives you more control over what programs are allowed to run as CGI programs

Always remember that you must trust the writers of the CGI script programs and have the ability to spot potential security holes in CGI, whether they were deliberate or accidental. All of the CGI script programs run as the same user unless you configure them otherwise. When programs run as the same user, they have the potential to conflict with other scripts or programs.

Kerberos for HTTP Server

This topic provides information about Kerberos for use with your HTTP Server (powered by Apache).

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Kerberos is a network authentication protocol designed to provide authentication for client or server applications with secret-key cryptography. The Kerberos server contains an authentication and a ticket

granting function for users who have been authenticated. The HTTP Server does not authenticate; however, it does support the WWW-Authenticate header and uses information received in that header with API calls to an Enterprise Identity Mapping (EIM) server. EIM then determines whether the user is authenticated and if mutual authorization should be initiated using Kerberos. EIM may also be configured to use multiple Kerberos realms (with multiple EIM servers) for authentication processing with users from one realm accessing entities defined in a different realm.

For more information on EIM, see EIM concepts.

Kerberos requirements

- Only supported on OS/400® V5R2 or later.
- Netscape does not support Kerberos; however, Netscape users can access Kerberos protected Websites through a Basic prompt enabled with the Authtype KerberosOrBasic directive.
- Windows® Internet Explorer (IE) does support Kerberos. At the minimum, IE 5.5 with the latest fix pack installed is required.

See the “JKL Toy Company enables single signon for HTTP Server (powered by Apache)” on page 96 scenario for a complete step-by-step instructions on how to enable Kerberos for your iSeries server.

User profiles and required authorities for HTTP Server

This topic provides information about user profiles and required authorities for the HTTP Server.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

User profiles and required authorities for HTTP Server (powered by Apache)

Webmaster user profile

The Webmaster user profile must have read, write, and execute authority to the directory path of the server root directory. This is necessary because the HTTP Administration server swaps to the Webmaster user profile during configuration and administration. 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 HTTP Server.

If there are directories in the path which already exist, the Webmaster user profile must have read, write, and execute authority to those directories prior to executing the **Create New HTTP Server wizard**. Note that directory `www` already exists when the product is shipped. If you plan to use the default server root path of the **Create New HTTP Server wizard** then the authority to directory `www` will need to be changed prior to executing the wizard.

The Webmaster user profile must have the following authorities to perform configuration and administration tasks:

- *IOSYSCFG special authority
- *SERVICE special authority if you plan to use the trace TCP application (TRCTCPAPP) function
- *CHANGE authority to the library object QUSRSYS
- *ALL authority to the following objects:
 - QUSRSYS/QATMHINSTA
 - QUSRSYS/QATMHINSTC
- *USE authority for the following command objects:
 - CRTVLDL

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

If the QPWFSERVER authorization list contains an entry that restricts *PUBLIC access to *EXCLUDE, and one of the authorization list objects is QSYS.LIB, an entry must be created to grant the webmaster profile *CHANGE authority. Use the "DSPAUTL AUTL(QPWFSERVER)" command to display the authorization list. The "ADDAUTLE AUTL(QPWFSERVER) USER(<webmaster>) AUT(*CHANGE)" command can be used to grant the appropriate authority.

Note: Granting *ALLOBJ authority to the Webmaster user profile is not recommended. Using the QSECOFR user profile as the Webmaster user profile is not recommended.

Server user profiles

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 (powered by Apache).

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 (powered by Apache). 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.

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

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

ASF Jakarta Tomcat

- **out-of-process:** The user profile configuring the out-of-process ASF Tomcat is the owner of the configuration files that are created. This user profile must have:
 - *JOBCTL authority
 - *ALL authority to the file QUSRSYS/QATMHASFT
 - *CHANGE authority to the library object QUSRSYS

This configured user profile can, but will not necessarily, have the following directories (with the given authorities) after going through the IBM Web Administration for iSeries interface to create a new ASF Tomcat server.

```
/tomcat_home/conf - execute authority
/tomcat_home/conf/server.xml - read authority
/tomcat_home/webapps - read, write, and execute authority
```

/tomcat_home/webapps/app1 - read and execute authority
 /tomcat_home/webapps/app1/WEB-INF - read and execute authority
 /tomcat_home/webapps/app1/WEB-INF/classes - read and execute authority
 /tomcat_home/webapps/app1/WEB-INF/lib - read and execute authority
 /tomcat_home/webapps/app1/WEB-INF/web.xml - read authority
 /tomcat_home/webapps/app1/*.jsp - read authority
 /tomcat_home/webapps/some_war_file.war - read authority
 /tomcat_home/webapps/ROOT - read and execute authority
 /tomcat_home/work - read, write, and execute authority
 /tomcat_home/logs - read, write, and execute authority
 /tomcat_home/java - execute authority
 /tomcat_home/Java/Java/lib - read and execute authority

In addition the configuration process creates the tomcat_home directory with public execute authority. The default out-of-process tomcat_home directory is /ASFTomcat/tomcat_server_name. If any of these directories existed prior to the ASF Tomcat configuration process, then the previous authorities are left unchanged.

- The user profile used to start the out-of-process ASF Tomcat must have:
 - *USE authority to the file QUSRSYS/QATMHASFT
 - *USE authority to the profile associated with the server user profile (this is QTMHHTTP by default)
 - *IOSYSCFG special authority
- By default the user profile that the out-of-process ASF Tomcat runs under is the QTMHHTTP user profile, but you can configure this to be another user profile.
 This user profile must have *USE authority to the file QUSRSYS/QATMHASFT.
 This user profile must NOT have the following:
 - *SECADM authority
 - *ALLOBJ authority (if the system is at security level 30 or greater).
- **in-process:** in-process ASF Tomcat configurations have the following authority considerations:
 The server user profile (QTMHHTTP) can but will not necessarily have all of the following directories with the given authorities after going through the IBM Web Administration for iSeries interface to create a new ASF Tomcat.

/tomcat_home/conf - execute authority
 /tomcat_home/conf/server.xml - read authority
 /tomcat_home/conf/workers.properties - read authority
 /tomcat_home/webapps - read, write, and execute authority
 /tomcat_home/webapps/app1 - read and execute authority
 /tomcat_home/webapps/app1/WEB-INF - read and execute authority
 /tomcat_home/webapps/app1/WEB-INF/classes - read and execute authority
 /tomcat_home/webapps/app1/WEB-INF/lib - read and execute authority
 /tomcat_home/webapps/app1/WEB-INF/web.xml - read authority
 /tomcat_home/webapps/app1/*.jsp - read authority
 /tomcat_home/webapps/some_war_file.war - read authority
 /tomcat_home/webapps/ROOT - read and execute authority
 /tomcat_home/work - read, write, and execute authority
 /tomcat_home/logs - read, write, and execute authority
 /tomcat_home/Java - execute authority
 /tomcat_home/Java/lib - read and execute authority

- In addition the configuration process creates the tomcat_home directory with public execute authority. The default in-process tomcat_home directory is /www/server_name/.
- When running JSPs on an in-process ASF Tomcat, in order to assure that the Java and .class files resulting from the compilation process of a JSP are owned by the configured profile for that server, the

JSPs should be precompiled by the server administrator under that configured profile. This will assure users swapped to by HTTP Server are not the first to cause the JSP to be compiled and thus become the owners of the Java and Class files that result.

The Java virtual machine (JVM) used to run in-process and out-of-process ASF Tomcat is by default set up to assign Public execute authority to any new IFS directories that are created and Public exclude authority to any new IFS files that are created by Java code running within the JVM.

If any of these directories existed prior to the ASF Tomcat configuration process, then the previous authorities are left unchanged.

See Basic system security and planning for more information on how to work with authorities.

Validation list on HTTP Server

This topic provides information about validation lists for limiting access to your HTTP Server.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 iSeries user profile is never created for the internet users.

A validation list is an AS/400® 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 iSeries 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). Application Programming Interfaces (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.



About Tomcat

This topic provides information about the Apache Software Foundation Jakarta Tomcat servlet engine.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

The iSeries implementation of Apache Software Foundation (ASF) Jakarta Tomcat servlet engine (hereafter referred to as ASF Tomcat) provides a lightweight servlet engine that supports servlets, JavaServer pages (JSPs), and Web application archive (WAR) files. The HTTP Server for iSeries supports a module (mod_jk) that allows communication between the HTTP Server (powered by Apache) and ASF Tomcat.

For more in-depth technical information about ASF Tomcat, see <http://jakarta.apache.org/tomcat/index.html> . For more in-depth technical information about Java servlet technology (including the Java Servlet Specification, V2.2), see <http://java.sun.com/products/servlet/> .

ASF Tomcat can be configured to run:

- In-process where ASF Tomcat and the HTTP Server (powered by Apache) run in the same process and communicate through a Java native interface (JNI). Generally, running in-process is easier to configure and gives better performance. Running in-process also allows you to configure authentication through the HTTP server (including the ability to swap to an authenticated user profile) where out-of-process does not.
- Out-of-process where ASF Tomcat runs in a separate process (this separate process can be on a different computer than the HTTP server) and communicates to the HTTP server through a TCP/IP sockets connection. It is important to note that if SSL support is configured between the Web Browser and the Web server, the connection between the Web server and the out-of-process ASF Tomcat server is not secured with SSL. Running out-of-process allows more flexibility in the number and type of processes that you can configure. An out-of-process configuration allows for:
 - application and functional isolation of information
 - reliability
 - serviceability
 - logically linking organizational synergies to application servlets
 For example, you may have different out-of-process ASF Tomcat servers on different systems to service specific functional areas of an organization, like shipping or accounting.

Configuration and administration of ASF Tomcat on the iSeries is accomplished through the IBM Web Administration for iSeries interface. The ASF Tomcat Basic wizard is provided as an easy way to get started configuring servlets, JSPs, and WAR files.

ASF Tomcat supports the following:

- ASF Tomcat 3.2.4
- Java server pages (JSP) 1.1
- Servlet 2.2 specification
- In-process ASF Tomcat
- Out-of-process ASF Tomcat using mod_jk ajp12 protocol support
- Out-of-process ASF Tomcat using mod_jk ajp13 protocol support
- Making SSL certificates and attributes associated with the client request available to the intended servlet, when configured as in-process or as out-of-process using the ajp13 protocol
- HTTP Server (powered by Apache) virtual host mapping to ASF Tomcat
- JDK 1.2 and 1.3

Directory structure

Directory	Description
tomcat_home	The tomcat_home directory is the base directory for ASF Tomcat. The tomcat_home directory can be located in the root or QOpenSys file systems. For an in-process ASF Tomcat configuration, the default tomcat_home directory is set to the HTTP server directory (/www/server_name/). For an out-of-process ASF Tomcat configuration, the default tomcat-home directory is set to /ASFTomcat/tomcat_server_name/. Within the tomcat_home directory there are subdirectories for logs and configuration information.

Directory	Description
tomcat_home/webapps	This directory contains WAR files if you have them. All WAR files are expanded and subdirectories are added as contexts.
Tomcat_home/webapps/ROOT	This directory is required by ASF Tomcat.
Tomcat_home/webapps/app1	This directory is known as a document base directory. You may have several document base directories under the webapps directory. These represent and map a directory structure to a servlet or JSP application.
Tomcat_home/webapps/app1/WEB-INF	This directory contains the web.xml file for the application.
Tomcat_home/webapps/app1/WEB-INF/classes	This directory contains any Java class files and associated resources that are required for your application. This directory is searched prior to the tomcat_home/webapps/app1/WEB-INF/lib directory for any servlet .class file that is specified in the URL.
Tomcat_home/webapps/app1/WEB-INF/lib	This directory contains any JAR files and associated resources that are required for your application.
Tomcat_home/conf	This directory contains the server.xml and workers.properties configuration files.
Tomcat_home/logs	This directory contains all log files.
Tomcat_home/work	This directory is automatically generated by ASF Tomcat as a place to store intermediate files.
Java/lib	This directory is created as a place to put .jar and Class files that you want to add to the class path.

See “User profiles and required authorities for HTTP Server” on page 40 for information about authority considerations for ASF Tomcat.

Log files

The jk.log file contains messages generated by mod_jk. It is important to note that this file is not erased or regenerated when the ASF Tomcat engine starts. Messages are appended to this file and the size of this file could grow very large if errors are being logged. You should periodically monitor the size of this file and reduce its size. By default, the jk.log is set to the logs directory under server_home (*/www/server_name/logs/*).

Each time the ASF Tomcat servlet engine is started, a set of log files is generated. These log files are all based on the configuration in the server.xml file. The default location of the log files is in the */logs* directory under the tomcat_home directory.

The following is a description of each log file:

Log file	Description
jasper.log	This log file contains messages resulting from trying to start or run JSPs.
servlet.log	This log file contains messages generated as a result of a servlet running in the ASF Tomcat servlet engine. When a servlet is initialized a ServletConfig object is provided to the servlet. Contained within the ServletConfig object is a ServletContext object that provides methods for a servlet to communicate to the Servlet container, in this case Tomcat. On the ServletContext object is a log method that allows Web applications to log to the servlet.log file.
tomcat.log	This log file contains ASF Tomcat servlet engine messages.
jvmstderr.txt	This log file can contain messages from any Java code that does a System.err.println().
jvmstdout.txt	This log file can contain messages from any Java code that does a System.out.println().

Process description

Running in-process means that the ASF Tomcat module (mod_jk) uses the Java invocation API to:

- Set up the JVM
- Attach a given HTTP Server (powered by Apache) thread to the JVM
- Run the servlet in the JVM

The servlet, written in Java, uses the JNI interface to call to the HTTP server to get the headers and data that accompany the HTTP request. The servlet container invokes the target servlet, packages the response, and sends the resulting response headers and data through the HTTP server using the JNI.

Running out-of-process means that the ASF Tomcat module (mod_jk) running in the HTTP server needs to take the headers and accompanying data, package it up into a protocol (ajp12 or ajp13), and then send that information across a TCP socket to the system where the JVM is running. This could be on the same system, or it could be on a different system.

The servlet running in the JVM does the following:

- Receives the request from the socket
- Finds the target servlet
- Invokes the target servlet
- Packages the response headers and data into an ajp12 or ajp13 response
- Sends the response across the TCP socket to the HTTP server

The ASF Tomcat module running in the HTTP Server decodes the ajp12 or ajp13 response to retrieve the HTTP headers and data. The ASF Tomcat module sends the HTTP headers and data to the browser.

Triggered cache manager for HTTP Server

This topic provides information about the triggered cache manager, page assembly, and wrappers.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

The triggered cache manager (TCM) function provides a means to manage cached copies of static and dynamically produced Web pages including those generated by CGI programs, Net.Data, or Java servlets. The triggered cache manager function does not provide a cache, but rather a cache content manager. Web pages are actually cached in such places as local file systems, proxy servers, or caching network routers. When used in conjunction with an application trigger, the triggered cache manager function keeps caches synchronized with the most current data while reducing the frequency of unnecessary dynamic page generation and cache synchronization. See “Trigger messages for triggered cache manager on HTTP Server (powered by Apache)” on page 49 for more information.

These Web pages can consist of a single object or they can consist of several objects (or page fragments). Dynamically produced Web pages that embed page fragments are updated anytime an embedded page fragment changes.

For example, several Web pages are constructed by a Java servlet using data extracted from a DB2[®] database. These Web pages are in turn cached in several caching network routers. Web pages are served from the network router caches rather than from the HTTP server running the Java servlet. When the database data is changed, an application sends a trigger message to the triggered cache manager function. Through the use of object dependency graphs, the triggered cache manager function determines which Web pages need to be updated. The triggered cache manager function then requests new versions of each affected Web page from the Java servlet and updates the cached copies in the caching network routers.

In this example, the Web site performance is greatly increased by serving cached copies of dynamically produced Web pages rather than serving newly generated Web pages for each request. The Web site consistency and performance is again increased by using an application trigger, with the triggered cache manager function, to update cached Web pages only when underlying data changes. This avoids unnecessary and costly cache updates and synchronization.

The triggered cache manager function is most effective for a Web site that has a large number of requests for content that is somewhat constant, but with variables that change frequently. An example of this might be a Web site that serves an on-line catalog that contains price and inventory information (the product information is static, but the price and inventory information changes frequently). One of IBM's first uses of the triggered cache manager function was to drive the 1996 Winter Olympic Games Web site.

Page assembly

Often, advanced Web sites contain information that appears on more than one page. If the information changes, all of the pages with that information need to be updated. There are several potentially difficult problems associated with this:

- It can be extremely difficult to find and update all affected pages, especially as a Web site grows in complexity.
- Information with a tendency to change also has a tendency to be expensive to maintain; for example, database activity might be required to effect an update.

If a Web site's pages can be composed from partial HTML fragments. Each fragment is unique and any page that contains its information acquires it by embedding the fragment. This can lead to more flexible, diverse, and complex Web sites.

The trigger cache manager function provides a way to assemble Web pages from a set of fragments. If a fragment changes, the author of the fragment needs only to publish that fragment; the triggered cache manager finds all the affected pages (data sources), rebuilds them, and copies the updated pages to the configured delivery locations (cache targets).

To take advantage of this facility, HTML segments must indicate which other fragments are to be embedded. This is done with a simple tag very similar to Server Side Includes (SSI). These tags are used for two purposes:

- To determine the dependency relationships among HTML fragments (dependency parsing).
- To physically construct pages from the fragments (page assembly).

A publish trigger handler is used to accomplish this task.

The tag used to specify that a fragment is to be included is specified in HTML as follows. The keyword *%fragment* is chosen to avoid conflicts with SSI.

```
<!-- %fragment (/source-name, /default-name) -->
```

Notes:

- The *source-name* is the name of the embedded fragment, relative to the data source specified in the server configuration. The data source is searched in this order:
 1. Among the objects triggered.
 2. Among objects in the assembled directory within the repository of published objects maintained by the triggered cache manager function. These are objects that might have been fetched as a result of previous triggers and correspond to the assembled versions of fragments, intermediate results, and final publishable pages.

If the data source is a file system, the source-name is a file name. If the data source is HTTP, the source-name is the file name portion of a URL.

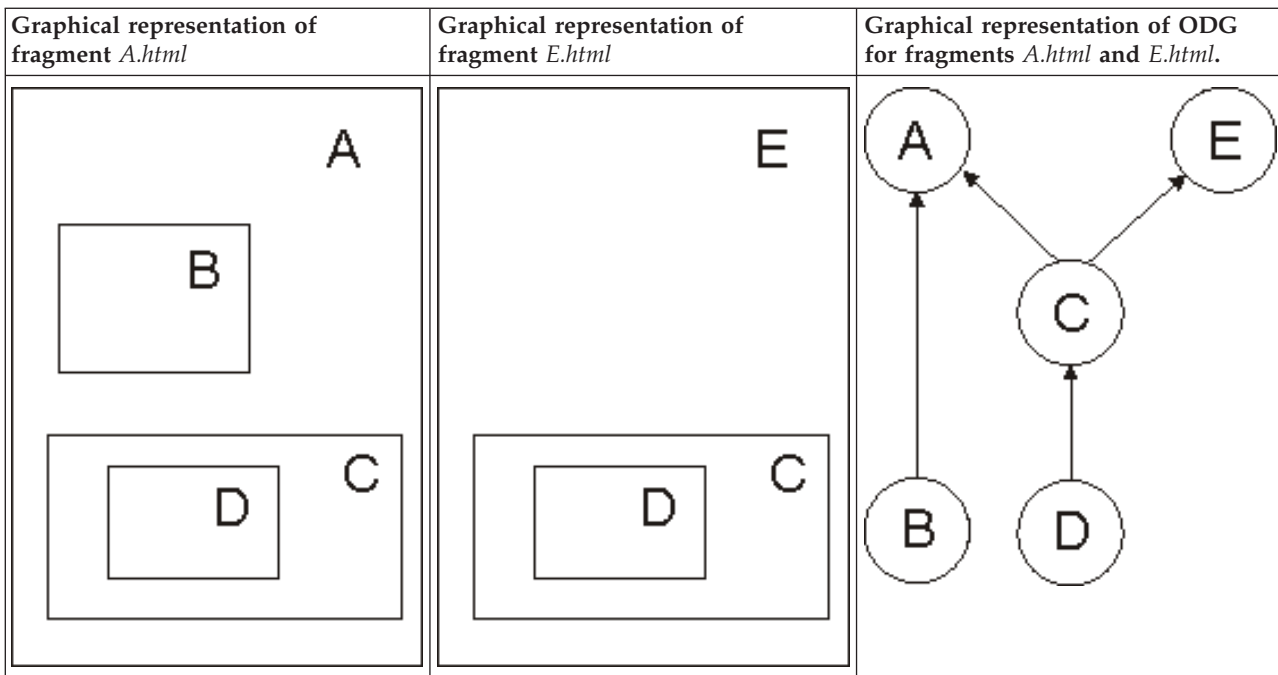
- The *default-name* is the name of another fragment that might be used as a default when a specified fragment cannot be found.
- Nesting is supported. The triggered cache manager function uses the object dependency graph (ODG) specified in the server configuration to start the page assembler in the correct order to build pages.

For example, when one fragment, *fragment A.html*, embeds another fragment, *fragment B.html*, it is said that *fragment A.html* is dependent on *fragment B.html*. This is denoted as either $B \rightarrow A$ or as the ordered pair (B, A) . It is said that there is a dependency relationship between B and A . Suppose that A also embeds another fragment, *fragment C.html*, and that *fragment C.html* in turn embeds *fragment D.html*.

Suppose further that some other fragment, *fragment E.html*, also embeds *fragment C.html*. This relationship can be represented as a directed graph called the object dependency graph (ODG). The object dependency graph is for *fragment A.html* and *fragment C.html*. The HTML fragments that describe this look something like the following:

<i>Fragment A.html</i>	<i>Fragment C.html</i>	<i>Fragment E.html</i>
<pre><html> ... <!-- %fragment(B.html) --> <!-- %fragment(C.html) --> ... </html></pre>	<pre><html> ... <!-- %fragment(D.html) --> ... </html></pre>	<pre><html> ... <!-- %fragment(C.html) --> ... </html></pre>

When the triggered cache manager function is instructed to publish *fragment C.html*, it determines that *C.html* embeds *D.html*. Similarly, when *A.html* and *E.html* are published, it determines $B \rightarrow A$, $C \rightarrow A$, and $C \rightarrow E$. These relationships are automatically entered into the object dependency graph when the fragments are processed by the publish trigger handler. If *D.html* is republished at a later time, the publish application can determine, by examining the ODG, that all of *C.html*, *A.html*, and *E.html* must be rebuilt but that the old copy of *B.html* can be used. Similarly, if only *B.html* changes, none of *C.html*, *D.html*, or *E.html* are affected. The entire process of discovering dependencies, updating the object dependency graph, and composing the correct and only the correct pages is all automatically performed by the publish trigger handler.



Wrappers

In some cases, most notably when HTML is generated automatically from HTML editors, it might be undesirable to use the full HTML content in a fragment. Such applications might insist on producing `<html>` and `<body>` tags, for example, which make the fragment nearly unusable. To manage this, fragments are also parsed for wrappers.

A wrapper directive begins with the following tag:

```
<!-- %begin-fragment( name ) -->
```

A wrapper directive ends with the following tag:

```
<!-- %end-fragment -->
```

When these are encountered within a document, all text before the initial tag and following the final tag is discarded. Only the text between these tags is actually used during page composition. The selected text can be given a name other than the name of the file in which it occurs. This permits multiple fragments to exist in the same file. That is, if multiple *%begin-fragment* and *%end-fragment* tags are found, the file is treated as multiple files for the purpose of composing the page and managing the object dependency graph.

Trigger messages for triggered cache manager on HTTP Server (powered by Apache)

This topic provides information about trigger requests and trigger messages.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Trigger requests

Communication with the triggered cache manager function is done using HTTP 1.0 protocol. The HTTP POST method is used to send trigger messages to the triggered cache manager function (unless otherwise noted). The Content-Length header is required on all POST requests. The header Content-Type may be set to `application/x-trigger-request`, but is not required. All other request headers are ignored.

The contents of a POST request should not be URL encoded. Each line of the POST request describes an independent trigger message. A single logical operation is associated with each trigger message. Any line that begins with a pound sign (#) is treated as a comment line and is ignored.

Trigger messages are issued by writing a program to submit them using the HTTP 1.0 protocol. You should write a program to issue a POST similar to the following:

```
POST /handler/ HTTP/1.0
Content-Type: application/x-trigger-request
Content-length: nn
-id trigid1 -qp A -update -from /item1.html -to /item1.html
```

Where:

- *handler* is the name of the trigger handler that should process your trigger message. The Admin trigger name is `admin`. The ODG-Admin trigger handler name is `odg-admin`. Update Cache trigger handler and Publish trigger handler names are defined by you in your configuration.
- *nn* is the actual length of your content.

Trigger messages

The following tasks (organized by trigger handler) are supported using trigger messages. Trigger messages are plain text printable strings. Trigger messages generally consist of keywords and associated values. Keywords may be abbreviated to the shortest unambiguous string. Optional letters are specified within square brackets. For example, `-fo[rce]` indicates that any of the following are valid: `-fo`, `-for`, `-forc`, or `-force`.

- Admin trigger handler
 - “Terminate the triggered cache manager function (Admin trigger handler)”
 - “Query request queues (Admin trigger handler)” on page 51
 - “Start logging (Admin trigger handler)” on page 51
 - “Stop logging (Admin trigger handler)” on page 52
 - “Roll log (Admin trigger handler)” on page 52
 - “Purge request (Admin trigger handler)” on page 52
 - “Query trigger messages (Admin trigger handler)” on page 53
 - “Enable and disable cache targets (Admin trigger handler)” on page 54
 - “Enable and disable acknowledgement targets (Admin trigger handler)” on page 54
 - “Change cache target priority (Admin trigger handler)” on page 55
- Update cache trigger handler
 - “Update cache target object (Update Cache trigger handlers)” on page 55
 - “Delete cache target object (Update Cache trigger handlers)” on page 56
- Publish trigger handlers
 - “Update cache target object (Publish trigger handlers)” on page 56
- ODG-Admin trigger handler
 - “Dump the ODG to disk (ODG-Admin trigger handler)” on page 56
 - “Add an object to the ODG (ODG-Admin trigger handler)” on page 57
 - “Delete an object from the ODG (ODG-Admin trigger handler)” on page 57
 - “Add an edge to the ODG (ODG-Admin trigger handler)” on page 58
 - “Delete an edge from the ODG (ODG-Admin trigger handler)” on page 58
 - “Query orphan objects (ODG-Admin trigger handler)” on page 59
 - “Query dependencies (ODG-Admin trigger handler)” on page 59
 - “Query dependents (ODG-Admin trigger handler)” on page 59
 - “Query chain (ODG-Admin trigger handler)” on page 60
 - “Retreive object from the ODG (ODG-Admin trigger handler)” on page 60

Terminate the triggered cache manager function (Admin trigger handler): Use this trigger message to stop the triggered cache manager function. Terminating the triggered cache manager function may result in significant delays during restart.

Syntax

`-id trigid -term[inate]`

Keywords

Keyword	Description
<code>-id</code>	Optional. Any printable string and is used for identification in the logs and within messages. If not specified, an internally generated unique ID is used.
<code>-term[inate]</code>	No value.

Response

See “Trigger responses” on page 61.

Query request queues (Admin trigger handler): Use this trigger message to monitor the status of request queues.

Syntax

`-id trigid -q[ueues]`

Keywords

Keyword	Description
<code>-id</code>	Optional. Any printable string and is used for identification in the logs and within messages. If not specified, an internally generated unique ID is used.
<code>-qu[ueues]</code>	No value.

Responses

This trigger message returns one line per request queue, where each line has the following format:

```
1140 id internalid admin ! description-name: active=nn queued=mm lifetime-total=tt
lifetime-failed=ff lifetime-retried=rr threads=numt
```

Response	Description
1140	The message ID 1140, response to query request queues.
id	The trigger ID from the query or the internally generated trigger ID if <code>-id</code> was omitted from the query.
internalid	The internally generated trigger id.
description-name	The name of the configuration description whose request queue this line pertains to.
nn	The number of requests in this request queue that are currently processing.
mm	The number of requests in this request queue that are waiting to be processed.
tt	The number of total requests processed by this request queue since the triggered cache manager function was last started.
ff	The total number of requests which were processed and failed by this request queue since the triggered cache manager function was last started.
rr	The total number of requests which were processed and retried by this request queue since the triggered cache manager function was last started.
numt	The number of threads used to handle this request queue. This number remains constant unless the triggered cache manager function is restarted and the configuration was changed to specify more threads. For example: 1140 6 6 admin ! publishapp: active=5 queued=7 lifetime-total=98770 lifetime-failed=0 lifetime-retried=0 threads=10

See “Trigger responses” on page 61.

Start logging (Admin trigger handler): Use this trigger message to start logging messages to the configured log.

Syntax

-id trigid -startlog

Keywords

Keyword	Description
-id	Optional. Any printable string and is used for identification in the logs and within messages. If not specified, an internally generated unique ID is used.
-startlog	No value.

Response

See "Trigger responses" on page 61.

Stop logging (Admin trigger handler): Use this trigger message to stop logging messages to the log.

Syntax

-id trigid -stoplog

Keywords

Keyword	Description
-id	Optional. Any printable string and is used for identification in the logs and within messages. If not specified, an internally generated unique ID is used.
-stoplog	No value.

Response

See "Trigger responses" on page 61.

Roll log (Admin trigger handler): Use this trigger message to roll over the log. The current log file is closed, .old is appended to the file name, and a new file is started.

Syntax

-id trigid -rolllog

Keywords

Keyword	Description
-id	Optional. Any printable string and is used for identification in the logs and within messages. If not specified, an internally generated unique ID is used.
-rolllog	No value.

Response

See "Trigger responses" on page 61.

Purge request (Admin trigger handler): Use this trigger message to purge an asynchronous request that has not yet completed. Note that it may not be possible to immediately purge a request if the request is

in a state which must complete before it can be removed from the system. When a purge trigger message arrives, the specified request is marked purged and it is deleted from the system at the first available synchronization point.

Syntax

```
-id trigid -purge triggerid
```

Keywords

Keyword	Description
-id	Optional. Any printable string and is used for identification in the logs and within messages. If not specified, an internally generated unique ID is used.
-purge	The internal ID assigned to the request. The internal ID is returned to the requester as part of the response when the trigger message is accepted and can also be found in the logs.

Response

See “Trigger responses” on page 61.

Query trigger messages (Admin trigger handler): Use these trigger messages to find the status of all trigger messages or a particular trigger message. You can query all trigger messages or query a specific trigger message.

Syntax

```
-id trigid -qa[11]
```

```
-id trigid -qt[rigger] triggerid
```

Keywords

Keyword	Description
-id	Optional. Any printable string and is used for identification in the logs and within messages. If not specified, an internally generated unique id is used.
-qa[11]	No value. Queries all trigger messages.
-qt[rigger]	The internal ID assigned to the request. The internal ID is returned to the requester as part of the response when the trigger message is accepted and can also be found in the logs.

Responses

The following message is returned as one line for each trigger message, where each line has the following format:

```
1151 id internalid admin ! trigger-id trigger-internalid handler state queue-policy collapsed-id purged
```

Response	Description
1151	The message ID 1151, response to a query trigger message.
id	The trigger ID from the query or the internally generated trigger ID if -id was omitted from the query.

Response	Description
internalid	The internally generated trigger id.
trigger-id	The trigger ID from the trigger message.
handler	The name of the trigger handler.
state	The state of the trigger message.
queue-policy	The queue policy specified on the trigger message. See "Queue policy" on page 61.
collapsed-id	The internal identifier of the trigger message into which this request has been collapsed, if the request has been collapsed.
purged	The word purged, if the request has been purged. For example: 1151 DaedalusCt1 212 admin ! trg65 180 updates Collapsed A 160 purged.

See "Trigger responses" on page 61.

Enable and disable cache targets (Admin trigger handler): Any cache target can be enabled or disabled without shutting down the triggered cache manager function. If a cache target is disabled, no data is sent to the cache target, and all other actions proceed as normal.

Syntax

```
-id trigid -chsi[nk] target {e[nable]|d[isable]}
```

Keywords

Keyword	Description
-id	Optional. Any printable string and is used for identification in the logs and within messages. If not specified, an internally generated unique id is used.
-chsi[nk]	The name of the cache target to enable or disable.
{e[nable] d[isable]}	Specify enable to enable the cache target. Specify disable to disable the cache target.

Response

See "Trigger responses" on page 61.

Enable and disable acknowledgement targets (Admin trigger handler): Any acknowledgement target can be enabled or disabled without shutting down the triggered cache manager function. If an acknowledgement target is disabled, no requests are sent to the acknowledgement target, and the system acts as if the request was successfully sent.

Syntax

```
-id trigid -chac[k] acknowledgement {e[nable]|d[isable]}
```

Keywords

Keyword	Description
-id	Optional. Any printable string and is used for identification in the logs and within messages. If not specified, an internally generated unique id is used.
-chac[k]	The name of the acknowledgement to enable or disable.

Keyword	Description
{e[nable] d[isable]}	Specify enable to enable the acknowledgement. Specify disable to disable the acknowledgement.

See “Trigger responses” on page 61.

Change cache target priority (Admin trigger handler): The cache target priority associated with an Update Cache trigger handler can be changed without shutting down the triggered cache manager function.

Syntax

```
id trigid -chsp[riority] handler num
```

Keywords

Keyword	Description
-id	Optional. Any printable string and is used for identification in the logs and within messages. If not specified, an internally generated unique id is used.
-chsp[riority]	The name of the trigger handler and the new value for the cache target priority.

Response

See “Trigger responses” on page 61.

Update cache target object (Update Cache trigger handlers): Use these messages to copy a single object (optionally renaming the object) or copy a list of objects from a data source to a cache target.

Syntax

```
-id trigid -qp[olicy] queue-policy -ob[jects] object-list
```

```
-id trigid -qp[olicy] queue-policy -up[date] -fr[om] source-d -to target-id
```

Keywords

Keyword	Description
-id	Optional. Any printable string and is used for identification in the logs and within messages. If not specified, an internally generated unique ID is used.
-qp[olicy]	Optional. See “Queue policy” on page 61.
-ob[jects]	Specifies a list of objects delimited by a space. Paths are relative to the cache target directory defined in your configuration. For example: -ob item1.html item2.html /dir3/item3.html.
-up[date]	No value.
-fr[om]	Required if -update is specified. Any printable string that specifies the name of the source object.
-to	Optional, defaults to the value specified by -to. Any printable string that specifies the name of the cache target object.

Response

See “Trigger responses” on page 61.

Delete cache target object (Update Cache trigger handlers): Use these messages to delete cache target objects.

Syntax

```
-id trigid -qp[olicy] queue-policy -de[lete] object
```

Keywords

Keyword	Description
-id	Optional. Any printable string and is used for identification in the logs and within messages. If not specified, an internally generated unique ID is used.
-qp[olicy]	Optional. See “Queue policy” on page 61.
-de[lete]	Specifies the object to delete.

Response

See “Trigger responses” on page 61.

Update cache target object (Publish trigger handlers): Use this message to move data from one or more data sources, construct pages, and write constructed pages to one or more targets. This message inputs a set of object names. The objects are copied from a data source and all other objects that have been defined as dependent upon the input objects are added to this set. All objects are then assembled and copied to the cache target. Object dependencies are identified through an object dependency graph (ODG).

Syntax

```
-id trigid -qp[olicy] queue-policy -ob[jects] itemlist
```

Keywords

Keyword	Description
-id	Optional. Any printable string and is used for identification in the logs and within messages. If not specified, an internally generated unique ID is used.
-qp[olicy]	Optional. See “Queue policy” on page 61.
-ob[jects]	Specifies a list of objects delimited by a space. Paths are relative to the cache target directory defined in your configuration. For example: -ob item1.html item2.html /dir3/item3.html.

Response

See “Trigger responses” on page 61.

Dump the ODG to disk (ODG-Admin trigger handler): Use this message to write the current state of the ODG into a text format. A file named snapshot.log is created in the directory containing the specified ODG.

Syntax

-id trigid -dos[napshot]

Keywords

Keyword	Description
-id	Optional. Any printable string and is used for identification in the logs and within messages. If not specified, an internally generated unique ID is used.
-dos[napshot]	No value.

Response

See “Trigger responses” on page 61.

Add an object to the ODG (ODG-Admin trigger handler): Use this message to add an object to the ODG.

Syntax

-id trigid -ao[bject] objectid

Keywords

Keyword	Description
-id	Optional. Any printable string and is used for identification in the logs and within messages. If not specified, an internally generated unique ID is used.
-ao[bject]	Specifies the name of the object to add.

Response

See “Trigger responses” on page 61.

Delete an object from the ODG (ODG-Admin trigger handler): Use these messages to delete an object from the ODG.

Syntax

-id trigid -dob[ject] objectid

-id trigid -dob[ject] objectid -fo[rce]

-id trigid -dob[ject] objectid -dor[phans]

Keywords

Keyword	Description
-id	Optional. Any printable string and is used for identification in the logs and within messages. If not specified, an internally generated unique ID is used.
-dob[ject]	Specifies the name of the object to delete.
-fo[rce]	Optional. No value. Specifies that all edges connected to the object are removed and the object is removed.

Keyword	Description
-dor[phans]	Optional. No value. Specifies that all edges connected to the object are removed, the object is removed, and orphaned objects are removed.

Response

See “Trigger responses” on page 61.

Add an edge to the ODG (ODG-Admin trigger handler): Use these messages to add an edge to the ODG.

Syntax

-id trigid -ae[dge] -fr[om] from-object -to to-object -ed[getype] composition

-id trigid -ae[dge] -fr[om] from-object -to to-object -ed[getype] composition -fo[rce]

Keywords

Keyword	Description
-id	Optional. Any printable string and is used for identification in the logs and within messages. If not specified, an internally generated unique ID is used.
-ae[dge]	No value.
-fr[om]	Specifies the file name of the independent object.
-to	Specifies the file name of the dependent object.
-ed[getype]	Must be composition.
-fo[rce]	Optional. No value. Specifies that any necessary endpoints are added and then the edge is added.

Response

See “Trigger responses” on page 61.

Delete an edge from the ODG (ODG-Admin trigger handler): Use these messages to delete an edge from the ODG.

Syntax

-id trigid -de[dge] -fr[om] from-object -to to-object -ed[getype] composition

-id trigid -de[dge] -fr[om] from-object -to to-object -ed[getype] composition -dor[phans]

Keywords

Keyword	Description
-id	Optional. Any printable string and is used for identification in the logs and within messages. If not specified, an internally generated unique ID is used.
-de[dge]	No value.
-fr[om]	Specifies the file name of the independent object.
-to	Specifies the file name of the dependent object.

Keyword	Description
-ed[getype]	Must be composition.
-dor[rphans]	Optional. No value. Specifies that the edge is removed and orphaned endpoints are removed.

Response

See “Trigger responses” on page 61.

Query orphan objects (ODG-Admin trigger handler): Use this message to retrieve a list of all orphaned objects.

Syntax

-id trigid -qo[rphans]

Keywords

Keyword	Description
-id	Optional. Any printable string and is used for identification in the logs and within messages. If not specified, an internally generated unique ID is used.
-qo[rphans]	No value.

Response

See “Trigger responses” on page 61.

Query dependencies (ODG-Admin trigger handler): Use this message to query an object for a list of all objects that this object is dependent upon.

Syntax

-id trigid -qdependen[cies] object -ed[dgetype] composition

Keywords

Keyword	Description
-id	Optional. Any printable string and is used for identification in the logs and within messages. If not specified, an internally generated unique ID is used.
-qdependen[cies]	Specifies the name of the object to query.
-ed[dgetype]	Must be composition.

Response

See “Trigger responses” on page 61.

Query dependents (ODG-Admin trigger handler): Use this message to query an object for a list of all objects that are dependent upon this object.

Syntax

-id trigid -qdependen[ts] object -ed[dgetype] composition

Keywords

Keyword	Description
-id	Optional. Any printable string and is used for identification in the logs and within messages. If not specified, an internally generated unique ID is used.
-qdependen[ts]	Specifies the name of the object to query.
-ed[dgetype]	Must be composition.

Response

See “Trigger responses” on page 61.

Query chain (ODG-Admin trigger handler): Use this message to return the list of all dependencies of the objects in the object list. This query returns a list of all the objects that might be affected if all of the objects in the object list were triggered to a Publish trigger handler.

Syntax

```
-id trigid -qc[hain] list -ed[dgetype] composition
```

Keywords

Keyword	Description
-id	Optional. Any printable string and is used for identification in the logs and within messages. If not specified, an internally generated unique ID is used.
-qc[hain]	Specifies a list of objects delimited by a space.
-ed[dgetype]	Must be composition.

Response

See “Trigger responses” on page 61.

Retrieve object from the ODG (ODG-Admin trigger handler): Use a GET request with the following URL to retrieve an object out of the object dependency graph (ODG) repository.

Syntax

```
/odg-name/{source|assembled}/object-name
```

Keywords

Keyword	Description
odg-name	The name of the ODG description in the configuration.
{source assembled}	Specify source to indicate that the original data should be returned. Specify assembled to indicate that the assembled version of the object should be returned.
object-name	The name of the object.

Response

The Content-Length: response header record is returned. If the specified URL is not valid, status code 404 is returned. A URL may not be valid for any of the following reasons:

- The ODG name specified is not a valid ODG.
- The second directory name is neither source or assembled.
- The object name does not exist in the specified ODG.

See “Trigger responses.”

Queue policy

Some trigger messages may specify a queuing policy. A queuing policy refers to how asynchronous trigger messages are sequenced through the request queues. There are three different queuing policies: A, S, and P. Trigger messages queued under policy A (asynchronous) are processed independently of each other and generally in parallel of each other. Trigger messages queued under policy S or P (sequential or parallel) are logically grouped into a dependent group consisting of at least one S trigger message and zero or more P trigger messages. For example, trigger messages may arrive in groups which can be visualized in either of the two following ways:

```
trailing-S group) P P P P P P P S P P P P S P P S  
(leading -S group) S P P P P P P P S P P P P P S P P P
```

Sets of consecutive P trigger messages are processed in parallel, but while any type P trigger message is being processed, no type S trigger message may be started. Once a type S trigger message has started, no type P trigger message may start. For any queue, there is at most one type S trigger message processed at any time. For any queue, no type S trigger message is being processed if a type P trigger message is being processed. Type A trigger messages are processed regardless of the current state of type S or P trigger messages.

Note that because the request queues are associated with trigger handler descriptions, you may have multiple S and P streams processing in parallel, so long as the incoming trigger messages are directed to different trigger handlers.

Trigger responses

Responses are sent when a trigger message is received and as an acknowledgement message. The format of the response is:

```
sss text  
Content-Length: nn  
Content-Type: application/x-trigger-msglist
```

```
mmmm requestors-requestid internal-requestid handlerid ! data\r\n
```

Where:

- *sss* is one of the defined status codes. See “Trigger status and message codes” on page 62.
- *text* is the message text associated with the status code.
- *nn* is the length of the content.
- *mmmm* is one on the defined message codes. See “Trigger status and message codes” on page 62.
- *requestors-requestid* is the trigger ID supplied by the caller. If no trigger ID is supplied with the trigger message, the internally generated ID is supplied.
- *internal-requestid* is the internally generated trigger id. This value is guaranteed to be unique.
- *handlerid* is the name of the trigger handler used to process this request.
- *data* is text relevant to the message.

Some handlers process trigger messages asynchronously. The initial response to asynchronous trigger messages is to indicate a status code of 202. Final status is provided through an acknowledgement message sent to an acknowledgement target.

Trigger status and message codes

The general status codes may be sent in response to any trigger message. Handler specific status codes are listed for status codes that have a more specific meaning.

General status codes

Value	Meaning	Example of cause or use
200	The request was completed successfully	Queries
202	The request has been accepted for processing. There may be an asynchronous response later.	Publish trigger handlers
400	Client error, handler specific. The returned text body may contain more information.	Malformed POST
404	Cannot find URL requested.	Handler does not exist
500	Server error, handler specific. The returned text body may contain more information.	Publish errors
501	Specified HTTP command not implemented.	HEAD, PUT

Admin trigger handler status codes

Value	Meaning
202	Success. The command response is in the body.
400	Failure. The request syntax is wrong.

Update cache trigger handler status codes

Value	Meaning
202	Success. The trigger message is queued for processing.
400	Failure. The request syntax is wrong.

Publish trigger handler status codes

Value	Meaning
202	Success. The operation is queued for processing.
400	Failure. The request syntax is wrong.

ODG-Admin trigger handler status codes

Value	Meaning
200	Success. The operation is complete, with possible warnings.
400	Failure. The requested syntax is wrong.

Trigger message codes

Code	Meaning
1101	{0} (Successful acknowledgement of queued trigger message. The data consists of the list of objects that are rebuilt)
1102	{0} request is queued (The trigger message was successfully queued)
1104	Server terminated
1105	Log roll-over successful
1106	Logging has been enabled
1107	Logging has been disabled
1108	Request "{0}" will be purged
1109	Specified object "{0}" has been deleted from ODG "{1}"
1110	Object "{0}" defined in ODG "{1}"
1111	Edge "{0}" to "{1}" was deleted from ODG "{2}"
1113	Edge "{0}" to "{1}" was added in ODG "{2}"
1114	Collapsed requests {0} will also be purged
1115	Server will terminate after active asynchronous request have completed
1120	Snapshot for {0} successful
1140	{0}: active={1} queued={2} lifetime-total={3} lifetime-failed={4} lifetime-retried={5} threads={6}
1141	Lifetime total server requests={0}
1150	No active requests.
1151	{0} {1} {2} {3} {4} {5}
1152	Version: {0} Started: {1}
1160	{0} {1} {2}
1161	{0}
1162	{0}
1170	= {0} "{1}" has been changed
1171	{0} "{1}" has been changed to have cache target priority {2}
2102	A value for the "{0}" flag was specified and will be ignored
2103	Changed "{0}" to "{1}" because all names specified on the command line must be absolute
2104	Error saving request to disk, request will not be recovered over restart
2105	Error rolling log
2106	Logging already enabled
2107	Logging already disabled
2108	Duplicate object "{0}" will be ignored
2110	Server is in the process of being shutdown. This request will not be executed until the server is restarted.
2115	No value found for the "{0}" flag. The flag has been ignored
2116	Value "{0}" for "{1}" keyword will be ignored
2117	Specification of the "{0}" keyword was done twice, "{1}" ignored

Code	Meaning
9011	Error reading "{0}" from data source specified in description "{1}" {2}
9012	Error writing "{0}" to cache target specified in description "{1}" {2}
9013	Error sending acknowledgment to acknowledgment specified in description "{0}" {1}
9014	Error erasing "{0}" from cache target specified in description "{1}" {2}
9102	Error assembling "{0}" {1}
9103	Error parsing "{0}" {1}
9106	Error creating log file: {0}
9108	Could not delete "{0}" from ODG "{1}": {2}
9110	Could not delete edge "{0}" to "{1}" from ODG "{2}": {3}
9112	Could not add edge "{0}" to "{1}" in ODG "{2}": {3}
9114	Invalid keyword "{0}" found, request rejected
9115	Required flag "{0}" was not specified
9116	One of the flags "{0}" must be specified
9117	Invalid edgetype "{0}" specified, request rejected
9118	Both keywords "{0}" and "{1}" are specified, but are mutually exclusive
9122	Exception taking snapshot: {0}
9126	Two arguments for the "{0}" flag must be specified
9127	One argument for the "{0}" flag must be specified
9128	Specified ODG "{0}" does not allow updates via API commands
9129	Specified ODG "{0}" does not exist
9130	Object "{0}" does not exist in ODG "{1}"
9131	ODG cycle detected, some objects in the chain: {0}
9132	Invalid edgetype "{0}" returned by dependency parser, verify configuration
9135	Leftover pages cannot be built - check for duplicates of: {0}
9138	Error writing "{0}" to repository in "{1}" {2}
9140	Request was purged before completion.
9141	{0} "{1}" does not exist
9142	{0} "{1}" is invalid, must be integer
9143	{0} "{1}" does not support enable/disable
9150	ODG Fatal Exception Processing Request
9151	Internal error {0}

WebDAV for HTTP Server (powered by Apache)

This topic provides information about Web-based distributed authoring and versioning (WebDAV).

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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.

See “Set up WebDAV for HTTP Server (powered by Apache)” on page 196 for information on how to configure WebDAV.

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	Get’s 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.

Virtual hosts on HTTP Server

This topic provides information about virtual host types.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 (powered by Apache) virtual host types

There are three variations of virtual hosts on HTTP Server (powered by Apache):

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.

For information on how to configure a name-based virtual host see “JKL Toy Company creates virtual hosts on HTTP Server (powered by Apache)” on page 81.

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`.

For information on how to configure dynamic virtual hosts on HTTP Server (powered by Apache) see “Set up virtual hosts on HTTP Server (powered by Apache)” on page 215.

WebSphere Application Server for iSeries

This topic provides information about the WebSphere Application Server for iSeries.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

The IBM WebSphere Application Server for iSeries is a Web application server for interactive, Web-based applications to be used with your HTTP Server (powered by Apache). The Web application server supports Java servlets, JavaServer Pages (JSP) files, resource adapter (connector) implementations, and other Web components.

In addition, the IBM Telephone Directory is a business application is provided with WebSphere Application Server - Express that delivers the ability to search, view, and manage entries in a directory. For example, the IBM Telephone Directory application can be used to provide access to online accounting information, personnel records, and inventory. In a new or existing directory, you can use the application to create and manage entries.

See the WebSphere Application Server - Express or the WebSphere Application Server for iSeries topics for information on:

- **Installation** - This topic provides instructions for installing the WebSphere Application Server products
- **Administration** - This topic describes how to configure and administer WebSphere Application Server products.
- **Application development** - This topic is a reference for developing applications for WebSphere Application Server.
- **Security** - This topic describes the security features for WebSphere Application Server, and how to use them.
- **Troubleshooting** - This topic provides information to help you determine the cause of problems with WebSphere Application Server.

See the IBM Telephone Directory topic for information on how to install, configure, and use the IBM Telephone Directory application.

WebSphere Portal Express for iSeries

This topic provides information about WebSphere Portal Express for iSeries.

WebSphere Portal for Multiplatforms on iSeries is one of the industry's most comprehensive portal offerings. It contains a wide range of portal technologies that help you develop and maintain first-class business-to-consumer (B2C), business-to-business (B2B), and business-to-employee (B2E) portals. The IBM Web Administration for iSeries interface provides an easy to use wizard to set up a WebSphere Portal Server. The WebSphere Portal wizard helps you through the necessary steps to create and configure servers, deploy the Portal EAR files, set up security, databases, LDAP, and perform any other necessary set up tasks to get WebSphere Portal up and running.

Create a New WebSphere Portal Wizard

Important: WebSphere Portal Express for iSeries is supported on OS/400 V5R2 or later only.

The IBM Web Administration for iSeries interface can create and configure your WebSphere Portal for you using the **Create a New WebSphere Portal Wizard**. The wizard will perform the following tasks for you:

- Create and configure a new WebSphere Portal server
- Create and configure a new WebSphere Application Server
- Create and configure a new HTTP Server (powered by Apache)
- Configure DB2 for WebSphere Portal
- Configure LDAP authentication for WebSphere Portal
- Configure Portal URIs
- Deploy portlets

See "WebSphere Portal wizard worksheet" on page 198 for a list of prerequisites and a step-by-step walkthrough for the wizard.

One Step WebSphere Portal Create Wizard


In addition, the IBM Web Administration for iSeries interface can create a production ready portal server without security in one easy step with the **Create WebSphere Portal - One Step Wizard**. The wizard will select appropriate default values based on information gathered from your system and a recommended a WebSphere Portal configuration. The wizard will perform the following tasks for you:

- Create and configure a new WebSphere Portal server
- Create and configure a new WebSphere Application Server
- Create and configure new HTTP Server (powered by Apache)
- Configure DB2 for Portal

Note: This wizard does not configure security. For security configuration, use the **Create a New WebSphere Portal Wizard** .

Install the WebSphere Portal Configuration Wizard

- Download and apply the latest group PTFs to your iSeries. See “Requirements” on page 199 in the “WebSphere Portal wizard worksheet” on page 198 topic for a list of required PTFs, software, and hardware.
- After you have applied the Group PTFs and the individual PTFs, ensure that the QHTTPSVR subsystem is started by typing Work Active Job (WRKACTJOB) and looking for the QHTTPSVR subsystem. If the subsystem has not been started, start it now with the following command: STRSBS QHTTPSVR/QHTTPSVR
- You must end the HTTP admin instance under the QHTTPSVR subsystem (if it is currently running) by using the following command: ENDTCPSVR SERVER(*HTTP) HTTPSVR(*ADMIN)
- Start the HTTP admin instance by using the following command: STRTCPSVR SERVER(*HTTP) HTTPSVR(*ADMIN)
- To access the WebSphere Portal wizard, type the URL `http://systemName:2001` (where *systemName* is the fully qualified domain name of your iSeries system) into a web browser.
- Log into the HTTP admin instance and click **IBM Web Administration for iSeries**.
- Click **Create WebSphere Portal** to start the wizard.

The iSeries installation and configuration documentation for WebSphere Portal - Express and WebSphere Portal - Express Plus for Mutliplatform is located in the WebSphere Portal Product Documentation  Web site.

Scenarios for HTTP Server

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

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 IBM Web Administration for iSeries 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 IBM Web Administration for iSeries 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://[iSeries_name]:[port]`

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

`http://jklserver:2001`

JKL Toy Company creates an HTTP Server (powered by Apache)

This scenario discusses how to create an HTTP Server (powered by Apache).

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Scenario

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

Prerequisites

- It is assumed you have read “Scenarios for HTTP Server” on page 68.

Start the IBM Web Administration for iSeries interface

Note: Enter your Webmaster user profile username and password when prompted.

1. Start a Web browser.
2. Enter **http://[iSeries_hostname]:2001** in the location or URL field .
Example: http://jkl_server:2001

Note: If you have changed your port number for the IBM Web Administration for iSeries interface, replace port 2001 with your port number.

3. Click **IBM HTTP Server for iSeries**.

Note: If the IBM Web Administration for iSeries interface does not start, see “Install and test the HTTP Server” on page 125.

Create your HTTP Server (powered by Apache)

The IBM Web Administration for iSeries 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. Select **HTTP server (powered by Apache) - recommended**.
5. Click **Next**.
6. Enter a descriptive, unique name in the **Server name** field.
Example: JKLTEST
7. Click **Next**.
8. Accept the default value.
Example: /www/jkltest
9. Click **Next**.
10. Accept the default value.
Example: /www/jkltest/htdocs
11. Click **Next**.

12. Accept the default values or replace with your own unique IP address and port.
Example: IP address All Addresses
Example: Port 1975
13. Click **Next**.
14. **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.
15. Click **Next**.
16. Accept the default values to specify the length of time to keep the log files or update with your preferences.
17. Click **Next**.
18. Review the displayed information. If any information is incorrect, click **Back** and correct it.
19. Click **Finish** to create your new HTTP Server (powered by Apache).

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

Restart your HTTP Server (powered by Apache)

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 (powered by Apache) does not start, see “Troubleshoot” on page 758.

Test your HTTP Server (powered by Apache)

1. Open a new Web browser.
2. Enter **http://[iSeries_hostname]:[port]** in the location or URL field .
Example: http://jkl_server:1975

Your new HTTP Server (powered by Apache) will display a generic HTML file provided by the IBM Web Administration for iSeries interface.

View your HTTP Server (powered by Apache) 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 (powered by Apache) 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
<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 (powered by Apache)

This scenario discusses how to add a new directory.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 for HTTP Server” on page 68.
- It is assumed you have read and completed “JKL Toy Company creates an HTTP Server (powered by Apache)” on page 69 or you have an existing HTTP Server (powered by Apache) configuration.

Start the IBM Web Administration for iSeries interface

Note: Enter your Webmaster user profile username and password when prompted.

1. Start a Web browser.
2. Enter **http://[iSeries_hostname]:2001** in the location or URL field .
Example: `http://jkl_server:2001`

Note: If you have changed your port number for the IBM Web Administration for iSeries interface, replace port 2001 with your port number.

3. Click **IBM HTTP Server for iSeries**.

Note: If the IBM Web Administration for iSeries interface does not start, see “Install and test the HTTP Server” on page 125.

Create a new directory

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server (powered by Apache) from the **Server** list.
Example: JKLTEST
4. Select **Global configuration** from the **Server** 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 (powered by Apache)

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 (powered by Apache) does not start, see “Troubleshoot” on page 758.

Test your HTTP Server (powered by Apache)

1. Open a new Web browser.
2. Enter `http://[iSeries_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 iSeries interface.

View your HTTP Server (powered by Apache) 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 (powered by Apache) 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 (powered by Apache)

This scenario discusses how to add a user directory.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 (powered by Apache). 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. The new user profile will be called **SJONES** on the iSeries and the new user directory will be located at **/home/sjones**.

Prerequisites

- It is assumed you have read “Scenarios for HTTP Server” on page 68.
- It is assumed you have read and completed “JKL Toy Company creates an HTTP Server (powered by Apache)” on page 69 or you have an existing HTTP Server (powered by Apache) configuration.
- It is assumed you have read and completed “JKL Toy Company adds a new directory to HTTP Server (powered by Apache)” on page 71.
- It is assumed you have installed and are familiar with iSeries Navigator.
- It is assumed you have read “User profiles and required authorities for HTTP Server” on page 40.

Create a new user profile with iSeries Navigator

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

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

1. Start **iSeries Navigator**.
2. Expand the iSeries 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 iSeries.
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 iSeries Navigator

Note: The `/home` directory comes preinstalled on your iSeries.

1. Start **iSeries Navigator**.
2. Expand the iSeries 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 iSeries Navigator

The new user directory does not contain any files. Use the iSeries Navigator to copy **index.html**, found in `/www/[server_name]/htdocs` directory of your HTTP Server (powered by Apache), to your new user directory.

Example: `/www/jkltest/htdocs`

1. Start **iSeries Navigator**.
2. Expand the iSeries 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 (powered by Apache) will look for when this directory is requested by the Web browser.

Start the IBM Web Administration for iSeries interface

Note: Enter your Webmaster user profile username and password when prompted.

1. Start a Web browser.
2. Enter **http://[iSeries_hostname]:2001** in the location or URL field .
Example: `http://jkl_server:2001`

Note: If you have changed your port number for the IBM Web Administration for iSeries interface, replace port 2001 with your port number.

3. Click **IBM HTTP Server for iSeries**.

Note: If the IBM Web Administration for iSeries interface does not start, see “Install and test the HTTP Server” on page 125.

Set up user directories for HTTP Server (powered by Apache)

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server (powered by Apache) 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 from 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 (powered by Apache) 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 (powered by Apache)

After creating the user directory, you must set up your HTTP Server (powered by Apache) 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 (powered by Apache)

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 (powered by Apache) does not start, see “Troubleshoot” on page 758.

Test your HTTP Server (powered by Apache)

1. Open a new Web browser.
2. Enter `http://[iSeries_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 (powered by Apache) 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 (powered by Apache) 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 (powered by Apache)

This scenario discusses how to enable cookie tracking for your HTTP Server (powered by Apache).

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 for HTTP Server" on page 68.
- It is assumed you have read and completed "JKL Toy Company creates an HTTP Server (powered by Apache)" on page 69 or you have an existing HTTP Server (powered by Apache) configuration.

Start the IBM Web Administration for iSeries interface

Note: Enter your Webmaster user profile username and password when prompted.

1. Start a Web browser.
2. Enter **http://[iSeries_hostname]:2001** in the location or URL field .
Example: `http://jkl_server:2001`

Note: If you have changed your port number for the IBM Web Administration for iSeries interface, replace port 2001 with your port number.

3. Click **IBM HTTP Server for iSeries**.

Note: If the IBM Web Administration for iSeries interface does not start, see “Install and test the HTTP Server” on page 125.

Create a cookie for HTTP Server (powered by Apache)

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server (powered by Apache) 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 (powered by Apache)

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 (powered by Apache)

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 (powered by Apache) does not start, see “Troubleshoot” on page 758.

Test your HTTP Server (powered by Apache)

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://[iSeries_hostname]:[port]** in the location or URL field.

Example: http://jkl_server:1975

View your HTTP Server (powered by Apache) 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 (powered by Apache) 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 JKLC00KIE
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 (powered by Apache)

This scenario discusses how to create virtual hosts.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 (powered by Apache) **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 for HTTP Server” on page 68.
- It is assumed you have read and completed “JKL Toy Company creates an HTTP Server (powered by Apache)” on page 69 or you have an existing HTTP Server (powered by Apache) configuration.
- It is assumed you are familiar with Domain Name Servers (DNS).

Start the IBM Web Administration for iSeries interface

Note: Enter your Webmaster user profile username and password when prompted.

1. Start a Web browser.
2. Enter **http://[iSeries_hostname]:2001** in the location or URL field .

Example: **http://jkl_server:2001**

Note: If you have changed your port number for the IBM Web Administration for iSeries interface, replace port 2001 with your port number.

3. Click **IBM HTTP Server for iSeries**.

Note: If the IBM Web Administration for iSeries interface does not start, see “Install and test the HTTP Server” on page 125.

Set up a name-based virtual host

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server (powered by Apache) 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 iSeries 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 iSeries interface provides the IP addresses used by your iSeries 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

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: 9.5.61.228
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 (powered by Apache)

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 (powered by Apache) does not start, see “Troubleshoot” on page 758.

Test your HTTP Server (powered by Apache)

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 (powered by Apache) 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 (powered by Apache) 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
  DocumentRoot /www/jkltest/companyinfo/
  <Directory /www/jkltest/companyinfo>
    Order Allow,Deny
    Allow From all
  </Directory>
  Alias /companyinfo/ /www/jkltest/companyinfo/
</VirtualHost>
```

JKL Toy Company adds password protection for HTTP Server (powered by Apache)

This scenario discusses how to add password protection.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 iSeries 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 iSeries, 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 for HTTP Server" on page 68.
- It is assumed you have read and completed "JKL Toy Company creates an HTTP Server (powered by Apache)" on page 69 or you have an existing HTTP Server (powered by Apache) configuration.
- It is assumed you have read and completed "JKL Toy Company adds a new directory to HTTP Server (powered by Apache)" on page 71.
- It is assumed you have access to or the correct authority to create an iSeries library.

Create a library for validation lists on your iSeries

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

1. Start a 5250 session on your iSeries.
2. Enter **CRTLIB** 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 iSeries interface

Note: Enter your Webmaster user profile username and password when prompted.

1. Start a Web browser.
2. Enter **http://[iSeries_hostname]:2001** in the location or URL field .
Example: `http://jkl_server:2001`

Note: If you have changed your port number for the IBM Web Administration for iSeries interface, replace port 2001 with your port number.

3. Click **IBM HTTP Server for iSeries**.

Note: If the IBM Web Administration for iSeries interface does not start, see "Install and test the HTTP Server" on page 125.

Set up password protection for a directory on HTTP Server (powered by Apache)

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server (powered by Apache) 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 (powered by Apache)” on page 71 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 iSeries library and **users** is the name of the validation list.

12. Click **Continue**.
13. Select **Default server profile** from the **OS/400 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 (powered by Apache)

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 (powered by Apache)

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 (powered by Apache) does not start, see “Troubleshoot” on page 758.

Test your HTTP Server (powered by Apache)

1. Open a new Web browser.
2. Enter **http://[iSeries_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 (powered by Apache) 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 (powered by Apache) 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 (powered by Apache)

This scenario discusses how to add dynamic content with server-side includes.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 for HTTP Server” on page 68.
- It is assumed you have read and completed “JKL Toy Company creates an HTTP Server (powered by Apache)” on page 69 or you have an existing HTTP Server (powered by Apache) configuration.
- It is assumed you have installed and are familiar with iSeries Navigator.

Edit the index file (or welcome page) with iSeries Navigator

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

1. Start **iSeries Navigator**.
2. Expand the iSeries 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 782 for more information about SSI commands.

Start the IBM Web Administration for iSeries interface

Note: Enter your Webmaster user profile username and password when prompted.

1. Start a Web browser.
2. Enter **http://[iSeries_hostname]:2001** in the location or URL field .
Example: http://jkl_server:2001

Note: If you have changed your port number for the IBM Web Administration for iSeries interface, replace port 2001 with your port number.

3. Click **IBM HTTP Server for iSeries**.

Note: If the IBM Web Administration for iSeries interface does not start, see “Install and test the HTTP Server” on page 125.

Set up server-side includes for HTTP Server (powered by Apache)

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server (powered by Apache) 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 (powered by Apache)

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 (powered by Apache) does not start, see “Troubleshoot” on page 758.

Test your HTTP Server (powered by Apache)

1. Start a new Web browser.
2. Enter `http://[iSeries_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 (powered by Apache) 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 (powered by Apache) 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\." 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
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 (powered by Apache)

This scenario discusses how to enable SSL protection.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Scenario

The JKL Toy company (a fictitious company) wants to enable Secure Sockets Layer (SSL) protection for a specific directory on their HTTP Server (powered by Apache). 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 iSeries 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 (powered by Apache) with the iSeries Digital Certificate Manager.

Prerequisites

- It is assumed you have read “Scenarios for HTTP Server” on page 68.
- It is assumed you have read and completed “JKL Toy Company creates an HTTP Server (powered by Apache)” on page 69 or you have an existing HTTP Server (powered by Apache) configuration.
- It is assumed that a certificate authority (and certificate store) is already established for the iSeries Digital Certificate Manager.
- It is assumed you are familiar with Domain Name Servers (DNS).

Start the IBM Web Administration for iSeries interface

Note: Enter your Webmaster user profile username and password when prompted.

1. Start a Web browser.
2. Enter `http://[iSeries_hostname]:2001` in the location or URL field .
Example: `http://jkl_server:2001`

Note: If you have changed your port number for the IBM Web Administration for iSeries interface, replace port 2001 with your port number.

3. Click **IBM HTTP Server for iSeries**.

Note: If the IBM Web Administration for iSeries interface does not start, see “Install and test the HTTP Server” on page 125.

Set up a name-based virtual host

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server (powered by Apache) 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 iSeries 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 iSeries interface provides the IP addresses used by your iSeries 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: 443

Note: Specify a port number other than the one currently being used for your HTTP Server (powered by Apache) 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 OS/400 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 **OS/400 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 (powered by Apache)

The application name (created during the SSL process) is assigned a system certificate via the iSeries Digital Certificate Manager (DCM). During the process of enabling SSL for a virtual host, an iSeries 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 iSeries Tasks screen). See iSeries 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 (powered by Apache)

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 (powered by Apache) does not start, see “Troubleshoot” on page 758.

Test your HTTP Server (powered by Apache)

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 iSeries 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 (powered by Apache) 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 (powered by Apache) 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 (powered by Apache)

This scenario discusses how to enable single signon for your HTTP Server (powered by Apache).

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Important: This scenario is for OS/400 V5R2 or later. Enterprise Identity Mapping (EIM) is not supported in OS/400 V5R1.

To learn more about Kerberos and network security on the iSeries, 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 iSeries 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 OS/400 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 iSeries 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 iSeries server, *iSeries 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 OS/400 and applications on *iSeries A*, including the HTTP Server (powered by Apache). 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 iSeries system, known as *iSeries 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, *iSeries A* must be configured for network authentication service.
- The directory server on *iSeries 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 Directory Server for iSeries, which can run on any system in the network defined in that domain. Administrators can configure systems (EIM clients), such as OS/400, 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.

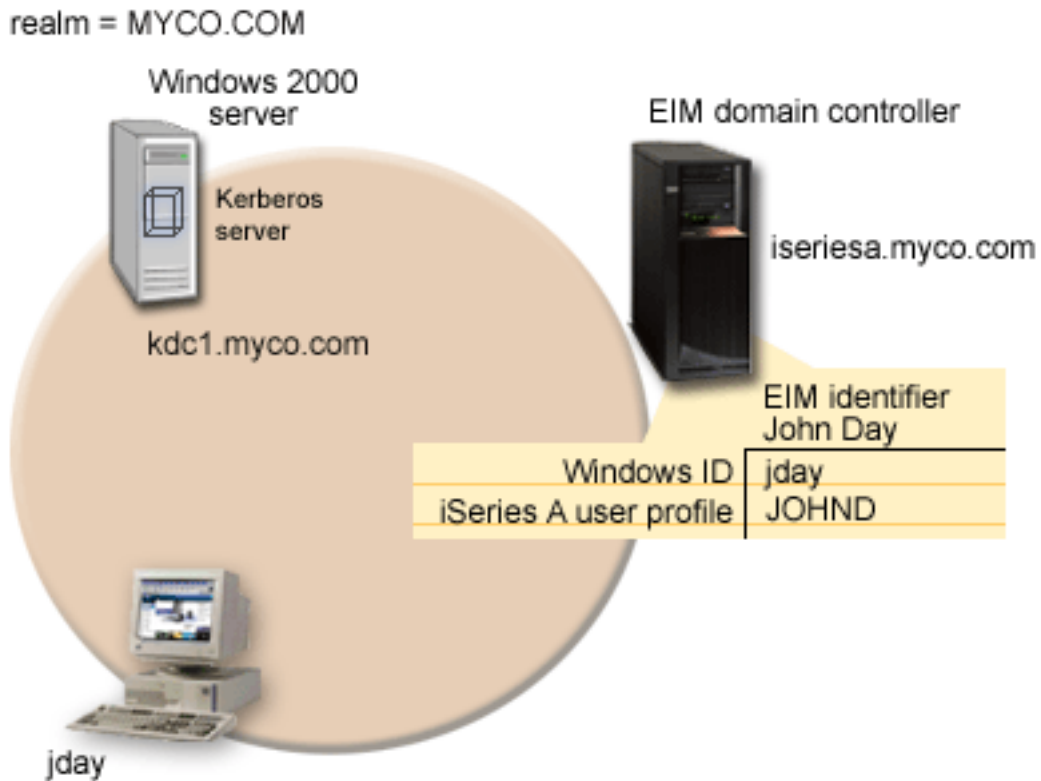
- **Microsoft Windows Active Directory**

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 *iSeries 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 iSeries.

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 *iSeries A* called *ISERIESA.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 *iSeries 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.

iSeries A

- Runs OS/400 Version 5 Release 2 (V5R2) with the following options and licensed products installed:
 - IBM HTTP Server for iSeries

- OS/400 Host Servers
- Qshell Interpreter
- iSeries Access for Windows
- Cryptographic Access Provider
- The IBM Directory Server for iSeries (LDAP) on *iSeries A* will be configured to be the EIM domain controller for the new EIM domain, *MyCoEimDomain*. *iSeries A* participates in the EIM domain, *MyCoEimDomain*.
- The principal name for *iSeries A* is *krbsvr400/iseriesa.myco.com@MYCO.COM*.
- The principal name for the HTTP Server on *iSeries A* is *HTTP/iseriesa.myco.com@MYCO.COM*.
- The user profile of *JOHND* exists on *iSeries A*. You will create a target association between this user profile and the EIM identifier, *John Day*.
- The home directory for the OS/400 user profile, *JOHND*, (*/home/JOHND*) is defined on *iSeries A*.

Client PC used for single signon administration

- Runs Microsoft Windows 2000 operating system.
- Runs V5R2 iSeries Access for Windows.
- Runs iSeries 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 for HTTP Server” on page 68.
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 iSeries Navigator, expand your **iSeries server** → **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 *iSeries A*.

Note: Instructions in this scenario are based on the assumption that the directory server has not been previously configured on *iSeries 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 iSeries server.

“Step 1: Planning work sheet”

“Step 2: Create a basic single signon configuration for *iSeries A*” on page 104

“Step 3: Add principal names to the KDC” on page 106

“Step 4: Add Kerberos keytab” on page 106

“Step 5: Create home directory for *John Day* on *iSeries A*” on page 107

“Step 6: Test network authentication service configuration on *iSeries A*” on page 107

“Step 7: Create EIM identifier for *John Day*” on page 108

“Step 8: Create a source association and target association for the new EIM identifier” on page 108

“Step 9: Configure iSeries Access for Windows applications to use Kerberos authentication” on page 109

“Step 10: Add *iSeries A* to an existing EIM domain” on page 109

“Step 11: Configure HTTP Server for single signon” on page 110

“Step 12: (Optional) Post configuration considerations” on page 111

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 9. Single signon prerequisite work sheet


Prerequisite work sheet	Answers
Are you running OS/400 at version V5R2 or higher?	Yes
Are the following options and licensed products installed on <i>iSeries A</i> ? <ul style="list-style-type: none"> • OS/400 Host Servers • Qshell Interpreter • iSeries Access for Windows • Cryptographic Access Provider 	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? Note: For this scenario, all of the participating PCs have iSeries Access for Windows installed and <i>iSeries A</i> has the HTTP Server for iSeries installed.	Yes
Is iSeries Navigator installed on the administrator’s PC? <ul style="list-style-type: none"> • Is the Security subcomponent of iSeries Navigator installed on the administrator’s PC? • Is the Network subcomponent of iSeries Navigator installed on the administrator’s PC? 	Yes
Have you installed the latest iSeries Access for Windows service pack? See iSeries Access  for the latest service pack.	Yes

Table 9. Single signon prerequisite work sheet (continued)

Prerequisite work sheet	Answers
Do you, the administrator, have *SECADM, *ALLOBJ, and *IOSYSCFG special authorities?	Yes
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. 1. Windows 2000 Server Note: Microsoft Windows 2000 Server uses Kerberos authentication as its default security mechanism. 2. Windows Server 2003 3. OS/400 PASE 4. AIX® server 5. zSeries®	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 iSeries 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 10. Single signon configuration planning work sheet for iSeries 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 iSeries A	Answers
How do you want to configure EIM for your system? • Join an existing domain • Create and join a new domain Note: 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.	Create and join a new domain Note: This will configure the directory server on the same system on which you are currently configuring EIM.
Do you want to configure network authentication service? Note: You must configure network authentication service to configure single signon.	Yes
The Network Authentication Service wizard launches from the EIM Configuration wizard. Use the following information to complete the Network Authentication Service wizard: Note: You can launch the Network Authentication Service wizard independently of the EIM Configuration wizard.	
What is the name of the Kerberos default realm to which your iSeries will belong? Note: A Windows 2000 domain is similar to a Kerberos realm. Microsoft Windows Active Directory uses Kerberos authentication as its default security mechanism.	MYCO.COM
Are you using Microsoft Active Directory?	Yes

Table 10. Single signon configuration planning work sheet for iSeries 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 iSeries 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?	KDC: <i>kdc1.myco.com</i> Port: <i>88</i> Note: 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 Password server: <i>kdc1.myco.com</i> Port: <i>464</i> Note: This is the default port for the Kerberos server.
For which services do you want to create keytab entries? • OS/400 Kerberos Authentication • LDAP • iSeries IBM HTTP Server for iSeries • iSeries NetServer™	OS/400 Kerberos Authentication Note: 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>iseriesa123</i> 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.
Do you want to create a batch file to automate adding the service principals for iSeries A to the Kerberos registry?	Yes
Do you want to include passwords with the OS/400 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. Note: 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.	Port: <i>389</i> Distinguished name: <i>cn=administrator</i> Password: <i>mycopwd</i> Note: 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 OS/400-- <i>ISERIESA.MYCO.COM</i> Kerberos-- <i>MYCO.COM</i> Note: The Kerberos principals stored on the Windows 2000 server are not case sensitive; therefore do not select Kerberos user identities are case sensitive.

Table 10. Single signon configuration planning work sheet for iSeries 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 iSeries A	Answers
Which EIM user do you want iSeries A to use when performing EIM operations? This is the system user Note: 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.	User type: Distinguished name and password User: <i>cn=administrator</i> Password: <i>mycopwd</i> 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.
After you complete the EIM Configuration wizard, use the following information to complete the remaining steps required for configuring single signon:	
What is the OS/400 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?	Source association: Kerberos principal <i>jday</i> Target association: OS/400 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 OS/400 user profile for which you are creating the target association?	<i>ISERIESA.MYCO.COM</i>

Step 2: Create a basic single signon configuration for iSeries A

You need to create a basic single signon configuration using the iSeries 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 *iSeries A*.

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

1. Start iSeries Navigator.
2. Expand **iSeries 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 OS/400 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 **OS/400 Kerberos Authentication**. Click **Next**.
- f. On the **Create OS/400 Keytab Entry** page, enter and confirm a password, and click **Next**. For example, *iSeries A123*. This password will be used when *iSeries 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 *iSeries 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 OS/400 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.

- 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 iSeries system to the Windows 2000 KDC, use the documentation for your KDC that describes the process of adding principals. By convention, the iSeries system name can be used as the username. Add the following principal names to the KDC:

```
krbsvr400/iSeriesA.ordept.myco.com@ORDEPT.MYCO.COM  
HTTP/iseriesa.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 iSeries system (select the **Users** folder, right-click, select **New**, then select **User**.) Specify *iSeriesA* as the Active Directory user and *HTTPiSeriesA* as the service principal for HTTP.
2. Access the properties on the Active Directory user *iSeriesA* and the service principal *HTTPiSeriesA*. From the **Account** tab, select the **Account is trusted for delegation**. This will allow the *HTTPiSeriesA* 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 *iSeriesA* and once for *HTTPiSeriesA*. 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/iSeriesA.ordept.myco.com@ORDEPT.MYCO.COM -mapuser iSeries A -pass iseriesa123
```

Then add the HTTP Server to the KDC:

```
ktpass -princ HTTP/iseriesa.myco.com@MYCO.COM -mapuser iSeries A -pass iseriesa123
```

For HTTP, an additional step (**setspn** - set service principal name) is required after the **ktpass** is done:

```
SETSPN -A HTTP/iseriesa.myco.com@MYCO.COM HTTPiSeriesA
```

Note: The value *iseriesa123* 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 OS/400 implementation of the Kerberos protocol) wizard creates a keytab entry for *iSeriesA*, 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 configured 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 *iSeries A*.
2. Type `QSH`.
3. Type `keytab add HTTP/iseriesa.myco.com`.
4. Type `iseries123` for the password.
5. Type `iseries123` again to confirm the password.
6. Type `keytab list`.

Note: The `keytab list` command lists the keytab information on your *iSeries* server.

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/iseriesa.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 *iSeries* 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/iseriesa.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 *iSeries 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 *iSeries A*.
2. Type `QSH`.
3. On a command line, enter: `CRTDIR '/home/user profile'` where *user profile* is your OS/400 user profile name. For example: `CRTDIR '/home/JOHND'`.

Step 6: Test network authentication service configuration on *iSeries A*

Now that you have completed the network authentication service configuration tasks for *iSeries A*, you need to test that your configuration. You can do this by requesting a ticket-granting ticket for the HTTP principal name, `HTTP/iseriesa.myco.com`.

To test the network authentication service configuration, complete these steps:

Note: Ensure that you have created a home directory for your OS/400 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/iseriesa.myco.com@MYCO.COM` displays as the principal name for *iSeries A*.
3. Enter `kinit -k HTTP/iseriesa.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/iseriesa.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 101. 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 iSeries Navigator.
2. Expand **iSeries 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* OS/400 user profile, which is the user identity that *John Day*, the person, uses to log in to iSeries Navigator and other OS/400 applications on *iSeries 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 iSeries Navigator.
2. Expand **iSeries 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 OS/400 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:** *iSeriesA.MYCO.COM*
 - **User:** *JOHND*

Note: The default behavior in V5R2 is to create the Kerberos registry as case sensitive. The **user** value entered here must be the same case as the user in Active Directory.

 - **Association type:** Target
9. Click **OK** to close the **Add Association** dialog.
10. Click **OK** to close the **Properties** dialog.

Step 9: Configure iSeries Access for Windows applications to use Kerberos authentication

You must use Kerberos to authenticate before you can use iSeries Navigator to access *iSeries A*. Therefore, from your PC, you need to configure iSeries Access for Windows to use Kerberos authentication. Jay Day will use iSeries Access to monitor the status of the iSeries HTTP Server and monitor the other activities on the iSeries.

To configure iSeries 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 iSeries Navigator on your PC, right-click *iSeries A* and select **Properties**.
3. On the **Connection** page, select **Use Kerberos principal name, no prompting**. This will allow iSeries 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 iSeries Navigator.

Step 10: Add *iSeries A* to an existing EIM domain:

The iSeries server does not require mapping, per the EIM configuration, as it is not a signon-type entity. You do, however, have to add the iSeries server to an existing EIM domain.

Note: IF EIM resides on the same iSeries server as the HTTP Server, then skip this step.

1. Start iSeries Navigator.
2. Expand **iSeries A** → **Enterprise Identity Mapping** → **Configuration**.
3. Click **Configure system for EIM**.
4. Click **Join an existing domain**. Click **Next**.
5. Type *iseriesa.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 *iseriasa.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 IBM Web Administration for iSeries interface.
2. Click the **Manage** tab.
3. Click the **HTTP Servers** subtab.
4. Select the HTTP Server (powered by Apache) 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 iSeries 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 iSeries system profile. If there is no appropriate target association for an iSeries system profile, the HTTP request will fail.
11. Click **Apply**.

Restart the HTTP Server (powered by Apache) 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 IBM Web Administration for iSeries interface.
2. Click the **Manage** tab.
3. Click the **HTTP Servers** subtab.
4. Select your HTTP Server (powered by Apache) 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 *iSeries A* has a high level of authority to all data on the directory server. Therefore, you might consider creating one or more DNs 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 DNs:

- 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 iSeries 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.

JKL Toy Company configures an in-process JSP with ASF Tomcat on HTTP Server (powered by Apache)

This scenario discusses how to configure ASF Tomcat to use an in-process JSP.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Scenario

The JKL Toy Company (a fictitious company) wants to take an existing Java server page (JSP) named **date.jsp** and enable it to run in-process using the ASF Tomcat servlet engine with the HTTP Server (powered by Apache). The Basic ASF Tomcat Server wizard is used to modify the existing Web server configuration called **JKLTEST** and create new workers.properties, server.xml, and web.xml configuration files.

Prerequisites

- It is assumed you have read “Scenarios for HTTP Server” on page 68.
- It is assumed you have read and completed “JKL Toy Company creates an HTTP Server (powered by Apache)” on page 69 or you have an existing HTTP Server (powered by Apache) configuration.
- It is assumed you have read “User profiles and required authorities for HTTP Server” on page 40.
- It is assumed you have a usable JSP.

Start the IBM Web Administration for iSeries interface

Note: Enter your Webmaster user profile username and password when prompted.

1. Start a Web browser.
2. Enter **http://[iSeries_hostname]:2001** in the location or URL field .
Example: **http://jkl_server:2001**

Note: If you have changed your port number for the IBM Web Administration for iSeries interface, replace port 2001 with your port number.

3. Click **IBM HTTP Server for iSeries**.

Note: If the IBM Web Administration for iSeries interface does not start, see “Install and test the HTTP Server” on page 125.

Set up ASF Tomcat for your HTTP Server (powered by Apache)

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server (powered by Apache) from the **Server** list.
Example: **JKLTEST**
4. Select **Global configuration** from the **Server area** list.
5. Expand **HTTP Tasks and Wizards**.
6. Click **Servlet and JSP Enablement**.
7. Click **Next**.

Note: If you have used the **Servlet and JSP Enablement** wizard prior to this scenario, the wizard will reset all values to the wizard default. Click **Next** to continue.

8. Select **I want to use a servlet or Java Server Page (JSP), and I either already have them or will provide them later.**
9. Click **Next**.
10. Select **I want to use a Java Server Page (JSP). I already have a JSP file or will provide it later.**
11. Click **Next**.
12. Click **Finish**.

Place your JSP file in the Web applications directory

Using a file transfer method such as iSeries Navigator, Netserver mapped drives, or FTP, transfer your JSP file to your HTTP Server (powered by Apache) `/www/[server_name]/webapps/app1/` directory.

Example: `/www/jkltest/webapps/app1/`

Note: All Java servlet class files supporting the JSP should be placed in `/www/[server_name]/webapps/app1/WEB-INF/classes/` directory.

Example: `/www/jkltest/webapps/app1/WEB-INF/classes/`

Restart your HTTP Server (powered by Apache)

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 (powered by Apache) does not start, see “Troubleshoot” on page 758.

Test your HTTP Server (powered by Apache)

1. Start a new Web browser.
2. Enter `http://[iSeries_hostname]:[port]/app1/[JSP_name.jsp]` in the location or URL field.

Example: `http://jkl_server:1975/app1/date.jsp`

Your JSP file will be displayed.

View your HTTP Server (powered by Apache) 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 (powered by Apache) from the **Server** list.

Example: JKLTEST

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

```
LoadModule jk_module /QSYS.LIB/QHTTPSVR.LIB/QZTCJK.SRVPGM
Listen *:1975
DocumentRoot /www/jkltest/htdocs
ServerRoot /www/jkltest
Options -ExecCGI -FollowSymLinks -SymLinksIfOwnerMatch -Includes -IncludesNoExec -Indexes -MultiViews
HotBackup Off
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
JkWorkersFile /www/jkltest/conf/workers.properties
JkLogFile /www/jkltest/logs/jk.log
JkLogLevel Error
JkMount /app1/* inprocess
JkMount /servlet/* inprocess
<Directory />
  Order Deny,Allow
  Deny From all
</Directory>
<Directory /www/jkltest/htdocs>
  Order Allow,Deny
  Allow From all
</Directory>
```

JKL Toy Company configures an in-process servlet with ASF Tomcat on HTTP Server (powered by Apache)

This scenario discusses how to configure ASF Tomcat to use an in-process servlet.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Scenario

The JKL Toy Company (a fictitious company) wants to take an existing Java servlet named `jklservlet.class` and enable it to run, in-process, using the ASF Tomcat servlet engine with the HTTP Server (powered by Apache). The **Servlet and JSP Enablement** wizard will be used to modify the existing HTTP Server (powered by Apache), JKLTEST.

Prerequisites

- It is assumed you have read “Scenarios for HTTP Server” on page 68.
- It is assumed you have read and completed “JKL Toy Company creates an HTTP Server (powered by Apache)” on page 69 or you have an existing HTTP Server (powered by Apache) configuration.
- It is assumed you have read “User profiles and required authorities for HTTP Server” on page 40.
- It is assumed you have a usable Java servlet compiled with JDK 1.2 or later.

Start the IBM Web Administration for iSeries interface

Note: Enter your Webmaster user profile username and password when prompted.

1. Start a Web browser.
2. Enter `http://[iSeries_hostname]:2001` in the location or URL field .
Example: `http://jkl_server:2001`

Note: If you have changed your port number for the IBM Web Administration for iSeries interface, replace port 2001 with your port number.

3. Click **IBM HTTP Server for iSeries**.

Note: If the IBM Web Administration for iSeries interface does not start, see “Install and test the HTTP Server” on page 125.

Set up ASF Tomcat for your HTTP Server (powered by Apache)

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server (powered by Apache) from the **Server** list.
Example: JKLTEST
4. Select **Global configuration** from the **Server area** list.
5. Expand **HTTP Tasks and Wizards**.
6. Click **Servlet and JSP Enablement**.
7. Click **Next**.

Note: If you have used the **Servlet and JSP Enablement** wizard prior to this scenario, the wizard will reset all values to the wizard default. Click **Next** to continue.

8. Select **I want to use a servlet or Java Server Page (JSP), and I either already have them or will provide them later**.
9. Click **Next**.
10. Select **I want to use a servlet. I either already have a class or jar file containing the servlet or will provide it later**.
11. Enter the name of your servlet in **Servlet class name** field.
Example: `jklservlet`
12. Click **Next**.
13. Click **Finish**.

Place your servlet file in the Web applications directory

Using a file transfer method such as iSeries Navigator, Netserver mapped drives, or FTP, transfer your servlet class files to your HTTP Server (powered by Apache) `/www/[server_name]/webapps/app1/WEB-INF/classes/` directory. Place JAR files in the `/www/server_name/webapps/app1/WEB-INF/lib/` directory.

Example: `/www/jkltest/webapps/app1/WEB-INF/classes/`

Example: /www/jkltest/webapss/app1/WEB-INF/lib/

Note: If you transfer the servlet source files only, you will need to compile them in the /classes directory. If transferring files over using a different user profile, ensure the files have the appropriate authorities after the transfer.

Restart your HTTP Server (powered by Apache)

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 (powered by Apache) does not start, see “Troubleshoot” on page 758.

Test your HTTP Server (powered by Apache)

1. Start a new Web browser.
2. Enter `http://[iSeries_hostname]:[port]/app1/[servlet_name]` in the location or URL field.
Example: `http://jkl_server:1975/app1/jklservlet`

Note: Servlet names must be in lowercase.

Your new servlet will be displayed.

View your HTTP Server (powered by Apache) 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 (powered by Apache) from the **Server** list.
Example: JKLTEST
4. Expand **Tools**.
5. Click **Display Configuration File**.

```
LoadModule jk_module /QSYS.LIB/QHTTPSVR.LIB/QZTCJK.SRVPGM
Listen *:1975
DocumentRoot /www/jkltest/htdocs
ServerRoot /www/jkltest
```

```

Options -ExecCGI -FollowSymLinks -SymLinksIfOwnerMatch -Includes -IncludesNoExec -Indexes -MultiViews
HotBackup Off
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
JkWorkersFile /www/jkltest/conf/workers.properties
JkLogFile /www/jkltest/logs/jk.log
JkLogLevel Error
JkMount /app1/* inprocess
JkMount /servlet/* inprocess
<Directory />
  Order Deny,Allow
  Deny From all
</Directory>
<Directory /www/jkltest/htdocs>
  Order Allow,Deny
  Allow From all
</Directory>

```

JKL Toy Company configures an in-process WAR file with ASF Tomcat on HTTP Server (powered by Apache)

This scenario discusses how to configure ASF Tomcat to use an in-process WAR.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Scenario

The JKL Toy Company (a fictitious company) wants to take an existing Web application archive (WAR) file named **jdklwar.war** and enable it to run in-process using the ASF Tomcat servlet engine with the HTTP Server (powered by Apache). The **Servlet and JSP Enablement** wizard is used to modify the existing HTTP Server (powered by Apache), **JKLTEST**.

Prerequisites

- It is assumed you have read “Scenarios for HTTP Server” on page 68.
- It is assumed you have read and completed “JKL Toy Company creates an HTTP Server (powered by Apache)” on page 69 or you have an existing HTTP Server (powered by Apache) configuration.
- It is assumed you have read “User profiles and required authorities for HTTP Server” on page 40.
- It is assumed you have a usable WAR file.

Start the IBM Web Administration for iSeries interface

Note: Enter your Webmaster user profile username and password when prompted.

1. Start a Web browser.
2. Enter **http://[iSeries_hostname]:2001** in the location or URL field .
Example: **http://jkl_server:2001**

Note: If you have changed your port number for the IBM Web Administration for iSeries interface, replace port 2001 with your port number.

3. Click **IBM HTTP Server for iSeries**.

Note: If the IBM Web Administration for iSeries interface does not start, see “Install and test the HTTP Server” on page 125.

Set up ASF Tomcat for your HTTP Server (powered by Apache)

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server (powered by Apache) from the **Server** list.
Example: JKLTEST
4. Select **Global configuration** from the **Server area** list.
5. Expand **HTTP Tasks and Wizards**.
6. Click **Servlet and JSP Enablement**.
7. Click **Next**.

Note: If you have used the **Servlet and JSP Enablement** wizard prior to this scenario, the wizard will reset all values to the wizard default. Click **Next** to continue.

8. Select **I want to use a Web Application Archive (WAR) file containing an entire server application, and I either already have the WAR file or will provide it later**.
9. Click **Next**.
10. Enter the name of your WAR file.
Example: jklwar.war
11. Click **Next**.
12. Click **Finish**.

Place your JSP file in the Web applications directory

Using a file transfer method such as iSeries Navigator, Netserver mapped drives, or FTP, transfer your WAR file to your HTTP Server (powered by Apache) `/www/[server_name]/webapps/` directory.

Example: `/www/jkltest/webapps/`

Restart your HTTP Server (powered by Apache)

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 (powered by Apache) does not start, see “Troubleshoot” on page 758.

Test your HTTP Server (powered by Apache)

1. Start a new Web browser.
2. Enter **http://[iSeries_hostname]:[port]/[WAR_file_name]/[WAR_file_content]** in the location or URL field.
Example: `http://jkl_server:1975/jklwar/jklwar.html`

The contents of your WAR file will be displayed.

View your HTTP Server (powered by Apache) 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 (powered by Apache) from the **Server** list.
Example: JKLTEST
4. Expand **Tools**.
5. Click **Display Configuration File**.

```
LoadModule jk_module /QSYS.LIB/QHTTPSVR.LIB/QZTCJK.SRVPGM
Listen *:1975
DocumentRoot /www/jkltest/htdocs
ServerRoot /www/jkltest
Options -ExecCGI -FollowSymLinks -SymLinksIfOwnerMatch -Includes -IncludesNoExec -Indexes -MultiViews
HotBackup Off
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
JkWorkersFile /www/jkltest/conf/workers.properties
JkLogFile /www/jkltest/logs/jk.log
JkLogLevel Error
JkMount /jklwar/* inprocess
JkMount /servlet/* inprocess
JkMount /jklwar inprocess
<Directory />
    Order Deny,Allow
    Deny From all
</Directory>
<Directory /www/jkltest/htdocs>
    Order Allow,Deny
    Allow From all
</Directory>
```

JKL Toy Company monitors Web server activity with logs on HTTP Server (powered by Apache)

This scenario discusses how to monitor Web server activity with logs.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 (powered by Apache) for advanced information.

Prerequisites

- It is assumed you have read “Scenarios for HTTP Server” on page 68.
- It is assumed you have read and completed “JKL Toy Company creates an HTTP Server (powered by Apache)” on page 69 or you have an existing HTTP Server (powered by Apache) configuration.

Start the IBM Web Administration for iSeries interface

Note: Enter your Webmaster user profile username and password when prompted.

1. Start a Web browser.
2. Enter `http://[iSeries_hostname]:2001` in the location or URL field .
Example: `http://jkl_server:2001`

Note: If you have changed your port number for the IBM Web Administration for iSeries interface, replace port 2001 with your port number.

3. Click **IBM HTTP Server for iSeries**.

Note: If the IBM Web Administration for iSeries interface does not start, see “Install and test the HTTP Server” on page 125.

Set up a log file

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server (powered by Apache) 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 (powered by Apache)

Select one of the following methods below:


Manage one server

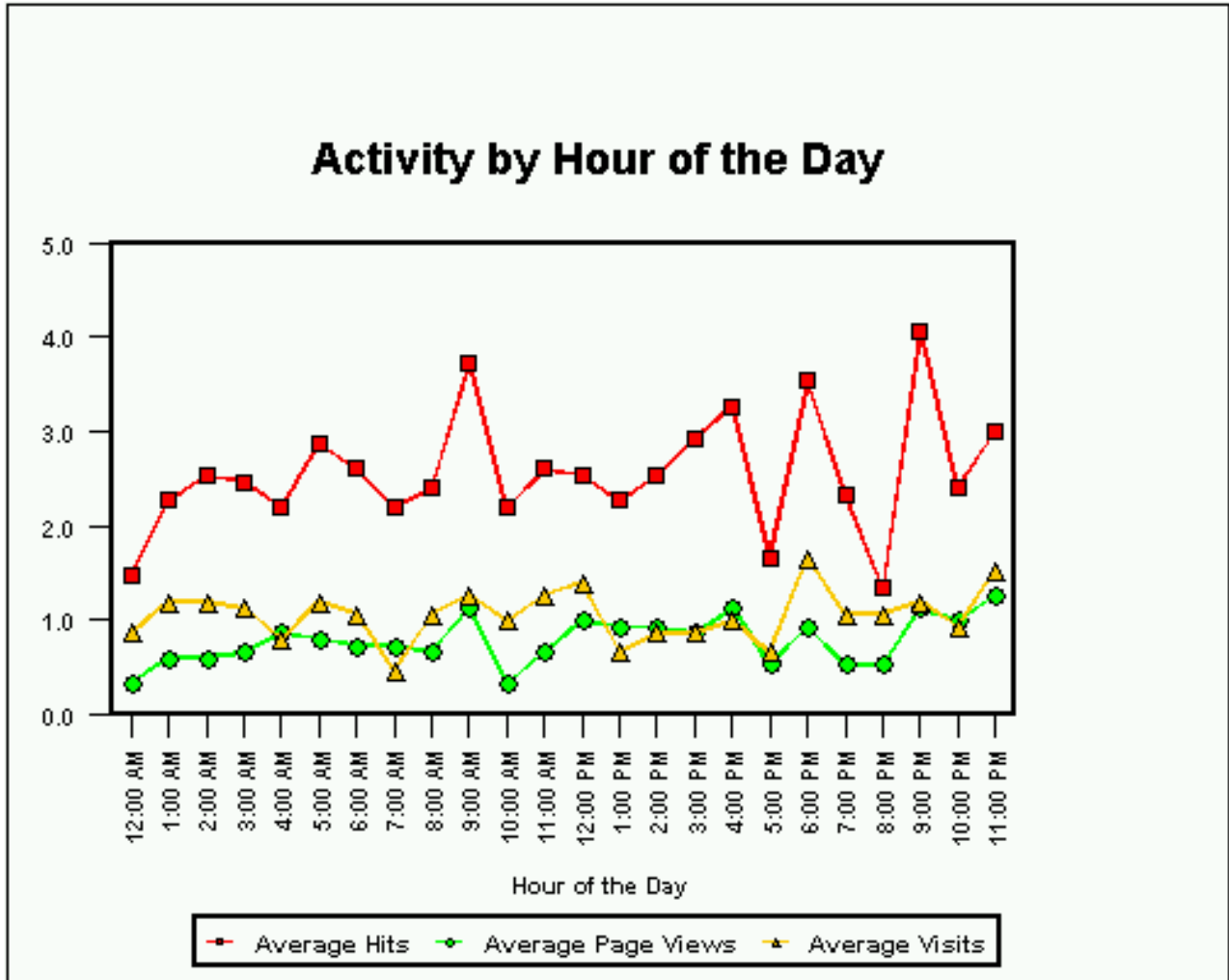
1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server from the Server list.
4. Click the **Stop** icon if the server is running.
5. Click the **Start** icon.

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 (powered by Apache) does not start, see “Troubleshoot” on page 758.

Logging will begin when the HTTP Server (powered by Apache) instance has started. The JKL Web administrator has decided to use the IBM WebSphere Site Analyzer  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 (powered by Apache)

1. Open a new Web browser.
2. Enter `http://[iSeries_hostname]:[port]` in the location or URL field.
Example: `http://jkl_server:1975`

Review your log for HTTP Server (powered by Apache) activity.

View your HTTP Server (powered by Apache) 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 (powered by Apache) 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\.\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>

```

Tasks

This topic provides step-by-step instructions for administration and management tasks with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Getting started with the IBM Web Administration for iSeries interface

This topic provides everything you need to know to get started with HTTP Server and the IBM Web Administration for iSeries interface, quickly.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Step 1: Install

Three easy steps to get you on your way to using the IBM Web Administration for iSeries interface.

1. Your first step is to install the IBM Web Administration for iSeries interface. See “Install and test the HTTP Server” on page 125 for more information.
2. Make sure you have the correct user profiles and required authorities for HTTP Server. See “User profiles and required authorities for HTTP Server” on page 40 for more information.
3. Finally, test your installation of the IBM Web Administration for iSeries interface. See “Test the installation of HTTP Server (powered by Apache) ” on page 126 for more information.

Step 2: Set up an HTTP Server (powered by Apache) instance

Once the IBM Web Administration for iSeries interface is ready, it is time to set up your first HTTP Server (powered by Apache) instance.

Use the **Create HTTP Server wizard** to quickly create a working HTTP Server (powered by Apache) configuration.

1. Start the IBM iSeries Tasks page.
Non-secure connection: `http://your.server.name:2001/`
Secure connection: `https://your.server.name:2010/`
2. Click **IBM Web Administration for iSeries**
3. Click the **Setup** tab.
4. Expand **Common Tasks and Wizards**.

Note: By default, all lists are expanded. If you collapse any list, the IBM Web Administration for iSeries interface displays the list as collapsed the next time you view it.

5. Click **Create HTTP Server**.
6. Enter a name to identify your HTTP Server (powered by Apache). This name is used later to configure and administer your server.
7. Click **Next**.
8. 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.
9. Click **Next**.
10. 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.
11. Click **Next**.
12. Leave the IP address list as **All addresses**. You may select a specific IP address if you so choose.
13. Enter a port number. Be default, the port is 80. This is the port your Web site runs (or "listen on"). It is suggested you enter a different port other than 80.
14. Click **Next**.
15. 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.
16. Click **Next**.
17. Specify how long you want to keep the error and access log files. Select **Keep, do not delete** or **Delete based upon age**.
18. Click **Next**.
19. The **Create HTTP Server wizard** displays a summary of HTTP Server (powered by Apache) configuration it creates. If you want to change an entry, simply click **Back**.
20. Click **Finish** and HTTP Server (powered by Apache) is created.

For more information on the IBM Web Administration for iSeries interface, see "IBM Web Administration for iSeries" on page 2.

Step 3: Start and test your HTTP Server (powered by Apache)

After using the **Create HTTP Server wizard**, it is time to start your Web server and go live.

1. Click **Manage newly created server**.
2. Click the **Start icon** under the **Server list**.
3. Click the Refresh icon and check if the server status is still shown as "Running".
If your HTTP Server (powered by Apache) does not start, see "Troubleshoot" on page 758.
4. Open another Web browser of your choice (see "Web browser requirements" on page 126) and go to `http://your.server.name:port/` where *your.server.name* is the name of your iSeries 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 IBM Web Administration for iSeries interface, expand its capabilities. See the “Scenarios for HTTP Server” on page 68 for more information.

Install and test the HTTP Server


This topic provides information about how to install and test the HTTP Server and the IBM Web Administration for iSeries interface after installation.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.


Install the HTTP Server

Basic requirements

- View the latest PTF requirements on the Internet at: <http://www.ibm.com/servers/eserver/iseries/software/http/services/service.htm> 

Note: To determine your current HTTP Server (powered by Apache) version , use the "-V" option on STRTCPSVR. For example, if HTTP Server (powered by Apache 2.0.43) is installed, STRTCPSVR SERVER(*HTTP) HTTPSVR(APACHEDFT '-V') displays:

```
Server version: Apache/2.0.43
Server built:   Nov 26 2002 15:57:01
```

- View the HTTP Server compatibility information at: <http://www.ibm.com/servers/eserver/iseries/software/http/product/compatibility.html> 
- Have a communication hardware adapter that is supported by the TCP/IP protocol stack.
- Add (ADDTCPIFC) and start (STRTCPIFC) at least one TCP/IP interface for the system's TCP/IP communications. You can use the NETSTAT OPTION(*IFC) command to work with the status of TCP/IP interfaces.
- Specify the system's TCP/IP host and domain name (CHGTCPDMN command) information.
- Add LOCALHOST to the TCP/IP host table.
- Install OS/400 Version 5 Release 3(5722-SS1).
- Have IBM Developer Kit for Java (5722-JV1) *BASE, Option , and Option 5 installed.
- Set the QSHRMEMCTL (shared memory) system value is to 1.

WebSphere Application Server - Express for iSeries

See WebSphere Application Server - Express product installation for information on how to install.

WebSphere Application Server for iSeries

If you plan to use WebSphere Application Server with the HTTP Server (powered by Apache), you will need to perform actions to install a version of the WebSphere Application Server Apache plug-in that is compatible with your current level of HTTP Server (powered by Apache). If the proper WebSphere Application Server PTFs are not loaded, the mismatch will prevent the HTTP Server from starting.

V5R3


•

See the WebSphere Application Server for iSeries product Web page  for information on the current PTFs, including those to upgrade to the latest WebSphere Application Server Apache plugin.

Optional software

- If you want to use secure sockets layer (SSL), you must install one of the IBM Cryptographic Access provider products:
 - Crypto Access Provider 56-bit for iSeries (5722-AC2)
 - Crypto Access Provider 128-bit for iSeries (5722-AC3)
- In order to provide the required support for handling digital server certificates used by SSL for secure Web serving, you must install Option 34 OS/400 - Digital Certificate Manager option (5722-SS1).
- If you want to use the native OS/400 interface to create and edit HTML files, install IBM WebSphere Development Studio for iSeries (5722-WDS).
- If you want to configure a high availability Web server cluster, then you need to install Option 41 HA Switchable Resources (5722-SS1), or use a business partner tool to manage clusters. See Clusters for more information.

To install

1. Insert the installation media for HTTP Server into your system.
2. At the OS/400 command line, type GO 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 5722-DG1 (HTTP Server for iSeries). If you want to use the Triggered Cache Manager, install 5722-DG1 Option 1. See the Software Installation guide  for help with licensed program installation.

Authorities

See “User profiles and required authorities for HTTP Server” on page 40 for information on user profiles and the required authorities necessary for the HTTP Server.

Web browser requirements

- To use the IBM Web Administration for iSeries interface you need a Web browser that supports the HTTP 1.0 or 1.1 protocol, frames, and Java Script. Suggested Web browsers include Internet Explorer 5.0 and later and Netscape 4.7x and later.

Note: Consult your Web browser documentation for more information.

Test the installation of HTTP Server (powered by Apache)

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

1. Verify that you have an active connection from the computer with the browser to the iSeries server.
2. Start the default server. In iSeries Navigator click **Network -> Servers -> TCP/IP** and right-click **HTTP Administration**.
3. Click **Start Instance -> APACHEDFT**.
4. Enter `http://your.server.name/` to view the default Welcome (or index) page.

Test the installation for HTTP Server (original)

The HTTP Server is installed with a default server configuration called CONFIG and default server instance called DEFAULT. The DEFAULT server instance is associated with the CONFIG server configuration. To test your installation, do the following:

1. Verify that TCP/IP is started and verify that you have an active connection from the computer with the browser to the server.
2. Start the DEFAULT server. In iSeries Navigator click **Network -> Servers -> TCP/IP** and right-click **HTTP Administration**.
3. Click **Start Instance -> DEFAULT**.
4. Enter `http://your.server.name/` to view the default Welcome (or index) page.

HTTP Server tasks

This topic provides step-by-step tasks for HTTP Server.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Related information

“Concepts of functions of HTTP Server” on page 2

This topic provides concepts of functions on HTTP Server and IBM Web Administration for iSeries interface.

Set up MIME types on HTTP Server (powered by Apache)

This topic provides information about how to set up MIME types for your HTTP Server with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 (powered by Apache) 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**.

Set up content and language negotiation for HTTP Server (powered by Apache)

This topic provides information about how to set up content and language negotiation for your HTTP Server with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 preferences.

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 (powered by Apache) 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**.

Set up customized error messages on HTTP Server (powered by Apache)

This topic provides information about how to set up customized error messages for your HTTP Server with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 (powered by Apache) 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 may be displayed in the Microsoft Internet Explorer browser, see “Symptom: Web browser problems with HTTP Server” on page 761.

Set up directory indexing and directory listing on HTTP Server (powered by Apache)

This topic provides information about how to set up a directory index and directory listing on your HTTP Server with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 (powered by Apache) 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 (powered by Apache) 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 an 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**.

Set up environment variables on HTTP Server (powered by Apache)

This topic provides information about how to set up environment variables on your HTTP Server to pass information about CGI requests to the server.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 (powered by Apache) 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 on HTTP Server” on page 766 for a list of environment variables.

Set up and administration of a highly available Web server cluster on HTTP Server (powered by Apache)

This topic provides information about how to set up and administer highly available Web server clusters for your HTTP Server with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

All required programs (HTTP Server, WebSphere, Servlets, Net.Data, and Clustering support) must already be installed on all nodes. See [!\[\]\(e3f255517d37bb309a3a931ec4849e6a_img.jpg\) http://www.iseries.ibm.com/ha](http://www.iseries.ibm.com/ha) and “Highly available Web server cluster on HTTP Server” on page 17 for more information.

Step 1 - Configure the iSeries Cluster for highly available HTTP Server (powered by Apache)

For each node, configure your cluster using the IBM Simple Cluster Management interface available through iSeries Navigator or use the cluster CL commands. See “Set up a highly available Web server using clustering CL commands for HTTP Server” on page 132 and Clusters for more information.

1. Start the node.

Use the Cluster CL commands found in “Set up a highly available Web server using clustering CL commands for HTTP Server” on page 132 or use the IBM Simple Cluster Management interface available through iSeries Navigator.

2. Continue to step 2.

Step 2 - Configure the highly available Web server

Configure IP addresses.

For each iSeries 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 IBM Web Administration for iSeries interface.
2. Click the **Manage** tab.
3. Click the **HTTP Servers** subtab.
4. Select your HTTP Server (powered by Apache) 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 the default value of *off* and ignored if currently configured for your HTTP Server (powered by Apache).

You may want to perform the next steps on one iSeries and copy (for example using FTP or NetServer) the HTTP Server (powered by Apache) configuration and instance files to each iSeries 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**.

Configure highly available HTTP Server (powered by Apache)

1. Start the IBM Web Administration for iSeries interface.
2. Click the **Manage** tab.
3. Click the **HTTP Servers** subtab.
4. Select your HTTP Server (powered by Apache) 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 3.

Step 3 - Start highly available HTTP Server (powered by Apache)

Start your highly available HTTP Server (powered by Apache).

1. Start a 5250 session on the iSeries you are going to use the highly available HTTP Server (powered by Apache).
2. Use **STRTCPSVR** CL command on the appropriate node.
3. Continue to step 4.

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 4 - Manage your highly available HTTP Server (powered by Apache)

Use the **ENDTCPSVR** CL command on the appropriate node or use the IBM Simple Cluster Management interface available through iSeries Navigator to stop or end your highly available HTTP Server (powered by Apache). 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. See "Set up a highly available Web server using clustering CL commands for HTTP Server" and Clusters for more information.

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.

Set up a highly available Web server using clustering CL commands for HTTP Server

This topic provides information about how to set up a highly available Web server using a 5250 session.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Set up your highly available HTTP Server instance using the clustering CL commands. Before using the following commands, see Clusters for more information.

The following is a subset of the Clustering CL commands that can be used to perform clustering operations. See “CL commands for HTTP Server” on page 765 for more information.

To create a cluster, use the following CL command:

CRTCLU

Create Cluster creates a new cluster for one or more nodes.

To add nodes to an existing cluster, use the following CL command:

ADDCLUNODE

Add Cluster Node Entry adds a node to the membership list of an existing cluster.

To start a node in the cluster, use the following CL command:

STRCLUNOD

Start Cluster Node starts Clusters Resource Services on a node in the cluster.

Note: It is suggested that you start a new cluster node from a node that is already active.

To determine the status of a node in a the cluster, use the following CL command:

DSPCLUINF

Display Cluster Information displays the status of a node in a cluster.

To perform a swtichover in the case of a Primary/Backup configuration, use the following CL command:

CHGCRGPRI

Change Cluster Resource Group Primary performs an administrative swtichover of the cluster resource group by changing the current roles of nodes in the recovery domain.

To remove a node from a cluster, use the following CL command:

RMVCLUNODE

Remove Cluster Node Entry removes a node from a cluster.

For more information on Clusters, see Clustering troubleshooting.

For instructions on how to set up a highly available Web server, see “Set up and administration of a highly available Web server cluster on HTTP Server (powered by Apache)” on page 130.

Set up a welcome or index page on HTTP Server (powered by Apache)

This topic provides information about how to set up a welcome or index page on your HTTP Server with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 (powered by Apache) 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 edit HTTP Server

This topic provides information about how to manually edit your HTTP Server configuration without the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Attention: It is strongly suggested that any editing to the HTTP Server configuration file be done through the IBM Web Administration for iSeries interface. 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 IBM Web Administration for iSeries interface has been designed to modify the HTTP Server configuration file by applying changes made to the various forms and wizards supplied by the IBM Web Administration for iSeries 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 IBM Web Administration for iSeries interface does not perform the usual error checking that is done when using the IBM Web Administration for iSeries 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 “Manage backup files for HTTP Server” on page 137 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.

Manually edit HTTP Server (powered by Apache) configuration: To modify the HTTP Server (powered by Apache) configuration manually, do the following:

1. Start the IBM Web Administration for iSeries interface.
2. Click the **Manage** tab.
3. Click the **HTTP Servers** subtab.

4. Select your HTTP Server (powered by Apache) 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.

Note: Command line **WRKHTTPCFG** (*Config_Name*) is not supported in V5R3 and later releases.

Manage HTTP Servers

This topic provides information about how to manage your HTTP Server with the with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 (powered by Apache), HTTP Server (original), or an ASF Tomcat server there are a number of basic tasks that you will need to know in order to manage your servers successfully.

- "Start and stop the ADMIN server"
- "Check status of server"
- "Start and stop server" on page 136
- "Rename server" on page 136
- "Manage HTTP Servers"
- "Delete server" on page 136

Start and stop the ADMIN server: The ADMIN server runs on port 2001 (or 2010 for a secure connection) and serves the iSeries Tasks page.

You can start the ADMIN server by doing one of the following:

- In iSeries Navigator click **Network** -> **Servers** -> **TCP/IP** and right-click **HTTP Administration**. Then click **Start Instance** -> **ADMIN**.
- On an OS/400 command line type **STRTCPSVR SERVER(*HTTP) HTTPSVR(*ADMIN)**.

You can stop the ADMIN server by doing one of the following:

- In iSeries Navigator click **Network** -> **Servers** -> **TCP/IP** and right-click **HTTP Administration**. Then click **Stop Instance** -> **ADMIN**.
- On an OS/400 command line type **ENDTCPSVR SERVER(*HTTP) HTTPSVR(*ADMIN)**.

Check status of server: To determine the status of your server, do the following using the IBM Web Administration for iSeries 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.

Start and stop server: Select one of the following methods below using the IBM Web Administration for iSeries 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 (powered by Apache) does not start, see "Troubleshoot" on page 758.

Rename server: To rename a server, do the following using the IBM Web Administration for iSeries 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.

Delete 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 IBM Web Administration for iSeries 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.

Manage addresses and ports for HTTP Server (powered by Apache)

This topic provides information about how to manage addresses and ports for your HTTP Server with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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.

If you add another Web server product such as Lotus® Domino® (with the HTTP task enabled) on the same iSeries, 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**.

Manage backup files for HTTP Server

This topic provides information about which files to backup for HTTP Server recovery.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Make sure that the following objects are included in your periodic backup activity:

Instance files

- QUSRSYS/QATMHINSTA
- QUSRSYS/QATMHINSTC

Configuration files

HTTP Server (powered by Apache)

Save the *conf* file located in the `/www/[server_name]/conf/` directory, where *Server_Name* is the name of your HTTP Server (powered by Apache) instance.

HTTP Server (original)

- QUSRSYS/QATMHTTP
- QUSRSYS/QATMHTTTPA
- QUSRSYS/QATMHTTTPC

ASF Tomcat

The ASF Tomcat files are located in the `/www/[server_name]/conf/` directory. For each Web application there is a configuration file (*web.xml*) in the same directory that the Web application is located under the `/www/[server_name]/webapps/` directory.

Additional files to save

- Signed certificates and certificate requests
- Group files
- Access control lists
- Validation lists
- Web content

Example: HTML, graphics, media objects, Java, CGI programs, Net.Data scripts

For more information about backup and recovery of files on the iSeries server, see Systems management.

Manage a directory on HTTP Server (powered by Apache)

This topic provides information about how to manage a directory on your HTTP Server with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

The following explains how to add a directory and how to remove a directory from your HTTP Server (powered by Apache) configuration.

Add a directory

The HTTP Server (powered by Apache) uses directories to serve Web pages and content. The IBM Web Administration for iSeries 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 (powered by Apache) 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 IBM Web Administration for iSeries interface will remove 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 (powered by Apache) 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**.

Manage highly available HTTP Server (powered by Apache)

This topic provides information about how to manage your highly available HTTP Server using a 5250 session.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

- To stop or end the highly available HTTP Server (powered by Apache), do the following:
Enter **ENDTCPSRV** CL command or use the IBM Simple Cluster Management interface available through iSeries Navigator.
- To force a switchover to cause the primary and backup HTTP Servers to switch roles (primary to backup, backup to primary), see “Set up a highly available Web server using clustering CL commands for HTTP Server” on page 132or using the IBM Simple Cluster Management interface available through iSeries Navigator.
- To determine which highly available HTTP Server is the primary or backup use **WRKACTJOB** CL command. The primary subsystem will have job QZHBEXPG running.

For more information on Clusters, see Clustering troubleshooting.

For instructions on how to set up a highly available Web server, see “Set up and administration of a highly available Web server cluster on HTTP Server (powered by Apache)” on page 130.

Manage ports for the administration (ADMIN) server on HTTP Server

This topic provides information about how to manage ports for the administration (ADMIN) HTTP Server with the IBM Web Administration for iSeries interface and using a 5250 session.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

The IBM Web Administration for iSeries interface runs on the ADMIN server. By default, the ADMIN server listens to port 2001. If this port is being used by another Web server or another application on your iSeries, the IBM Web Administration for iSeries interface will not start. You can change the default port number for the ADMIN server at any time to avoid conflicts with shared port numbers.

Two methods of editing your ADMIN server's port number are given below.

Note: After changing the port number, you will use the new port number to start the IBM Web Administration for iSeries interface.

Change ADMIN port with a 5250 session on iSeries:

1. Start a 5250 session and connect to your iSeries.
2. Enter **WRKLNK** ("**/QIBM/UserData/HTTPAdmin/conf/admin-cust.conf**") on the command line.
3. Edit **admin-cust.conf** configuration file.
4. Add a line after **# Customer additions to the admin configuration**.
5. Enter **Listen [port_number]** (where [port_number] is the new port number you want the ADMIN server to listen on).

Example: **# Customer additions to the admin configuration Listen 2222**

6. Save and exit the configuration file.

Stop and restart your server.

Change ADMIN port:

You will need authority to the file **/QIBM/UserData/HTTPAdmin/conf/admin-cust.conf** before you can change the port number of the ADMIN server.

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select **ADMIN** from the **Server** list.
4. Select **Include /QIBM/UserData/HTTPAdmin/conf/admin-cust.conf** from the **Server area** list.
5. Expand **Server Properties**.
6. Click **General Server Configuration**.
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 a port number other than 2001.
10. Click **Continue**.
11. Click **OK**.

Stop and restart your server.

Manage server performance for HTTP Server (powered by Apache)

This topic provides information about how to manage your HTTP Server performance.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

There are several things that can affect your server's performance. Consider the following performance related topics:

- "Local cache" on page 141

- “Files to cache when server has started”
- “Threads” on page 142
- “DNS lookups” on page 142
- “Server-side includes” on page 142
- “Content negotiation” on page 142
- “Document tree” on page 142
- “.htaccess files” on page 142
- “Virtual host log files” on page 142
- “KeepAlive and KeepAliveTimeout” on page 143
- “Logging” on page 143
- “CGI programs” on page 143
- “TCP/IP settings” on page 143
- “Network” on page 143

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 (powered by Apache) 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 iSeries 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 effect 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 (powered by Apache) 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 (powered by Apache) 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 Adjusting your TCP/IP configuration 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

This topic provides step-by-step tasks for configuration and management of compression files.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Related information

"File compression for HTTP Server (powered by Apache)" on page 26

This topic provides information on file compression and how output from your server is compressed before being sent to the client over the network.

Set up input decompression for HTTP Server (powered by Apache)

This topic provides information about how to set up input decompression for GZIP compressed input bodies for HTTP Server with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 browser 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 (powered by Apache)” on page 26 for more information.

To set up input compression, do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server (powered by Apache) from the **Server** list.
4. Select the context you want to work with from the **Server area** list.
5. Expand **Server Properties**.
6. Click **Container Management**.
7. Click **Compression**.
8. Click the **Input Filter** tab.
9. Click **Add** under the **Set input filter** table.
10. Specify a filter name under the **Filter name** column.
11. Click **Apply**.

You should now have something like the below example in your configuration.

Example

```
SetInputFilter DEFLATE
```

See “Set up output compression for HTTP Server (powered by Apache)” for more information.

Set up output compression for HTTP Server (powered by Apache)

This topic provides information about how to set up output compression for HTTP Server (powered by Apache) with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 (powered by Apache)” on page 26 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 (powered by Apache) 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 the same filter name used when the output filter was added.
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 “Set up input decompression for HTTP Server (powered by Apache)” on page 143 for more information.

Fast Response Cache Accelerator tasks

This topic provides step-by-step tasks for the Fast Response Cache Accelerator.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Related concepts

“Fast Response Cache Accelerator (FRCA) for HTTP Server (powered by Apache)” on page 27

This topic provides information on the Fast Response Cache Accelerator.

Set up Fast Response Cache Accelerator (FRCA) for HTTP Server (powered by Apache)

This topic provides information about how to set up FRCA for your HTTP Server with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

The following explains how to enable FRCA, FRCA logging, and FRCA file caching. See Fast Response Cache Accelerator (FRCA) for HTTP Server (powered by Apache) for more information.

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.

- “Enable FRCA” on page 146
- “Enable FRCA logging” on page 146
- “Enable FRCA file caching” on page 146

- “Enable FRCA reverse proxy caching” on page 147

Enable 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 (powered by Apache) 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.

Enable 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 (powered by Apache) for more information.

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server (powered by Apache) 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 **Attribute** column.
11. Click **Continue**.
12. Click **OK**.
13. Stop and restart your server.

Enable 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 (powered by Apache) 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** or keep the default value.
10. Enter a new value for **Maximum file size to cache** 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.

Enable 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 (powered by Apache) 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** or keep the default value.
10. Enter a new value for **Maximum proxy response size to cache** or keep the default value.
11. Enter a new value for **Document retention period** 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.
19. Click **Continue**.
20. Click **OK**.
21. Stop and restart your server.

Log and log file tasks

This topic provides step-by-step tasks for log and log file tasks.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Related information

“Log formats for HTTP Server (powered by Apache)” on page 29
This topic provides information about log formats and log files.

Set up logs on HTTP Server (powered by Apache)

This topic provides information about how to set up logs to record events and other information for your HTTP Server with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Your HTTP Server (powered by Apache) can generate a record of events commonly referred to as a Log. Logs can contain error messages, information on what is being access on your HTTP Server (powered by Apache), who is accessing your HTTP Server (powered by Apache), script logs, and FRCA logs.

The following topic 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 (powered by Apache) 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 (powered by Apache) 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 (powered by Apache). 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 (powered by Apache) 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 (powered by Apache) 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 (powered by Apache) 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 (powered by Apache) 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 (powered by Apache) 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 (powered by Apache)” on page 29.

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**.

For information on ASF Tomcat logs, see “About Tomcat” on page 43.

HTTP server job logs: The HTTP Server job logs contain messages or exceptions. The HTTP Server (powered by Apache) job log is maintained in the QHTTPSVR subsystem, listed with a job name matching the name of your HTTP Server instance.

Run a trace: The HTTP Server (powered by Apache) 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 iSeries 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:
 - ENDTCPSVR - 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(QP0Znnnnnn).
- 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

This topic provides step-by-step tasks for proxy.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Related information

“Proxy server types and uses for HTTP Server (powered by Apache)” on page 31

This topic provides information about proxy server types and uses.

Set up forward proxy for HTTP Server (powered by Apache)

This topic provides information about how to set up forward proxy for your HTTP Server with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Configure your HTTP Server (powered by Apache) for forward proxy using the IBM Web Administration for iSeries interface. Only the steps necessary to configure a forward proxy are discussed.

To configure your HTTP Server (powered by Apache) for forward proxy, do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server (powered by Apache) 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 (powered by Apache) 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 (powered by Apache). Edit the default values now or return to this form at a later time.

10. Click **OK**.

Set up reverse proxy for HTTP Server (powered by Apache)

This topic provides information about how to set up a reverse proxy for your HTTP Server with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Configure your HTTP Server (powered by Apache) for reverse proxy using the IBM Web Administration for iSeries interface. Only the tabs necessary to configure reverse proxy are discussed.

To configure your HTTP Server (powered by Apache) for reverse proxy, do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server (powered by Apache) 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 (powered by Apache) 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 (powered by Apache) for reverse proxy, you can configure your server for a proxy chain.

Set up proxy chaining for HTTP Server (powered by Apache)

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 iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Configure your HTTP Server (powered by Apache) for proxy chaining using the IBM Web Administration for iSeries interface. Only the steps necessary to configure a proxy chain are discussed. Before you can configure your HTTP Server (powered by Apache) for a proxy chain, you must configure your HTTP Server (powered by Apache) for forward proxy or reverse proxy.

To configure your HTTP Server (powered by Apache) for a proxy chain, do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server (powered by Apache) 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 (powered by Apache)" on page 81 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`

10. Enter the full or partial URL in the **Match requests to forward** column.
11. Click **Continue**.
12. Click **OK**.

Search tasks

This topic provides step-by-step tasks for the Webserver search engine.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Related information

“Webserver search engine on HTTP Server” on page 34

This topic provides information about the Webserver search engine and national language considerations.

Manage document lists for the Webserver search engine on HTTP Server

This topic provides information about how to manage document lists for the Webserver search engine with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Register a document list: Register a document list if it was created manually or does not appear listed in the IBM Web Administration for iSeries interface.

Note: See “Set up a document list for the Webserver search engine on HTTP Server” on page 161 for more information on creating document lists.

To register a document list, do the following:

Note: Document lists created using Search Administration forms or the CFGHTTSPSCH command starting in Version 4 Release 5 do not need to be registered.

1. Click the **Advanced** tab.
2. Click the **Search Setup** subtab.
3. Expand **Search Engine Setup**.
4. Click **Register document list**.
5. Enter the directory and file name of the document list file. For example, `/QIBM/UserData/HTTPSVR/index/mydoclist.DOCUMENT_LIST`.

Choose one of the two following options:

Document list built from document on this server

Select this option if you created the document list from a local directory.

Document list built from documents found by crawling web sites

Select this option if you created the document list from a remote server.

6. Click **Apply**.

Delete a document list: To delete the document list, do the following:

1. Click the **Advanced** tab.
2. Click the **Search Setup** subtab.
3. Expand **Search Engine Setup**.
4. Click **Delete document list**.
5. Select the document list to be deleted from the list.
6. Click **Delete**.

Note: If the document list was not registered, it will not be listed.

Work with document list status: The Webserver search engine can show the status of a document list. To view the status of a document list, do the following:

1. Click the **Advanced** tab.
2. Click the **Search Setup** subtab.
3. Expand **Search Engine Setup**.
4. Click **Work with document list status**.
5. Select the document list you want to work with from the **Document list file name list**
6. Click **Apply**.

The status page displays all the current information available for the document list. Use this page to view information about your document list and to keep track of its status. Different information will be displayed for document lists created from local documents and those created from remote sites. If your document list contains documents from remote web sites, you can also control the current crawling session by using the buttons displayed at the end of the form. An active crawling session can be stopped or paused and a paused crawling session can be stopped or resumed. Once a crawling session is stopped, it can be started again from the Build document list form.

Set up a thesaurus dictionary for the Webserver search engine on HTTP Server

This topic provides information about how to set up a thesaurus dictionary file for use with the Webserver search engine with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

The thesaurus support allows you to automatically expand a search query by using a thesaurus. To make sure information is always found in your indexed documents, you can create your own thesaurus in which you can list common terms and associate them with the terms that exist in your documents. For example, if a person typically searches for *PC* but your documents only refer to a personal computer, just add *PC* to your thesaurus as a synonym of *personal computer*. You must first create a thesaurus definition file that contains terms that are related. Then you build the thesaurus dictionary to be used by the Webserver search engine. See "Build the thesaurus dictionary" on page 157 for more information.

Note: The thesaurus definition file can be created in IFS or QSYS.LIB.

Create a thesaurus definition file: To create a thesaurus definition file, do the following:

1. Open a text editor on the iSeries, such as edtf.

Note: Use edtf or some other iSeries editor rather than a PC editor. It is important that the file is tagged with the correct CCSID since the words will be matched with words in the documents themselves.

2. Create the content of the thesaurus definition file using the following file format:

- a. Open a text editor on the iSeries, such as edtf.

Note: Use edtf or some other iSeries editor rather than a PC editor. It is important that the file is tagged with the correct CCSID since the words will be matched with words in the documents themselves.

- b. Create the content of the thesaurus definition file using the following file format:

A thesaurus definition file consists of blocks containing elements. Each element of the block is defined by a capitalized keyword. The block also contains terms that are single or multiple words. For example "cake" and "chocolate cake" are terms.

Each block starts with :WORDS followed on the same line by one of the following:

RELATED

Where :RELATED indicates related terms that are not synonyms.

:SYNONYM

Where :SYNONYM indicates terms that are synonyms.

Member terms are listed in the block starting on the second line of the block, one term per line.

For example:

:WORDS:SYNONYM

PC

personal computer

The following relationships can also be specified within the block:

.LOWER_THAN

Where the block member terms are more specific in meaning than the term following .LOWER_THAN.

.HIGHER_THAN

Where the block member terms are less specific in meaning than the term following .HIGHER_THAN.

.RELATED_TO

Block member terms are related to this term.

.SYNONYM_OF

Block member terms are synonyms of this term.

A related term is specified on the same line as the relationship. A term is a single or multiple words. The relationships can be specified in any order within the block. For example the two following blocks are interpreted exactly the same:

:WORDS

rain

snow

hail

.LOWER_THAN precipitation

.RELATED_TO weather

:WORDS

.LOWER_THAN precipitation

.RELATED_TO weather

rain

snow

hail

When creating a thesaurus definition file, keep the following in mind:

- Preceding and trailing blanks are removed.
- Preceding and trailing control characters are removed.
- Terms beginning with a period (.) or a colon (:) are not allowed.
- Capital letters and small letters of the same character are treated as the same character.
- Leave the keywords that are UPPERCASED as-is.
- Terms in the file may be in any language.

- The maximum length of a term is 64 characters or 64 bytes.

Note: A sample thesaurus definition file is stored in /QIBM/ProdData/HTTP/Public/HTTPSVR/sample_thesaurus.txt.

- c. Once you have created the thesaurus definition file, save it as a text file (txt).

Note: Terms can be added in any supported language; however, the keywords (:RELATED, :SYNONYM, .LOWER_THAN, .HIGHER_THAN, .RELATED_TO, .SYNONYM_OF, and :WORDS) can not be changed in order for the definition file to work.

After you have created a thesaurus dictionary, you can manage a thesaurus dictionary . See “Manage a thesaurus dictionary for the Webserver search engine on HTTP Server” for more information. To use the dictionary on a search, select your index and the search option. After you have selected to do a simple or advanced search, you will reach a form that allows you to add a thesaurus dictionary to your search.

Build the thesaurus dictionary: To build the thesaurus dictionary, allowing it to be used by the Webserver search engine, do the following:

1. Click the **Advanced** tab.
2. Click the **Search Setup** subtab.
3. Expand **Search Engine Setup**.
4. Click **Build thesaurus dictionary**.
5. Enter the directory and name of the thesaurus definition file that contains relationship data for generating a thesaurus dictionary in the **Thesaurus definition file** field. A definition file is a simple text file with formatting tags to indicate word relationships.
6. Enter a name for the thesaurus dictionary in the **Thesaurus dictionary name** field. For example, mydict.
7. Enter the directory that is used to hold the thesaurus dictionary files that are created in the **Thesaurus dictionary directory** field. Possible values include /QIBM/UserData/HTTPSVR/search (the default setting), or any valid directory path.
8. Click **Apply**.

Manage a thesaurus dictionary for the Webserver search engine on HTTP Server

This topic provides information about how to manage your thesaurus dictionary with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

This topic explains how to manage thesauruses for the Webserver search engine on the HTTP Server.

Test thesaurus dictionaries: Test your thesaurus for accuracy and completeness. To test your thesaurus, do the following:

1. Click the **Advanced** tab.
2. Click the **Search Setup** subtab.
3. Expand **Search Engine Setup**.
4. Click **Test thesaurus dictionary**.
5. Enter the thesaurus dictionary name to be tested.
6. Enter the directory where the thesaurus dictionary is located.
7. Enter a term or multiple terms to test.

There are a number of options you can test on your thesaurus dictionary.

Find terms with all relationships

Check this box to find all the terms in the dictionary that have a broader or narrower meaning than the entered term, are synonyms of the term, or are related to the term.

Find terms with a broader meaning

Check this box to find all terms that have a broader meaning than the entered term. For example, dessert has a broader meaning than cake.

Find terms with a narrower meaning

Check this box to find all terms that have a narrower meaning than the entered term. For example, apple has a narrower meaning than fruit.

Find terms that are synonyms

Check this box to find all terms that are synonyms. For example, Ping-Pong is a synonym for table tennis.

Find terms that are related

Check this box to find all terms that are related to the entered term. For example, apple and pear are related terms

8. Enter the number of related terms that will be displayed for each term entered. Possible values are 1 - 10.
9. Click **Expand terms**.

Retrieve thesaurus definitions: You can retrieve a thesaurus definition from a thesaurus dictionary. Specify the file to use for the thesaurus definition. The thesaurus data will be retrieved from the specified thesaurus dictionary. To retrieve a thesaurus definition, do the following:

1. Click the **Advanced** tab.
2. Click the **Search Setup** subtab.
3. Expand **Search Engine Setup**.
4. Click **Retrieve thesaurus definition**.
5. Enter the directory and name of the thesaurus definition file that contains relationship data for generating a thesaurus dictionary in the **Thesaurus definition file** field. A definition file is a simple text file with formatting tags to indicate word relationships.
6. Enter a name for the thesaurus dictionary in the **Thesaurus dictionary name** field. For example, *mydict*.
7. Enter the directory that is used to hold the thesaurus dictionary files that are created in the **Thesaurus dictionary directory** field. Possible values include `/QIBM/UserData/HTTPSVR/search` (the default setting), or any valid directory path.
8. Click **Apply**.

Delete thesaurus dictionaries: You can delete your thesaurus dictionaries. Files that were created for this dictionary will be deleted. The directory will not be deleted. To delete a thesaurus dictionary, do the following:

1. Click the **Advanced** tab.
2. Click the **Search Setup** subtab.
3. Expand **Search Engine Setup**.
4. Click **Delete thesaurus dictionary**.
5. Enter a name for the thesaurus dictionary in the **Thesaurus dictionary name** field. For example, *mydict*.
6. Enter the directory that is used to hold the thesaurus dictionary files that are created in the **Thesaurus dictionary directory** field. Possible values include `/QIBM/UserData/HTTPSVR/search` (the default setting), or any valid directory path.

7. Click **Delete**.

Manage search indexes for the Webserver search engine on HTTP Server

This topic provides information about how to manage your search indexes with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

This topic explains how to manage search indexes for the Webserver search engine on HTTP Server. "Set up the Webserver search engine on HTTP Server (powered by Apache)" on page 169 to learn how to create a search index.

The following topics explain how to manage your search index.

- "Update search index"
- "Merge search index" on page 160
- "Delete search index" on page 161
- "View the status of search index" on page 161

Update search index: After creating a search index, you will want to update it when ever you change the documents being searched. For example, if you add additional content to some of your files, that content will not be found in using the Webserver search engine until you update your search index. You can add new or changed documents or delete documents from the index. If documents are added, they will be indexed and added to the supplemental index. Use the fields to specify information about the documents to add to the index. Once you have finished adding the documents, you may want to merge the supplemental and main indexes.

To update your search index, do the following:

1. Click the **Advanced** tab.
2. Click the **Search Setup** subtab.
3. Expand **Search Engine Setup**.
4. Select your search index from the **Index** list.
5. Click **Update search index**.
6. Choose one of the following:

Build a document list from this directory

Select this option to create a document list containing the set of documents to add, update, or delete from your search index.

There are two additional options that you may select.

Traverse subdirectories in this directory

Select to include any documents in subdirectories of the directory you provided in the field above.

Document filter

Select this option if you want the document list to be made of specific file types. For example, entering **.htm** will only build a document list of file types *htm* and *html*.

Use the document list in this file

Select this option if you want to use an existing document list to add, update, or delete document from your search index.

7. Choose an updating processing option:

Add new or changed documents to the index

Select this option to use the document list to add or update document in your search index.

Delete documents from the index

Select this option to use the document list to delete documents from your search index.

8. Choose a document content option:

Documents are HTML documents

Select this option if the documents to be searched are HTML.

Documents are TEXT documents

Select this option if the documents to be searched are text documents (do not contain HTML tags).

9. Choose a processing option:

Update index immediately

Select this option if you want the search index to be updated as soon as you complete the form.

Update index in a background task

Select this option if you want the search index to be updated in a background task. This is the suggested method if there is a large number of documents to be indexed.

10. Choose a document error processing option:

Skip documents in error

Select this option if you want the search index to continue updating if document errors are found. The documents found with an error will not be included in the search index.

Stop processing when error occurs

Select this option if you want the search index to stop updating if document errors are found. All documents before the error will be indexed.

11. Click **Apply**.

To update the search index using the CFGHTTPSCH *ADDDOC and CFGHTTPSCH *MGRIDX CL commands, do the following:

```
CFGHTTPSCH OPTION(*ADDDOC) IDX(myindex) IDXDIR('/mydir') +
DOCLIST('/mydir/myindex.DOCUMENT_LIST')
CFGHTTPSCH OPTION(*MGRIDX) IDX(myindex) IDXDIR('/mydir')
```

You can also update your index through scheduled batch jobs. For example, to update an index on Friday of every week at 11:30 p.m., do the following:

```
ADDJOBSCDE JOB(UPDATE) CMD( CFGHTTPSCH OPTION(*ADDDOC) IDX('myindex')
DOCLIST('/QIBM/UserData/HTTPSVR/index/myindex.DOCUMENT.LIST'))
JOB(UPDATE)
FRQ(*WEEKLY)
SCDDATE(*NONE)
SCDDAY(*FRI)
SCDTIME('23:30:00')
```

To update an index at 11:30 p.m. on the last day of every month, do the following:

```
ADDJOBSCDE JOB(UPDATE) CMD(CFGHTTPSCH OPTION(*ADDDOC) IDX('myindex')
DOCLIST('/QIBM/UserData/HTTPSVR/index/myindex.DOCUMENT.LIST') )
JOB(UPDATE)
SCDDATE(*MONTHEND)
SCDTIME('23:30:00')
FRQ(*MONTHLY)
```

Merge search index: The supplemental index contains indexed documents that were added with the update search index form and are not yet merged into the main index. This merged index will include both the supplemental index and the main index.

To merge your search indexes, do the following:

1. Click the **Advanced** tab.
2. Click the **Search Setup** subtab.
3. Expand **Search Engine Setup**.
4. Select your supplemental search index from the **Index** list.
5. Click **Merge search index**.
6. Choose one of the two processing options:

Merge index immediately

Select this option if you want the search indexes to merge as soon as you complete the form.

Merge index in a background task

Select this option if you want the search indexes to merge in a background task. This is the suggested method if the search indexes are large.

7. Click **Apply**.

Delete search index: You can delete both the main and supplemental indexes or just the supplemental index. The supplemental index contains indexed documents that were added with the update search index form and are not yet merged into the main index.

To delete your search indexes, do the following:

1. Click the **Advanced** tab.
2. Click the **Search Setup** subtab.
3. Expand **Search Engine Setup**.
4. Select your search index from the **Index** list.
5. Click **Delete search index**.
6. Choose one of the two processing options:

Delete both main and supplemental indexes

Select this option if you want to delete both the supplemental and main search indexes from your iSeries.

Delete only the supplemental index

Select this option if you want to delete the supplemental search index from your iSeries. Your main search index will not be deleted.

7. Click **Delete**.

View the status of search index: The Webserver search engine can show the status of a search index allowing you to keep track of its progress when updating or merging.

To view the status of a search index, do the following:

1. Click the **Advanced** tab.
2. Click the **Search Setup** subtab.
3. Expand **Search Engine Setup**.
4. Select your search index from the **Index** list.
5. Click **View status of search index**.

The status page displays all the current information available for the search index. Use this page to see any file errors that occurred during indexing.

Set up a document list for the Webserver search engine on HTTP Server

This topic provides information about how to create a document list for the Webserver search engine with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

A document list is a file that contains a list of documents used to create or update a search index. When a request for a search title or description is sent, it is compared to the document list for possible matches.

To set up a document for use with the Webserver search engine, complete the following steps:

Create a document list: To create a document list, do the following:

1. Click the **Advanced** tab.
2. Click the **Search Setup** subtab.
3. Expand **Search Engine Setup**.
4. Click **Build document list**.
5. Choose one of the two options:

“Build a document list from documents on this server”

Select this option if the documents to be included in the document list are in a local directory.

“Build the document list by crawling a URL” on page 163

Select this option if the documents to be included in the document list reside in a remote server.

There are two additional options if you choose to build the document list using the Web crawler. These are:

Build the document list by crawling a URL

Select this option to crawl a single URL.

“Build the document list from selected URL and options objects” on page 164

Select this option only if you have previously created a “Set up a URL object for the Webserver search engine on HTTP Server” on page 167 and an “Set up an options object for the Webserver search engine on HTTP Server” on page 165.

6. Click **Apply**.

Build a document list from documents on this server: If you opted to build a document list from a local directory, follow these instructions to complete your document list:

1. Choose one of the two document list file name options:

Create a new document list file

Select this option to create a new document list file. Replace the asterisk (*) with a new name for your document list file.

Use the document list in this file

Select this option to use an existing document list file. Select the document list file from the list.

There are two additional options if you choose to use an existing document list file. These are:

Replace the document list file

Select this option to overwrite the existing document list file.

Append the new list to the document list file

Select this option to add any new information to the existing document list file. This option will not delete existing information.

2. Enter the directory the document list will build from in the **Build a document list from this directory** field. For example, */www/mydocs/public/info*.

There are two additional options that you may select. These are:

Traverse subdirectories in this directory

Select to include any documents in subdirectories of the directory you provided in the field above.

Document filter

Select this option if you want the document list to be made of specific file types. For example, entering **.htm** will only build a document list of file types *htm* and *html*.

3. Click Apply.

Build the document list by crawling a URL: If you opted to build a document list with the Web crawler that will crawl a URL, follow these instructions to complete your document list:

1. Choose one of the two document list file name options:**Create a new document list file**

Select this option to create a new document list file. Replace the asterisk (*) with a new name for your document list file.

Use the document list in this file

Select this option to use an existing document list file. Select the document list file from the list.

There are two additional options if you choose use an existing document list file. These are:

Replace the document list file

Select this option to overwrite the existing document list file.

Append the new list to the document list file

Select this option to add any new information to the existing document list file. This option will not delete existing information.

2. Enter the Web crawler options:

URL Enter the URL the Web crawler will visit to add documents to your document list. For example, *http://www.ibm.com*.

URL domain filter

Enter the URL domain filter the Web crawler will stay on. For example, *ibm.com*.

Maximum crawling depth

Enter the depth of the crawling from the starting URL. For example, entering a depth of 0 will download only the starting URL page. Selecting a depth of 1, will continue the crawl to the first layer of links from the starting URL.

Support robot exclusion

If you select *Yes*, any site or pages that contain robot exclusion META tags or files will not be downloaded. Excluded files do not usually contain HTML or text. See "Manage Web spiders, Web crawlers, and robots on HTTP Server" on page 166 for more information.

3. Choose crawling options:**Directory to store documents**

Enter the directory to store the documents the Web crawler finds. For example, */www/mydocs/public/crawl*.

Document language

Select the language of the documents being retrieved by the Web crawler.

Proxy server for HTTP

Enter the proxy server for HTTP requests. Possible values include any valid server name.

Proxy port for HTTP

Enter the port number for the above proxy server. A port is required if a proxy server for HTTP is specified.

Proxy server for HTTPS

Enter the proxy server for HTTPS requests.

Proxy port for HTTPS

Enter the port number for the above proxy server.

Maximum file size to download

Enter the maximum size for a downloaded file (in KB).

Maximum storage for files

Enter the maximum storage space for all downloaded files (in MB).

Maximum threads

Enter the maximum number of threads used during web crawling. Set this value based on the system resources that are available.

Maximum run time

Enter the maximum amount of time the crawling session remains active in hours and minutes.

Activity log file

Enter the action to take for an activity log file. This file contains information about the crawling session plus any errors that occur. This file must be in a directory of the IFS. You can choose to run a crawling session with or without an activity log file. You also have the option of replacing the log file each time a crawling session is started or appending information to the existing file.

There are two additional options if you choose to write an activity log. These are:

Create or replace the logging file

Select this option if the log file does not exist or you want to overwrite an existing log file.

Append to the existing logging file

Select this option to add any new information to the existing log file. This option will not delete existing information.

4. Click Apply.

Build the document list from selected URL and options objects: If you opted to build a document list with the Web crawler using selected URL and options objects, follow these instructions to complete your document list:

1. Choose one of the two document list file name options:**Create a new document list file**

Select this option to create a new document list file. Replace the asterisk (*) with a new name for your document list file.

Use the document list in this file

Select this option to use an existing document list file. Select the document list file from the list.

There are two additional options if you choose use an existing document list file. These are:

Replace the document list file

Select this option to overwrite the existing document list file.

Append the new list to the document list file

Select this option to add any new information to the existing document list file. This option will not delete existing information.

2. Select the "Set up a URL object for the Webserver search engine on HTTP Server" on page 167.
3. Select the "Set up an options object for the Webserver search engine on HTTP Server" on page 165.
4. Select "Set up validation lists for the Webserver search engine on HTTP Server" on page 172:

Validation list

Select **Do not use a validation list** if you know the server the Web crawler will visit does not use a validation list for authentication. Otherwise, select **Use this validation list for sites requiring a userid and password** and select the validation list to be used from the list.

5. Click **Apply**.

Set up an options object for the Webserver search engine on HTTP Server

This topic provides information about how to set up an options object for use with the Webserver search engine with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

An options object contains values that are used when remote web sites are crawled. If you select to edit an existing object, the current values are displayed and can be changed. This object can be selected together with a URL object to use when you select to build document lists by crawling remote web sites. To create an options object, do the following:

1. Click the **Advanced** tab.
2. Click the **Search Setup** subtab.
3. Expand **Search Engine Setup**.
4. Click **Build options object**.
5. Choose options object options:

Create an options object

Select this option to create a new options object. Enter the name of the new options object.

Select an options object to edit

Select this option to edit an existing options object. Select the options object from the list.

6. Click **Apply**.
7. Enter crawling options:

Proxy server for HTTP

Enter the proxy server for HTTP requests. Possible values include any valid server name.

Proxy port for HTTP

Enter the port number for the above proxy server. A port is required if a proxy server for HTTP is specified.

Proxy server for HTTPS

Enter the proxy server for HTTPS requests. Possible values include any valid server name.

Proxy port for HTTPS

Enter the port number for the above proxy server. A port is required if a proxy server for HTTPS is specified.

Maximum file size to download

Enter the maximum size for a downloaded file (in KB).

Maximum storage for files

Enter the maximum storage space for all downloaded files (in MB).

Maximum threads

Enter the maximum number of threads used during web crawling. Set this value based on the system resources that are available.

Maximum run time

Enter the maximum amount of time the crawling session remains active in hours and minutes.

Activity log file

Enter the action to take for an activity log file. This file contains information about the crawling session plus any errors that occur. This file must be in a directory of the IFS. You can choose to run a crawling session with or without an activity log file. You also have the option of replacing the log file each time a crawling session is started or appending information to the existing file.

There are two additional options if you choose to write an activity log.

Create or replace the logging file

Select this option if the log file does not exist or you want to overwrite an existing log file.

Append to the existing logging file

Select this option to add any new information to the existing log file. This option will not delete existing information.

8. Click Apply.

Your new options object can now be used when Web crawling remote sites.

Manage Web spiders, Web crawlers, and robots on HTTP Server

This topic provides information about how to manage Web spider, Web crawlers, and robots.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Web spiders, Web crawlers, and robots are programs that traverse the Internet retrieving documents and following links in those documents. You may have noticed entries in your log files that document requests for /robots.txt files or requests for many of your Web documents. These requests may be from a robot. Most robots adhere to the robot exclusion protocol. If you want to control what portion of your Web site robots attempt to visit, you can either use a robots.txt file or the robots meta tag.

The robots.txt file

The robots.txt file must be placed in the document root directory of the server. The following is an example of a robots.txt file:

```
User-agent: *  
Disallow: /cgi-bin/
```

Note: Make sure that you do not alert hackers to important directories or files by listing them in the robots.txt file.

Robots meta tag

The robots meta tag can be placed in HTML documents to tell the robot:

- Do not index a document
<META NAME="ROBOTS" CONTENT="NOINDEX">
- Do not follow links in a document
<META NAME="ROBOTS" CONTENT="NOFOLLOW">

Set up a URL mapping rules file for the Webserver search engine on HTTP Server

This topic provides information about how to set up a URL mapping rules file for use with the Webserver search engine with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

A mapping rules file is used to set the URL for documents found on a search. The file is built using the configuration of the selected HTTP Server and the specified URL prefix.

In order to allow documents to be opened when found on a search, you will need an HTTP Server configuration that contains at least one routing directive. The routing directive has two purposes. It determines what URLs will be accepted, and it optionally maps those URLs to physical file names. These routing directives are added to the mapping rules file and are used to set the URL for the documents found on a search.

The format of the routing directive depends on the HTTP Server that is selected. If you have documents in directory */root/clothing/* but want the URL to contain just */clothing* as the directory, then you need a routing directive in the HTTP Server configuration. If there is no routing directive that maps a URL to an actual path, an error may occur attempting to display the document.

To create a URL mapping rules file, do the following:

1. Start the IBM Web Administration for iSeries interface.
2. Click the **Advanced** tab.
3. Click the **Search Setup** subtab.
4. Expand **Search Engine Setup**.
5. Select your search index from the **Index** list.
6. Click **Build URL mapping rules file**.
7. Select your HTTP Server instance from the HTTP servers list.
8. Enter a prefix to use for URL addresses. For example, *http://www.ibm.com*.

This optional prefix is the first part of the URL that is created for a document. For example, if you enter a prefix such as *http://www.ibm.com*, the URL of a document found on a search would be *http://www.ibm.com/clothing/doc1.htm*. If this field is left blank, then the name of the server and port would be assumed for the URL, for example, */clothing/doc1.htm*. If the server is started on a TCP/IP port other than the default of port 80, be sure to include the port number in the prefix as follows: *http://myserver.com:8282*.

9. Accept the default entry in the **Mapping rules file name** field.

There are two additional options for you to choose.

Create or replace the mapping rules file

Select this option to overwrite the existing mapping rules file.

Append the new list to the mapping rules file

Select this option to add any new information to the existing mapping rules file. This option will not delete existing information.

10. Click **Apply**.

Set up a URL object for the Webserver search engine on HTTP Server

This topic provides information about how to set up a URL object file for use with the Webserver search engine with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

A URL object contains a list of URLs plus additional web crawling attributes. If you select to edit an existing URL object, the contents of the current object are displayed. The URL object can be selected together with an options object to use when you select to build document lists by crawling remote web sites. See “Set up a document list for the Webserver search engine on HTTP Server” on page 161 for more information.

To create a URL object, do the following:

1. Start the IBM Web Administration for iSeries interface.
2. Click the **Advanced** tab.
3. Click the **Search Setup** subtab.
4. Expand **Search Engine Setup**.
5. Click **Build URL object**.
6. Choose URL object options:

Create a new URL object

Select this option to create a new URL object. Enter the name of the new URL object.

Edit this URL object

Select this option to edit an existing URL object. Select the URL object from the list.

7. Click **Apply**.
8. Enter document storage and language options:

Directory to store documents

Enter the directory where documents found on web sites are stored. Possible values include any valid directory path name.

Document language

Select the language of the documents that are downloaded. The list provides all valid language entries.

9. Enter URL list options:

Action

Click **Add** to add a new row.

Click **Remove** to remove an existing row.

URL Enter a URL in the form, for example, `http://www.ibm.com`. If you enter a URL that requires authentication, create a validation list using the Build validation list form. See “Set up validation lists for the Webserver search engine on HTTP Server” on page 172 for more information.

URL domain filter

Enter a domain to limit crawling, for example, *ibm.com*.

Maximum crawling depth

Enter the depth of links from the starting URL to continue crawling. The starting URL is at depth 0. The links on that page are at depth 1.

Support robot exclusion

Choose whether to support robot exclusion. If you select Yes, any site or pages that contain robot exclusion META tags or files will not be downloaded.

10. Click **Apply**.

Your new URL object can now be used when Web crawling remote sites.

Set up the Webserver search engine on HTTP Server (powered by Apache)

This topic provides information about how to set up the Webserver search engine for your HTTP Server with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

The Webserver search engine allows you to perform full text searches on HTML and text files. You can control what options are available to the user and how the search results are displayed through customized Net.Data macros.

Follow these steps to configure the Webserver search engine for HTTP Server (powered by Apache) using the IBM Web Administration for iSeries interface.

Edit HTTP Server (powered by Apache)

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server (powered by Apache) 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**.

Using the wizard, create a new directory to serve static Web content, with directory `/www/MyServer/scripts/`, and with an alias `/scripts/`, where *MyServer* is the name of your HTTP Server (powered by Apache). See "JKL Toy Company adds a new directory to HTTP Server (powered by Apache)" on page 71 for more information.

Note: The new scripts directory and the directory containing the files to be searched must have *Public *RX authority or Net.data will return an error.

7. Expand **Server Properties**.
8. Click **Container Management**.
9. Click the **Directories** tab in the form.
10. Click **Add** under the **Directory/Directory Match container** table.
11. Select **Directory** from the list in the **Type** column.
12. Enter `/QSYS.LIB/QHTTPSVR.LIB/` in the **Directory path or expression** column.
13. Click **Continue**.
14. Click **OK**.
15. Click **URL Mapping**.
16. Click the **Aliases** tab in the form.
17. Click **Add** under the **URL to host file system mappings** table.
18. Select **Alias** from the list in the **Alias Type** column.
19. Enter `/htdocs/` in the **URL path** column.
20. Enter `/www/MyServer/htdocs` in the **Host directory or file** column, where *MyServer* is the name of your HTTP Server (powered by Apache).
21. Click **Add**.
22. Select **ScriptAlias** from the list in the **Alias Type** column.

23. Enter `/cgi-bin/db2www/` in the **URL path** column.
24. Enter `/QSYS.LIB/QHTTSPVR.LIB/DB2WWW.PGM/` in the **Host directory or file** column.
25. Click **Continue**.
26. Click **OK**.
27. Select **Directory \www\MyServer\scripts** from the **Server area list**, where *MyServer* is the name of your HTTP Server (powered by Apache).
28. Click **Dynamic Content and CGI**.
29. Click the **General Settings** tab in the form.
30. Select **Enabled** for **Allow CGI programs to be run**.
31. Click **OK**.
32. Click **Security**.
33. Click the **Control Access** tab in the form.
34. Select **All authenticated users (valid user name and password)** under **Control access based on who is making the request**.
35. Select **Allow then deny** from **Order for evaluating access** under **Control access based on where the request is coming from**.
36. Select **All client hosts** under **Client hosts allowed access to this resource**.
37. Click **OK**.
38. Select **Directory /QSYS.LIB/QHTTSPVR.LIB/** from the **Server area list**.
39. Click **Dynamic Content and CGI**.
40. Click the **General Settings** tab in the form.
41. Select **Enabled** for **Allow CGI programs to be run**.
42. Click **OK**.
43. Click **Security**.
44. Click the **Control Access** tab in the form.
45. Select **All authenticated users (valid user name and password)** under **Control access based on who is making the request**.
46. Select **Allow then deny** from **Order for evaluating access** under **Control access bade on where the request is coming from**.
47. Select **All client hosts** under **Client hosts allowed access to this resource**.
48. Click **OK**.

Create search index

1. Click the **Advanced** tab.
2. Click the **Search Setup** subtab.
3. Expand **Search Engine Setup**.
4. Click **Create search index**.
5. Enter an index name in the **Index name** field.
6. Click **Apply**.
7. Select **Build a document list from this directory** under **Document list**.
8. Enter the directory where the HTML and text files to searched are located. Example, `/www/MyServer/htdocs`, where *MyServer* is the configuration directory of your HTTP Server (powered by Apache).
9. Select **Create a mapping rules file from this HTTP server** under **Mapping rules**.
10. Select your HTTP Server (powered by Apache) configuration name from the list.
11. Enter a URL prefix in the **Prefix to use for URL address field**. Example, `http://iSeries:Port`, where *iSeries:Port* is the name of your iSeries and port number.

Note: Remember the search index name and mapping rules directory and file name. This information will be used in the following steps.

12. Click **Apply**.

Copy sample_search.ndm and sample_html.html files

1. Open directory `\QIBM\ProdData\HTTP\Public\HTTPSVR`.
2. Copy `sample_html.html` to `\www\MyServer\htdocs` directory, where *MyServer* is the configuration directory of your HTTP Server (powered by Apache).
3. Copy `sample_search.ndm` to `\www\MyServer\scripts` directory, where *MyServer* is the configuration directory of your HTTP Server (powered by Apache).

Note: File `sample_search.ndm` must have `*Public *RX` authority or `Net.data` will return an error.

Edit sample_search.ndm

1. Open `sample_search.ndm` with a text editor.
2. Change the following lines found under `%{ ---- STEP 1 -----%}`:

`idxIndexName="Recipes"`

Search index name: replace **Recipes** with the name of your search index.

`mapFile=""`

Map file directory and name: add `/QIBM/UserData/HTTPSVR/index/SearchIndex.MAP_FILE`, where *SearchIndex* is the name of your search index.

3. Save and close the file.

Note: The `sample_search.ndm` file has additional fields that allow you to modify the search results displayed with the Webserver search engine. Read the documentation provided in the `sample_search.ndm` file for more details.

Edit sample_html.html

1. Open `sample_html.html` with a text editor.
2. Change the following lines:

`FORM NAME="search" ACTION="/cgi-bin/db2www/qibm/proddata/http/public/httpsvr/sample_search.ndm/output" METHOD="post">`

Replace the **ACTION** attribute value with `/cgi-bin/db2www/www/MyServer/scripts/sample_search.ndm/output`, where *MyServer* is the configuration directory of your HTTP Server (powered by Apache)

`<INPUT TYPE="hidden" NAME="frmIndexName" VALUE="Recipes">`

Search index name: replace **Recipes** with the name of your search index.

`<INPUT TYPE="hidden" NAME="frmMapFile" VALUE="">`

Map file directory and name: add `/QIBM/UserData/HTTPSVR/index/SearchIndex.MAP_FILE` to the **VALUE** attribute, where *SearchIndex* is the name of your search index.

3. Save and close the file.

Note: The `sample_html.html` file contains additional fields for the Webserver search engine. Only the necessary information for the required fields are documented here.

Start the server

1. Click the **Manage** tab.
2. Click the **All Servers** subtab.
3. Click the **All HTTP Servers** tab.
4. Select your HTTP Server (powered by Apache) from the table.

5. Click **Start**.

Test the Webserver search engine

1. Open a Web browser.
2. Enter **http://iSeries:Port/sample_html.html** to start the Webserver search engine, where *iSeries:Port* is the name of your iSeries and port number.
3. Enter a search term.
4. Click **Search**.

Note: If the MAP_FILE is incorrectly built, the search results will not contain the correct URL and the Web page will not be displayed.

Set up validation lists for the Webserver search engine on HTTP Server

This topic provides information about how to set up a validation list for use with the Webserver search engine with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

A validation list contains URLs, userids, and password sets. This list can be specified when you select to build document lists by crawling remote web sites. To create a validation list, do the following:

1. Start the IBM Web Administration for iSeries interface.
2. Click the **Advanced** tab.
3. Click the **Search Setup** subtab.
4. Expand **Search Engine Setup**.
5. Click **Build validation list**.
6. Choose validation list options:

Create a new validation list

Select this option to create a new validation list. Enter the name of the new validation list.

Edit this validation list

Select this option to edit an existing validation list. Select the validation list from the list.

7. Click **Apply**.
8. Enter validation list content options:

Action

Click **Add** to add a new row.

Click **Remove** to remove an existing row.

URL domain filter

Enter the URL domain filter. For example, *ibm.com*.

Userid

Enter the Userid to be used on the URL domain.

Password

Enter the password to be used for the Userid on the URL domain.

9. Click **Apply**.

Your new validation list can now be used when Web crawling remote sites.

Security tasks

This topic provides step-by-step tasks for security.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Related information

“Security tips for HTTP Server” on page 38

This topic provides tips to secure your HTTP Server.

Set up password protection on HTTP Server (powered by Apache)

This topic provides information about how to set up password protection for resources on your HTTP Server with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 IBM Web Administration for iSeries interface.
- 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,  
bapel 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 (powered by Apache) 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 OS/400 profile of client** under **User authentication method**.

Note: Your selection should be based off of the incoming traffic your HTTP Server (powered by Apache) 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 OS/400 profiles. If incoming traffic is from a local access network, using OS/400 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 an 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 iSeries 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 (powered by Apache) 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 OS/400 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 **OS/400 user profile to process requests** field.

Select a user name under **OS/400 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 (powered by Apache) can act as a LDAP server 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 (powered by Apache) 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**.

Set up SSL for the administration (ADMIN) server for HTTP Server

This topic provides information about how to secure your administration server configuration with Secure Socket Layers with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

You can SSL enable the ADMIN server by doing the following:

1. Make sure that you have the following products and product options installed:
 - IBM Cryptographic Access provider product (5722-AC3)
 - Digital Certificate Manager Option 34 of 5722-SS1
2. To complete this task you must supply a digital certificate. For more information on how to obtain a digital certificate, see Digital certificate management.
3. Make sure you have proper authority to the directories and file. See “User profiles and required authorities for HTTP Server” on page 40 for more information.
4. Make sure that the ADMIN server is running.
5. Click the **Manage** tab.
6. Click the **All HTTP Servers** subtab.
7. Select **ADMIN** from the **Server** list.
8. Select **Include /QIBM/UserData/HTTPA/admin/conf/admin-cust.conf** from the **Server area** list.
9. Expand **Tools**.
10. Select **Edit Configuration File**.

Note: The following changes must be made using the **Edit Configuration File** tool. Use of other editing tools may result in errors.

11. Enter the following information into the configuration file or remove the “#” symbol to uncomment these lines:

```
LoadModule ibm_ssl_module /QSYS.LIB/QHTTPSVR.LIB/QZSRVSSL.SRVPGM
Listen 2001
Listen 2010
SetEnv HTTPS_PORT 2010
<VirtualHost *:2010>
  SSLEnable
  SSLAppName QIBM_HTTP_SERVER_ADMIN
</VirtualHost>
```

12. Click **OK**.
13. Select **Virtual Host *:2010** from the **Server area** list.
14. Expand **Server properties**, and select **Security**.
15. Click **OK**.
16. Click the **Related Links** tab.
17. Click **Digital Certificate Manager**.

18. Click **Select a Certificate Store**.
19. Select ***SYSTEM**.
20. Click **Continue**.
21. Enter a password in the **Certificate store password** field.
22. Click **Continue**.
23. Click **Manage Applications**.
24. Select **Update certificate assignment**.
25. Click **Continue**.
26. Select **Server**.
27. Click **Continue**.
28. Select **QIBM_HTTP_SERVER_ADMIN** application name.
29. Click **Update Certificate Assignment**.
30. Select the appropriate certificate.
31. Click **Assign New Certificate** to assign the certificate to the application name selected in the previous step.
32. Restart the ADMIN server.
33. Restart your Web browser.

To use the ADMIN server, type **http://[iSeries_hostname]:2001** for a non-secure connection or **https://[iSeries_hostname]:2010** for a secure connection.

Note: If you have trouble getting the secure connection working, check the ADMIN error log file located in the (\QIBM\UserData\HTTPAdmin\admin\logs\ directory for information.

Set up to secure against a Telnet denial of service attack for HTTP Server

This topic provides information about how to secure your HTTP Server against a Telnet denial of service attack with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

A Telnet attack could result in a denial of service to your HTTP Server. The configuration to protect against attacks has default settings, but you may want to change them to suit your individual needs.

Your HTTP Server can detect a denial of service 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 denial of service 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 denial of service protection is built into each of these entities. You should turn off the Telnet denial of service protection for this HTTP Server instance so that the HTTP Server does not falsely detect a denial of service condition.

Secure against a Telnet denial of service attack for HTTP Server (powered by Apache):

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server (powered by Apache) 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 (powered by Apache)” on page 40 for more information if you encounter authority problems.

Tomcat tasks

This topic provides step-by-step tasks for the Apache Software Foundation Tomcat servlet engine.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Related information

“About Tomcat” on page 43

This topic provides information about the Apache Software Foundation Jakarta Tomcat servlet engine.

Manage ASF Tomcat servers on HTTP Server (powered by Apache)

This topic provides information about how to manage your Tomcat server with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Use the IBM HTTP Server Administrator interface to manage your HTTP Server (powered by Apache) from using ASF Tomcat or stop an out-of-process ASF Tomcat server.

Enable and disable ASF Tomcat for HTTP Server (powered by Apache): You can enable or disable (without discarding your configuration) the ASF Tomcat servlet engine by doing the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server (powered by Apache) from the **Server** list.
4. Expand **Server Properties**.
5. Click **ASF Tomcat Settings**.
6. Check or uncheck **Enable servlets from this HTTP Server**.
7. Click **OK**.

Start and stop out-of-process ASF Tomcat server: You can start or stop an out-of-process ASF Tomcat server by doing the following:

1. Click the **Manage** tab.
2. Click the **All Servers** subtab.
3. Click the **All ASF Tomcat Servers** tab.
4. Select the ASF Tomcat out-of-process server you want to stop.
5. Click **Stop** or **Start**.

Set up JSP files for out-of-process ASF Tomcat server on HTTP Server (powered by Apache)

This topic provides information about how to set up a Java JavaServer Page (JSP) file for use with the Tomcat servlet engine with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Follow the instructions to use a JSP file with out-of-process ASF Tomcat servlet engine on your HTTP Server (powered by Apache).

Before you begin, review the “User profiles and required authorities for HTTP Server” on page 40 topic.

Set up HTTP Server (powered by Apache) for ASF Tomcat:

1. Start the IBM Web Administration for iSeries interface.
2. Click the **Manage** tab.
3. Click the **HTTP Servers** subtab.
4. Select your HTTP Server (powered by Apache) from the **Server** list.
5. Select **Global configuration** from the **Server area** list.
6. Expand **Server Properties**.
7. Click **ASF Tomcat Setup** task from the task list.
8. Select **Enable** from **Enable servlets for this HTTP Server** option.
9. Accept the default workers definition file provided.
10. Click **Next**.
11. Select **Enable** from **Enable “out-of-process” servlet engine connections** option.

Note: Do not select to enable an “in-process” servlet engine.

12. Click **Add**.
13. Enter the name of your ASF Tomcat server in the **Worker name** field.
14. Select **Binary (AJP13)** from **Worker type**.
15. Enter information based off one of the below conditions for **Hostname:Port**.
 - a. Enter **localhost** and a **port number** if your out-of-process ASF Tomcat server is on your iSeries. For example, localhost:8009.
 - b. Enter the **name of the remote server host** and the **port number** if your out-of-process ASF Tomcat server is not on your local iSeries. For example, IBMiSeries:8009.
16. Click **Continue**.
17. Click **Next**.
18. Click **Add**.
19. Enter **/app1/*** for **URL (Mount point)**.

20. Accept the default value for **ASF Tomcat worker**.
21. Click **Add**.
22. Enter the name of your JSP file with a forward slash "/" before it and *.jsp* after for **URL (Mount point)**. For example, */myjsp.jsp*.
23. Accept the default value for **ASF Tomcat worker**.
24. Click **Add**.
25. Enter */servlet/** for **URL (Mount point)**.
26. Accept the default value for **ASF Tomcat worker**.
27. Click **Add**.
28. Enter */app1* for **URL (Mount point)**.
29. Accept the default value for **ASF Tomcat worker**.
30. Click **Continue**.
31. Click **Next**.
32. Click **Finish**.
33. Click **OK**.

Set up the out-of-process ASF Tomcat:

1. Click the **Manage** tab.
2. Click the **ASF Tomcat Servers** subtab.
3. Expand **Tomcat Tasks and Wizards**.
4. Click **Create ASF Tomcat Server**.
5. Enter your worker name in the ASF Tomcat server name field. This is the name of the ASF Tomcat server.
6. Click **Next**.
7. Accept the default value (QTMHHTTP) or enter a specific user id in **Server userid**.
8. Accept the default value for **Java version (JDK)**.
9. Accept the default value for **ASF Tomcat home**.
10. Accept the default value for **Java classpath entires**.
11. Click **Next**.
12. Accept the default value for **IP address**.
13. Enter the **port number** you provided for your **localhost** or **remote server host** in **Port**.

Note: If no port number was given (you accepted the default localhost:8009), accept the default value.

14. Select **Binary (AJP13)** for **Server type**.
15. Click **Next**.
16. Click **Add**.
17. Enter the name of your JSP file with a forward slash "/" before it and *.jsp* after for **URL path**. For example, */myjsp.jsp*.
18. Enter **webapps/app1** for **Application base directory**.

Note: Do not select the **Reloadable** option.

19. Click **Continue**.
20. Click **Next**.
21. Click **Finish**.
22. Click **OK**.

Place your JSP file in the correct directory: Using a file transfer method such as Netserver mapped drives or FTP, transfer the JSP file to the following directory: `/www/myserver/webapps/app1/jsp` (where *myserver* is your server's name).

Start HTTP Server and out-of-process ASF Tomcat server:

1. Click the **Manage** tab.
2. Click the **All Servers** subtab.
3. Click the **All HTTP Servers** tab.
4. Select your HTTP Server (powered by Apache) from the table.
5. Click **Start**.
6. Click the **All ASF Tomcat Servers** tab.
7. Select your out-of-process ASF Tomcat server from the table.
8. Click **Start**.

Test your configuration:

1. Start a Web browser.
2. Enter `http://myserver:port/app1/jsp/myjsp.jsp`, where *myserver:port* is the name of your iSeries and port number and *myjsp.jsp* is the name of your JSP file.

Set up servlets for out-of-process ASF Tomcat server on HTTP Server (powered by Apache)

This topic provides information about how to set up Java servlets for use with the Tomcat servlet engine with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Follow the instructions to use a servlet with out-of-process ASF Tomcat servlet engine on your HTTP Server (powered by Apache).

This topic assumes that you have an existing HTTP Server (powered by Apache) and you do not have a Tomcat configuration. Before you begin, review the "User profiles and required authorities for HTTP Server" on page 40 topic.

Configure the out-of-process ASF Tomcat:

1. Click the **Manage** tab.
2. Click the **ASF Tomcat Servers** subtab.
3. Expand **Tomcat Tasks and Wizards**.
4. Click **Create ASF Tomcat Server**.
5. Enter your worker name in the ASF Tomcat server name field. This is the name of the ASF Tomcat server. For example, *mytomcat*.
6. Click **Next**.
7. Accept the default value (QTMHHTTP) or enter a specific user id for **Server userid**.
8. Accept the default value for **Java version (JDK)**.
9. Accept the default value for **ASF Tomcat home**.
10. Accept the default value for **Java classpath entries**.
11. Click **Next**.
12. Accept the default value for **IP address**.

13. Enter the port number you provided for **Port**.

Note: If no port number was given (you accepted the default localhost:8009) accept the default value.

14. Select **Binary (AJP13)** for **Server type**.
15. Click **Next**.
16. Click **Add**.
17. Enter the name of your URL path for **URL path**. For example, */calc*.
18. Enter the name of your application base directory for **Application base directory**. For example, *webapps/example*.

Note: Do not select the **Reloadable** option at this time.

19. Click **Continue**.
20. Click **Configure**.
21. Click **Add**.
22. Enter the name of your servlet class file for **Servlet classname**.
If a jar file is used, enter the name of the servlet class in the jar file. For example, *CalculatorExample*.
23. Accept the default value for **URL patterns**.
24. Accept the default value for **Startup load sequence**.
If you are having trouble, one possible cause is incorrect authority and permissions. See "User profiles and required authorities for HTTP Server" on page 40 for more information.
25. Click **OK**.
26. Click **Next**.
27. Click **Finish**.
28. Click **OK**.

Configure HTTP Server (powered by Apache) for ASF Tomcat:

1. Start the IBM Web Administration for iSeries interface.
2. Click the **Manage** tab.
3. Click the **HTTP Servers** subtab.
4. Select your HTTP Server (powered by Apache) from the **Server** list.
5. Select **Global configuration** from the **Server area** list.
6. Expand **Server Properties**.
7. Click **ASF Tomcat Setup** task from the task list.
8. Select **Enable** from **Enable servlets for this HTTP Server**.
9. Accept the default workers definition file for **Workers definition file**.
10. Click **Next**.
11. Select **Enable** for **Enable "out-of-process" servlet engine connections**.

Note: Do not select the **Enable an "in-process" servlet engine** option at this time.

12. Click **Add**.
13. Enter the name of the out-of-process ASF Tomcat server for **Worker name**. For example, *mytomcat*.
14. Select **Binary (AJP13)** for **Worker type**.
15. Enter information based off one of the below conditions for **Hostname:Port**.
 - a. Enter **localhost** and a **port number** if your out-of-process ASF Tomcat server is on your iSeries. For example, *localhost:8009*.
 - b. Enter the **name of the remote server host** and the **port number** if your out-of-process ASF Tomcat server is not on your local iSeries. For example, *IBMiSeries:8009*.

16. Click **Continue**.
17. Click **Next**.
18. Click **Add**.
19. Enter the name of the URL mount point for **URL (Mount point)**. For example, */calc*.
20. Accept the default value for **ASF Tomcat worker**.
21. Click **Next**.
22. Click **Finish**.
23. Click **OK**.

Place your servlet file in the correct directory: Using a file transfer method such as Netserver mapped drives or FTP, transfer the servlet to one of the following directories:

- .class files
 - /ASFTomcat/myserver/webapps/example/WEB-INF/classes (where *myserver* is your server's name)
- .jar files
 - /ASFTomcat/myserver/webapps/example/WEB-INF/lib (where *myserver* is your server's name)

Start HTTP Server and out-of-process ASF Tomcat server:

1. Click the **Manage** tab.
2. Click the **All Servers** subtab.
3. Click the **All HTTP Servers** tab.
4. Select your HTTP Server (powered by Apache) from the table.
5. Click **Start** or **Restart**.
6. Click the **All ASF Tomcat Servers** tab.
7. Select your out-of-process ASF Tomcat server from the table.
8. Click **Start**.

Test your configuration:

1. Start a Web browser.
2. Enter **http://myserver:port/calc**, to test the configuration, where *myserver:port* is the name of your iSeries and port number.

Set up WAR files for out-of-process ASF Tomcat server on HTTP Server (powered by Apache)

This topic provides information about how to set up a Web Archive (WAR) file for use with the Tomcat servlet engine with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Follow the instructions to use a WAR file with out-of-process ASF Tomcat servlet engine on your HTTP Server (powered by Apache).

Before you begin, review the “User profiles and required authorities for HTTP Server” on page 40 topic.

Configure the out-of-process ASF Tomcat:

1. Click the **Manage** tab.
2. Click the **ASF Tomcat Servers** subtab.
3. Expand **Tomcat Tasks and Wizards**.

4. Click **Create ASF Tomcat Server**.
5. Enter a name for your ASF Tomcat server.
6. Click **Next**.
7. Accept the default value (QTMHHTTP) or enter a specific user id for **Server userid**.
8. Accept the default value for **Java version (JDK)**.
9. Accept the default value for **ASF Tomcat home**.
10. Accept the default value for **Java classpath entries**.
11. Click **Next**.
12. Accept the default value for **IP address**.
13. Enter the **port number** you provided for your localhost or remote server host for **Port**.

Note: If no port number was given (you accepted the default localhost:8009) accept the default value.

14. Select **Binary (AJP13)** for **Server type**.
15. Click **Next**.
16. Click **Add**.
17. Enter the name of your WAR file with a forward slash "/" before it for **URL path**. For example, */warexample*.
18. Enter **webapps** followed by a forward slash "/" and the name of your WAR file for **Application base directory**. For example, *webapps/warexample*.

Note: Do not select the **Reloadable** option at this time.

19. Click **Continue**
The WAR file contains a web.xml file. The first time the WAR file is accessed, the files within the WAR file are extracted into the correct directory.
20. Click **Next**.
21. Click **Finish**.
22. Click **OK**.

Configure HTTP Server (powered by Apache) for ASF Tomcat:

1. Start the IBM Web Administration for iSeries interface.
2. Click the **Manage** tab.
3. Click the **HTTP Servers** subtab.
4. Select your HTTP Server (powered by Apache) from the **Server** list.
5. Select **Global configuration** from the **Server area** list.
6. Expand **Server Properties**.
7. Click **ASF Tomcat Setup task** from the task list.
8. Select **Enable** from **Enable servlets fro this HTTP Server**.
9. Accept the default workers definition file for **Workers definition file**.
10. Click **Next**.
11. Select **Enable** from **Enable "out-of-process" servlet engine connections**.

Note: Do not select the **Enable an "in-process" servlet engine** option at this time.

12. Click **Add**.
13. Accept the default name or enter a new name of your ASF Tomcat server for **Worker name**.

Note: The name of the worker will be used below.

14. Select **Binary (AJP13)** for **Worker type**.

15. Enter information based off one of the below conditions for **Hostname:Port**.
 - a. Enter **localhost** and a **port number** if your out-of-process ASF Tomcat server is on your iSeries. For example, localhost:8009.
 - b. Enter the **name of the remote server host** and the **port number** if your out-of-process ASF Tomcat server is not on your local iSeries. For example, IBMiSeries:8009.
16. Click **Continue**.
17. Click **Next**.
18. Click **Add**.
19. Enter the name of your WAR file with a forward slash "/" before it for URL (Mount point). For example, /warexample.

Note: This name will be used below.
20. Accept the default value for **ASF Tomcat worker**. Select the worker defined by you above.
21. Click **Add**.
22. Enter the name of your WAR file with a forward slash "/" before it and after it, followed by an asterisk "*" for **URL (Mount point)**. For example, /warexample/*.

This URL (Mount point) is necessary so the files contained in your WAR file can be accessed.
23. Accept the default value for **ASF Tomcat worker**.
24. Click **Continue**.
25. Click **Next**.
26. Click **Finish**.
27. Click **OK**.

Place your WAR file in the correct directory: Using a file transfer method such as Netserver mapped drives or FTP, transfer the WAR file to the following directory: /ASFTomcat/[myserver]/webapps (where *myserver* is your server's name).

Start HTTP Server and out-of-process ASF Tomcat server:

1. Click the **Manage** tab.
2. Click **All Servers** subtab.
3. Click the **All HTTP Servers** tab.
4. Select your HTTP Server (powered by Apache) from the table.
5. Click **Start**.
6. Click the **All ASF Tomcat Servers** tab.
7. Select your out-of-process ASF Tomcat server from the table.
8. Click **Start**.

Test your configuration:

1. Start a Web browser.
2. Enter **http://myserver:port/warexample**, where *myserver:port* is the name of your iSeries and port number and *warexample* is the name of your WAR file.

Triggered cache manager tasks

This topic provides step-by-step tasks for the triggered cache manager.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Related information

“Triggered cache manager for HTTP Server” on page 46

This topic provides information about the triggered cache manager, page assembly, and wrappers.

“Trigger messages for triggered cache manager on HTTP Server (powered by Apache)” on page 49

This topic provides information about trigger requests and trigger messages.

Set up triggered cache manager on HTTP Server (powered by Apache)

This topic provides information about how to set up the triggered cache manager for your HTTP Server with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

The triggered cache manager (TCM) may be used with the default configuration or you may create your own customized configuration.

Default configuration: The following provides an overview of the process of configuring the triggered cache manager function with the default configuration.

You must write an application to send trigger messages to the triggered cache manager server. Messages your application sends cause the server to update, delete, or publish items in one or more cache targets (specified in the server configuration), using data retrieved from data sources (also specified in the server configuration). For more information on how to write a program to send trigger messages, see “Trigger messages for triggered cache manager on HTTP Server (powered by Apache)” on page 49.

Ensure that LPP 5722-DG1 Option 1 is installed on your iSeries system and HTTP *ADMIN server is up and running.

1. Start the IBM Web Administration for iSeries interface.
2. Click the **Advanced** tab.
3. Click the **TCM** subtab.
4. Click **Create server**.
5. Complete the following in the **Create server** form.

Field	Description
Server Name	Enter a name that you will later use to identify this server. For example, JKLTCServer
Autostart	Select Yes . This will cause your server to automatically startup when TCP/IP Services is started on your iSeries system. If you do not want your server to startup automatically, select No .
TCP Port	Use the default value or enter a new TCP Port. You will use this port to communicate with your new server. When the server is started it will establish a listening port on every IP interface configured for TCP/IP Services on your iSeries system. Enter a different port number if you do not want to use this default. Servers cannot share ports so if another server is already configured to use the default port (7049) then you must enter a different, unused port number for this server.

Field	Description
Options	Use the default, Create with default configuration , or select a new option. If you would rather copy the configuration of a server you have already configured, select Create based on existing configuration and then select that server name.

6. Click **Create**. The message, "Server 'JKLTCMServer' has been created with default configuration settings" will display at the bottom of your screen.
7. In the list on the left, click **Work with servers**.
8. Select the server that you just created (for example, JKLTCMServer).
9. Click **Start**. The message, "A request to start server 'JKLTCMServer' has been submitted" will display at the bottom of your screen. It may take a few minutes for the server to fully activate for the first time (click **Refresh** periodically until you see the server listed as **Active** under the **Status** column).

You have just created and started a triggered cache manager server that uses the default configuration settings provided by IBM. Servers using the default configuration settings are fully operational. IBM has established the following settings for a default configuration:

- A host description named **LOCALHOST** is defined for IP address 127.0.0.1. To view host descriptions, click **Hosts** in the list on the left.
IP address 127.0.0.1 is the 'loopback' interface. The loopback interface is used by many application servers to communicate with each other without using an actual physical network. In this case, the triggered cache manager server uses the loopback interface to communicate with a data source and potential cache target applications that may be added to the configuration (see the section titled **Custom configuration**, below, for details on how to add custom configuration setting to the default configuration).
- A data source description named **LOCAL_HTTP** is defined for a local HTTP server. To view data source descriptions, click on **Data sources** in the list on the left.
The local HTTP server is assumed to use the default HTTP port (80) on the loopback interface (described previously). The triggered cache manager server communicates with the local HTTP server over the loopback interface to request Web pages it needs to send (or publish) to cache targets.
- A cache target description named **LOCAL_DIRECTORY** is defined for the local IFS (the iSeries Integrated File System). To view cache target descriptions, click on **Cache targets** in the list on the left.
This description specifies the root (/) directory by default. The triggered cache manager server uses the local system's IFS as a cache target.
- Two trigger handler descriptions are set, one named **PUBLISH**, and the other named **UPDATE_CACHE**. To view trigger handler descriptions, click on **Trigger handlers** in the list on the left.
Both descriptions are set to establish trigger request handlers that manage the cache defined by **LOCAL_DIRECTORY** using data obtained from the data source defined by **LOCAL_HTTP**. The trigger request handler named **UPDATE_CACHE** simply updates cached items by copying them from data source to cache target(s), whereas the trigger request handler **PUBLISH** may be used to perform dependency parsing and page assembly prior to sending updated items to cache.
- An object dependency graph description named **DEFAULT** is defined for document publishing. To view object dependency graph descriptions, click on **Object dependency graphs** in the list on the left.
By default, the trigger request handler **PUBLISH** (described previously) is set to use the object dependency graph **DEFAULT** to record and query information pertaining to Web document dependency parsing and page assembly.

Your application must send trigger messages to one of the request handlers (either **UPDATE_CACHE** or **PUBLISH**). This type of application is referred to as an application trigger. When the server receives a

trigger message from an application trigger, it places the request in queue for the specified request handler and returns message code 1102 (indicating that the request was accepted). The server then continues to process the request asynchronously.

Servers running with a default configuration may be used in conjunction with application triggers and a local HTTP server (using port 80) to store dynamically produced Web pages as static files in the local file system. An HTTP server may then be used to serve these static files rather than the dynamically produced files, while application triggers ensure the cache of static files are updated only when necessary.

Custom configuration: The following section describes how to create customized configurations for the triggered cache manager function.

1. Follow the steps outlined in the previous section, “Default configuration” on page 186, to create and start a new server.
2. Select the server that you just created (for example, JKLTCServer).
3. Click **Hosts**.
4. Click **Create New Description** to add additional host descriptions to the configuration. If you don’t want to add additional host descriptions, skip to step 5.

Note: Host descriptions must be added before certain data source types or cache target descriptions may be added (steps 6d, 8d, and 8e).

- a. Enter the server (or servers) host name or IP address you want this server to use as a data source or cache target (for example, myserver.ibm.com).
- b. Click **Create**.
- c. To add more host descriptions, repeat steps 4 through 4b.
5. In the list on the left, click **Data sources**.
6. Click **Create New Description** to add additional data source descriptions to the configuration. If you don’t want to add additional data source descriptions, skip to step 7.

Note: Data source descriptions must be added before trigger handler descriptions can be added (re: steps 10c and 10d).

- a. Do the following on the first **Create data source description** form.

Field	Description
Name	Enter a name for this description. For example fields for a file system data source, or webDS for an HTTP Server data source.
Type	Select File System to define a local IFS directory as a data source. Select HTTP Server to define an HTTP server as a data source. For HTTP server data sources, the server hostname or IP address is provided by selecting a host description on the next screen. Host descriptions are not used when defining file system data sources.

- b. Click **Next**.
- c. If **File System** was chosen previously, do the following on the second **Create data source description** form. Otherwise, skip to the next step.

Field	Directory
Directory	Enter a directory name that is the data source. For example, /tcm/files All files requested from this data source will be relative to the specified directory.

Field	Directory
Threads	Use the default value or enter a new value. When started, the server will run in a multithreaded process on the iSeries system. It will use the number of specified threads to interact with the data source.

- d. If **HTTP Server** was chosen previously, do the following on the second **Create data source description** form.

Field	Description
Host	Select the description that describes the HTTP server data source hostname or IP address. Host descriptions were created in steps 4 through 4b. If there is no host description for the data source you are describing, you must first create it by repeating steps 4 through 4b.
TCP Port	Use the default value or enter a new value. If the HTTP Server data source is not using the default HTTP port, enter it here.
Root Directory	Select the default, (/). All URLs requested from this data source will be relative to the specified directory (path).
Keep Alive	Select No . If you know the data source supports persistent HTTP connections, commonly referred to as 'keepalive' support, you may want to select Yes to avoid having the server disconnect after each read request.
Timeout	Use the default value or enter a new value. Zero (0) indicates the server should wait indefinitely for responses from the data source after requests are sent. If you would rather the server eventually timeout rather than wait indefinitely, enter the number of seconds it should wait before dropping connections. Just remember that an HTTP server may take some time to respond to a request, especially if it needs to dynamically produce the requested file. Be sure to specify a timeout value that is acceptable, otherwise cache targets may not get updated.
Threads	Use the default value or enter a new value. When started, the server will run in a multithreaded process on the iSeries system. It will use the number of specified threads to interact with this data source. A data source thread is dedicated to fulfill one read request at a time. When threads are waiting for a response from a data source (see Timeout value above), it is not available to send other requests. Be sure to specify enough threads to provide for a number of simultaneous requests acceptable to the data source without overwhelming it.

- e. Click **Create**.
- f. To add more data source descriptions, repeat steps 6 through 6e.
7. Click **Cache targets**.
8. Click **Create New Description** to add additional cache target descriptions to the configuration. If you don't want to add additional cache target descriptions, skip to step 9.

- a. Do the following on the first **Create cache target description** form.

Field	Description
Name	<p>Enter a name for this description. For example fileCT for a file system cache target, or webCT for an HTTP Server cache target, or routerCT for a router cache target.</p> <p>This name is only used to refer to the description you are about to create. It is not used for communication with the cache target.</p>
Type	<p>Select File System to define a local IFS directory as a cache target.</p> <p>Select HTTP Server to define an HTTP server as a cache target. Select Router to define an IBM model 2212 or IBM model 2216 caching router as a cache target.</p> <p>For HTTP server and router cache targets, the server/router hostname or IP address is provided by selecting a host description on the next screen. Host descriptions are not used when defining file system cache targets.</p>

- b. Click **Next**.
- c. If **File System** was chosen previously, do the following on the second **Create cache target description** form. Otherwise, skip to the next step.

Field	Description
Status at Startup	<p>Select Enabled.</p> <p>The server creates an internal handler for each cache target description at startup. These handlers direct the server's interaction with cache targets. This option indicates the initial handler state when the server is started.</p> <p>Requests sent to a disabled cache target handler always appear to be handled successfully (when the handler actually does nothing). This allows you to take cache targets off-line without having failed cache update attempts logged by your triggered cache manager server. When cache targets go on-line again, their corresponding handlers may be re-enabled.</p>
Directory	<p>Enter a directory name that is the data source. For example: /tcm/filect</p> <p>All files requested from this cache target will be relative to the specified directory.</p>
Threads	<p>Use the default value or enter a new value.</p> <p>When started, the server will run in a multithreaded process on the iSeries system. It will use the number of specified threads to interact with this cache target.</p>

- d. If **HTTP Server** was chosen previously, do the following on the second **Create cache target description** form. Otherwise, skip to the next step.

Field	Description
Status at Startup	<p>Select Enabled.</p> <p>The server creates an internal handler for each cache target description at startup. These handlers direct the server's interaction with cache targets. This option indicates the initial handler state when the server is started.</p> <p>Requests sent to a disabled cache target handler always appear to be handled successfully (when the handler actually does nothing). This allows you to take cache targets off-line without having failed cache update attempts logged by your triggered cache manager server. When cache targets go on-line again, their corresponding handlers may be re-enabled.</p>

Field	Description
Host	Select the description that describes the HTTP cache target source hostname or IP address. Host descriptions were created in steps 4 through 4b. If there is no host description for the cache target you are describing, you must first create it by repeating steps 4 through 4b.
TCP Port	Use the default value or select a new value. If the HTTP server cache target is not using the default HTTP port, enter it here.
Root Directory	Use the default, (/). All URLs posted to this cache target will be relative to the specified directory (path).
Keep Alive	Select No . If you know the cache target supports persistent HTTP connections, commonly referred to as 'keepalive' support, you may want to select Yes to avoid having the server disconnect after each post request.
Timeout	Use the default value or enter a new value. Zero (0) indicates the server should wait indefinitely for responses from the cache target after requests are sent. If you would rather the server eventually timeout rather than wait indefinitely, enter the number of seconds it should wait before dropping connections. Just remember that an HTTP server may take some time to respond to a request, especially if it needs to dynamically produce the requested file. Be sure to specify a timeout value that is acceptable, otherwise you cannot determine if the cache target received the request.
Threads	Use the default value or enter a new value. When started, the server will run in a multithreaded process on the iSeries system. It will use the number of specified threads to interact with this cache target. A cache target thread is dedicated to fulfill one post request at a time. When threads are waiting for a response from a cache target (see Timeout value above), it is not available to send other requests. Be sure to specify enough threads to provide for a number of simultaneous requests acceptable to the cache target without overwhelming it.

- e. If **Router** was chosen previously, do the following on the second **Create cache target description** form.

Field	Description
Status at Startup	Select Enabled . The server creates an internal handler for each cache target description at startup. These handlers direct the server's interaction with cache targets. This option indicates the initial handler state when the server is started. Requests sent to a disabled cache target handler always appear to be handled successfully (when the handler actually does nothing). This allows you to take cache targets off-line without having failed cache update attempts logged by your triggered cache manager server. When cache targets go on-line again, their corresponding handlers may be re-enabled.
Host	Select the description that describes the hostname or IP address of the router hosting the web document cache target. Host descriptions were created in steps 4 through 4b. If there is no host description for the cache target you are describing, you must first create it by repeating steps 4 through 4b.
TCP Port	Use the default value or enter a new value. If the router is not using the default port, enter it here.

Field	Description
Cluster ID	<p>Enter the fully qualified host name or IP address of the web server cluster for which the router is caching documents.</p> <p>This name applies to the name of the web document cache, not the router that is hosting the cache. The router, specified by the Host field, may contain multiple caches for different web server clusters. Use this field to specify which web document cache you want to manage.</p>
Cluster TCP Port	<p>Enter the TCP port number associated with the cluster ID specified above.</p> <p>The server combines the cluster ID and the cluster TCP port to uniquely identify a cache target hosted by the router.</p>
Root Directory	<p>Use the default, (/).</p> <p>All URLs posted to this cache target will be relative to the specified directory (path).</p>
Keep Alive	<p>Select No.</p> <p>If you know the cache target supports persistent HTTP connections, commonly referred to as 'keepalive' support, you may want to select Yes to avoid having the server disconnect after each post request.</p>
Threads	<p>Use the default, 5.</p> <p>When started, the server will run in a multithreaded process on the iSeries system. It will use the number of specified threads to interact with this cache target.</p> <p>A cache target thread is dedicated to fulfill one post request at a time. When threads are waiting for a response from a cache target it is not available to send other requests. Be sure to specify enough threads to provide for a number of simultaneous requests acceptable to the cache target without overwhelming it.</p>

- f. Click **Create**.
- g. To add more cache target descriptions, repeat steps 8 through 8f.
9. Click **Trigger handlers**.
10. Click **Create New Description** to add additional trigger handler descriptions to the configuration. If you don't want to add additional trigger handler descriptions, skip to step 11.
 - a. Do the following on the first **Create trigger handler description** form.

Field	Description
Name	<p>Enter a name for this description. For example simpleDocUpdate for an update cache trigger handler, or docPublisher for a publish trigger handler.</p> <p>The name you choose is used by your application triggers when sending messages to the server. All trigger messages sent to the server must contain the name of the trigger handler that is to process the request.</p>

Field	Description
Type	<p>Select Update Cache to define a trigger handler that does simple cache updates.</p> <p>Select Publish to define a trigger handler that does document publishing.</p> <p>Update Cache trigger handlers perform simple data transfers. They retrieve objects from a data source and copy them to cache targets.</p> <p>Publish trigger handlers may use object dependency graphs and rule sets to perform dependency parsing and page assembly (prior to doing cache updates). You may use the DEFAULT object dependency graph description included in the default configuration, or you may create and use a new one specifically for your handlers.</p> <p>For more information, click on Publishing rules and Rule set in the list on the left, then click on the help icon in the top-right corner of these pages. If you decide to create your own publishing rules and rule set(s) you may do so, and then return to step 9 to add publish trigger handler descriptions to reference your new rule set(s).</p>

- b. Click **Next**.
- c. If **Update Cache** was chosen previously, do the following on the second **Create trigger handler description** form. Otherwise, skip to the next step.

Field	Description
Data Source	<p>Select the description that describes the data source for this trigger handler.</p> <p>Trigger handlers go to one, and only one, data source to retrieve information. The same data source, however, may be used by more than one trigger handler. Data source descriptions were created in steps 6 through 6f. If there is no description for the data source you want your handler to use, you must first create it by repeating steps 6 through 6f.</p>
Cache Targets	<p>Select the descriptions that describe the cache targets for this trigger handler, or select no descriptions to create a trigger handler that retrieves data but does not sent it to cache.</p> <p>Trigger handlers can manage data for multiple cache targets. Moreover, the same cache target can be managed by more than one trigger handler. Cache target descriptions were created in steps 8 through 8g. If there is no descriptions for the cache targets you want your handler to manage, you may first create them by repeating steps 8 through 8g, or continue on to create this description and then come back to include them later.</p>
Trigger Queue Collapse Policy	<p>Use the default value.</p> <p>Trigger messages sent from application triggers to a server are always placed on a particular trigger handler's request queue. The handler then processes the requests asynchronously according to the queuing policy specified within the message. It is possible for identical trigger messages to be in queue, either due to the queuing policy or due to the server being overwhelmed by requests. This setting defines how identical triggers in queue are to be handled.</p>
Cache Request Queue Priority	<p>Use the default value.</p> <p>The same cache target can be managed by more than one trigger handler. Cache update requests from trigger handlers are queued and processed according to priority. This setting defines the priority this handler is to specify when placing cache update requests. The lower the number, the higher the priority. The default is the lowest priority. Enter a lower number if you want this handler to have a higher priority.</p>
Threads	<p>Use the default value or enter a new value.</p> <p>When started, the server will run in a multithreaded process on the iSeries system. It will use the number of specified threads to process requests sent to this trigger handler.</p>

Field	Description
Success/ Failures	<p>Use the default value.</p> <p>Trigger handlers can post success or failure messages to acknowledgment targets after requests are handled. This setting allows you to specify the two acknowledgment target lists this handler uses. One list is used to send success messages; the other is used to send failure messages.</p> <p>Default configuration settings do not include acknowledgment target descriptions. If you want your trigger handler to post completion messages, you must first write an application that listens for such messages (using HTTP POST method). You can then add an acknowledgment target description to this server's configuration and reference it here.</p>

- d. If **Publish** was chosen previously, do the following on the second **Create trigger handler description** form.

Field	Description
Data Source	<p>Select the description that describes the data source for this trigger handler.</p> <p>Trigger handlers go to one, and only one, data source to retrieve information. The same data source, however, may be used by more than one trigger handler. Data source descriptions were created in steps 6 through 6f. If there is no description for the data source you want your handler to use, you must first create it by repeating steps 6 through 6f.</p>
Cache Targets	<p>Select the descriptions that describe the cache targets for this trigger handler, or select no descriptions to create a trigger handler that retrieves data but does not sent it to cache.</p> <p>Trigger handlers can manage data for multiple cache targets. Moreover, the same cache target can be managed by more than one trigger handler. Cache target descriptions were created in steps 8 through 8g. If there is no descriptions for the cache targets you want your handler to manage, you may first create them by repeating steps 8 through 8g, or continue on to create this description and then come back to include them later.</p>
Default Include Object	<p>Use the default value.</p> <p>All document component must be triggered before they can be processed by a publish trigger handler. This setting defines an object (by name) that is included during page assembly when neither the source object nor the specified default object in a %fragment tag has been triggered. The default value, none, indicates that such a global default include object is not specified for this handler.</p>
Object Dependency Graph	<p>Select the DEFAULT description.</p> <p>Publish trigger handlers only use one object dependency graph to record and query object dependency information. The same graph, however, may be used by more than one publish trigger handler.</p>
Edge Type to Traverse	<p>Use the default value.</p> <p>When publish trigger handlers query an object dependency graph to determine dependency relationships between document components, they traverse a particular edge type (by name). Other edge types, perhaps added by other handlers, are ignored. This setting defines the name of the edge type this handler is to build and traverse. If not specified (the default) a system supplied edge type is used.</p>

Field	Description
Rule Set	<p>Use the default value.</p> <p>Publish trigger handlers abide by a set of publishing rules when publishing documents. The same rule set, however, may be used by more than one publish trigger handler. If not specified (the default) the default publishing rule is used for all components.</p> <p>For more information, click on Publishing rules and Rule set in the list on the left (after creating this trigger handler description), then click on the help icon in the top-right corner of these pages. If you decide to create your own publishing rules and rule sets you may then change this trigger handler description to reference your new rule sets.</p>
Threads	<p>Use the default value.</p> <p>When started, the server will run in a multithreaded process on the iSeries system. It will use the number of specified threads to process requests sent to this trigger handler.</p>
Success/ Failures	<p>Use the default value.</p> <p>Trigger handlers can post success or failure messages to acknowledgment targets after requests are handled. This setting allows you to specify the two acknowledgment target lists this handler uses. One list is used to send success messages; the other is used to send failure messages.</p> <p>Default configuration settings do not include acknowledgment target descriptions. If you want your trigger handler to post completion messages, you must first write an application that listens for such messages (using HTTP POST method). You can then add an acknowledgment target description to this server's configuration and reference it here.</p> <p>The Include object dependency information in acknowledgments option may be used.</p>

- e. Click **Create**.
- f. To add more trigger handler descriptions, repeat steps 10 through 10e.
11. Click **Work with servers**.
12. Select the server that you just customized. For example: JKLTCMServer.
13. If your server is listed as "Active" under the Status column, click Stop. Otherwise skip to step 15. The message, "A request to end server 'JKLTCMServer' has been submitted" will display at the bottom of your screen. It may take a few moments for the server end (click **Refresh** periodically until you see the server listed as "Not Active").
14. Select the server that you just customized. For example: JKLTCMServer.
15. Click **Start**.
16. The message, "A request to start server 'JKLTCMServer' has been submitted" will display at the bottom of your screen. It may take a few minutes for the server to fully activate for the first time (click **Refresh** periodically until you see the server listed as "Active" under the Status column).

You have just created, configured, and started a triggered cache manager server using a custom configuration. Your application triggers may now send trigger messages to the active server to request that documents be updated, deleted, or published to the cache targets specified in the configuration.

WebDAV tasks

This topic provides step-by-step tasks for Web-based distributed authoring and versioning (WebDAV).

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Related information

“WebDAV for HTTP Server (powered by Apache)” on page 64

This topic provides information about Web-based distributed authoring and versioning (WebDAV).

Set up WebDAV for HTTP Server (powered by Apache)

This topic provides information about how to set up WebDAV for your HTTP Server with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 (powered by Apache)” on page 64 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 (powered by Apache) 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**.

WebSphere tasks

This topic provides step-by-step tasks for managing WebSphere applications.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Manage WebSphere Application Server

V5R3 is shipped with a pre-configured WebSphere Application Server - Express Version 5 application server instance referred to as the System Application Server Instance (SYSINST). This topic describes how to start SYSINST.

The SYSINST has the following IBM supplied system administrative web applications pre-installed, providing an easy-to-use web GUI interface to administration tasks:

- iSeries Navigator Tasks for the Web
- Access core systems management tasks and access multiple systems through one iSeries from a web browser
- Tivoli® Directory Server Web Administration Tool
- Setup new or manage existing (LDAP) directories for business application data.

The SYSINST is not started by default when V5R3 is installed. Before you begin working with the above functions, the Administration instance of the HTTP Server (port 2001) must be running on your system. The above administrative web applications are accessible after the SYSINST is started. Perform these steps to start the SYSINST.

Set up the system instance to start when the ADMIN HTTP server starts.

1. Using a Web browser, go to the IBM HTTP Server Configuration and Administration forms by performing these steps:
 - a. Open this URL in your browser:
`http://your.server.name:2001/`

where *your.server.name* is the name of your iSeries server.
 - b. Enter a valid user ID and password for your iSeries server.
 - c. At the **iSeries Tasks** page, select **IBM Web Administration for iSeries** .
2. Click the **Manage** tab and the **HTTP Server** tab.
3. Select **Admin** from the **Server** drop-down list.
4. Under **Server Properties**, select **General Server Configuration**.
5. On the **General Server Configuration** page, click the **General Settings** tab.
6. Select **Yes** in the **Start the system application server instance when the 'Admin' server is started** field.
7. The **Stop the system application server instance when the 'Admin' server is stopped** field displays. It is recommended that you select **Yes** to stop the system application server instance when the ADMIN server is stopped. Click **OK**.
8. Restart the administrative instance, and both the administrative HTTP server and the SYSINST are started.

(Optional) Start the system instance independent of the ADMIN HTTP server.

If you have already set up the system instance using the previous steps, you can perform these steps to start the system instance independent of the ADMIN HTTP server.

1. Using a Web browser, go to the IBM HTTP Server Configuration and Administration forms by performing these steps:
 - a. Open this URL in your browser:
`http://your.server.name:2001/`


where *your.server.name* is the name of your iSeries server.
 - b. Enter a valid user ID and password for your iSeries server.
 - c. At the **iSeries Tasks** page, select **IBM Web Administration for iSeries** .

2. Click the **Manage** tab and the **Application Servers** tab.
3. Select the system application server instance from the **Instance/Server** drop-down list.
4. Click **Start** to start SYSINST.

WebSphere Portal wizard worksheet

This topic helps you with the Create WebSphere Portal wizard in the IBM Web Administration for iSeries interface.

Note: See the “Requirements” on page 199 in this document for information on all required PTFs, software, and hardware.

It is recommended that you use the **Create WebSphere Portal wizard** in the IBM Web Administration for iSeries 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 iSeries is located in the WebSphere Portal Product Documentation  Web site.

The Create WebSphere Portal wizard. 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 iSeries.

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 consists of the middleware, portlets, and development tools for building and managing secure business portals. A portal is a Web site that provides end users with a single point of access to Web-based resources. For more information on WebSphere Portal, see the WebSphere Portal Product Documentation  Web site.

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 Directory Services (LDAP).

Database

DB2 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 DB2 Universal Database™ for iSeries.

Portlet deployment

The wizard deploys portlets to your Portal server.

Before you begin


1. Make sure you have the required software and PTFs installed. See “Requirements” on page 199.
2. Plan to set aside enough time for the wizard to complete the WebSphere Portal server configuration. Depending on your iSeries server and if you are accessing remote iSeries servers for DB2 or LDAP, the process may take an hour or more once you have clicked the **Finish** button in the wizard.

Note: The installation of PTFs for products required by WebSphere Portal might require at least one initial program load (IPL or restart) of the server, so plan appropriately

3. You must have a valid user ID and password on the local iSeries server. The user profile must at least have *ALLOBJ, *IOSYSCFG, and *JOBCTL special authority to install and configure the WebSphere Portal server.
4. You must have a database user profile, a user profile that owns the database, with at least *USER authority on the iSeries server where the DB2 is located to use an existing database user profile. If you have *SECADM authority, you can create a new database user profile through the wizard. If using a remote iSeries server for the Portal server database, you must have a user profile with at least *USER authority on the remote iSeries server.
5. Verify the following network setup requirements before installation:
 - **Static IP address:** WebSphere Portal requires that you use a static IP address on the machine where you are installing.
 - **Fully-qualified host name:** WebSphere Portal also requires the use of a fully-qualified host name, which is typically the host name of the iSeries server, along with its fully-qualified domain name. To be sure that this is configured correctly, you can check with a simple ping before you start installation. For example, you could enter the following command at a command prompt: `ping yourserver.yourcompany.com`

Requirements

Minimum hardware requirements

You can use the IBM 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.

- iSeries Model 810 with processor feature 2465
- 2 GB of memory
- 1.5 GB of disk space for installation of WebSphere Portal and WebSphere Application Server V5.0 Enterprise Enablement
- 500 MB of disk space for each WebSphere Application Server instance configured for use with WebSphere Portal

Note: These requirements represent the recommended minimum requirements. Deployments which must support many users or require shorter response times might require additional resources. These minimum requirements are based on a non-cluster environment running a single WebSphere Application Server instance. To run multiple WebSphere Application Server instances at the same time, additional system resources are required.

Required software

The following software is required. Items noted with a CD number following their description are included with WebSphere Portal. All other required and optional software must be obtained separately.

- 5722SS1 OS/400 V5R2 or 5722SS1 OS/400 V5R3
- 5722JV1 (Option *BASE) IBM Developer Kit for Java
- 5722JV1 (Option 5) Dev Toolkit for Java (Version 1.3), requires the *BASE option
- 5722SS1 (Option 12) Host Servers
- 5722SS1 (Option 30) Qshell Interpreter
- 5722SS1 (Option 33) Portable Application Solution Environment (PASE)
- 5722DG1 IBM HTTP Server for iSeries




- 5722TC1 TCP/IP Utilities
- 5733WS5 (Option *BASE) WebSphere Application Server V5.0 (CD 1-1)
- 5733WS5 (Option 1) WebSphere Application Server V5.0 Client development and runtime (CD 1-1)
- 5733WS5 (Option 2) WebSphere Application Server V5.0 Application server runtime (CD 1-2)
- 5733WS5 (Option 10) WebSphere Application Server V5.0 Enterprise Enablement (CD 1-12)

Note: The WebSphere Application Server V5.0.2 PTF Group must be installed, and the system must be restarted prior to installing WebSphere Application Server V5.0 Enterprise Enablement.

Required PTFs

Note: The Create WebSphere Portal wizard in the IBM Web Administration for iSeries interface has a built-in dependency checklist. If your system does not have all of the required PTFs installed, a message displays with a link to a detailed list of missing requirements. The wizard does not allow you to continue until the missing PTFs are applied.

See these websites for a complete list of all required PTFs.

- See the WebSphere Portal for iSeries  website for a partial list of iSeries specific PTFs.
- See the WebSphere Portal product website  for a partial list of required PTFs.
- Additional required PTFs are provided on CD 1-13. Refer to the Release Notes at Release Notes  for this list.

To determine if the correct PTF packages are installed, perform the following steps:

1. Sign onto your iSeries server.
2. Enter the Display PTF Status (DSPPTF) command on an OS/400 command line. The Display PTF Status screen is displayed. This screen lists the PTFs that have been applied to your server.
3. Enter the Work with PTF Groups (WRKPTFGRP) command on an OS/400 command line. The Work with PTF Groups status screen is displayed. This screen lists the PTF group level and what group PTFs have been applied to your server.

Note: See the IBM eServer™ iSeries Support Fixes  for instructions on ordering PTFs.

Planning worksheet

The planning worksheet lists the information needed for installing the WebSphere Portal server and its components. Fill in the table with values appropriate for your iSeries server and do not assume the default values listed within the table are correct. When possible, the Create WebSphere Portal wizard provides iSeries and LDAP server system values for you. These values are noted in the planning worksheet.

Form and field name	Write your value here
Step 1: Create WebSphere Application Server for the Portal - Specify Name	
<p>The first step is to create a new WebSphere Application Server for your WebSphere Portal server. The wizard creates a new WebSphere Application Server for you with the specified name. You cannot use an existing application server. The name must be unique within your WebSphere Application Server installation. If the name you type is currently used by another application server, the wizard displays a warning. The wizard creates the application server, updates the virtual host information for the HTTP Server (that is specified in the next step), updates the Web server plug-in, and creates the necessary JDBC providers and datasources that are required for WebSphere Portal.</p>	



Form and field name	Write your value here
<ul style="list-style-type: none"> • Application server name: Specifies a name for the application server that identifies the application server. The name can be any alphanumeric value. Names cannot contain a blank or any of the following special characters: &!@%*:/\#;?;=^<> +' " () { } [] . The name must be unique. A default application server name is entered for you. This default value is unique. Example: <i>WAS5Portal5</i> 	
<ul style="list-style-type: none"> • Server description: Specifies a description for the application server. This field is optional and is only used to identify your application server throughout the IBM Web Administration for iSeries interface. A default description is entered for you. This field is optional. Example: <i>WebSphere Portal server WAS5Portal5, created by the portal wizard</i> 	
<p>Step 2: Select HTTP Server Type</p> <p>The application server requires an association with an HTTP Server. The HTTP Server routes incoming requests from Web clients to the application server. Using an external HTTP Server is the recommended method for a production Web server. The application server has its own internal HTTP Server, but it is recommended that it only be used for development as it is not enabled for production-level security. Using an HTTP Server (powered by Apache) gives you control of the security of your server. If you have a firewall, you can put the HTTP Server (powered by Apache) outside the firewall so only specific requests are enabled to reach your application server, providing for a greater degree of security.</p> <p>Choose to create a new HTTP Server (powered by Apache) (option A) or use an existing HTTP Server (powered by Apache) already configured on your iSeries server (option B).</p>	
<ul style="list-style-type: none"> • Option A: Create a new HTTP server (powered by Apache): Enables you to create a new HTTP Server to be used with your WebSphere Portal server. Select this option to create a new HTTP Server to use with your WebSphere Portal server. If you select this option, skip to Step 3: Create a new HTTP Server (powered by Apache) to continue. 	
<ul style="list-style-type: none"> • Option B: Select an existing HTTP Server (powered by Apache): Enables you to select an existing HTTP Server previously configured on your iSeries server. Select this option to use an existing HTTP Server instead of creating a new HTTP Server. By choosing this option, the existing HTTP Server is now associated with your new WebSphere Portal server. Any other application server that was previously configured to use this particular HTTP Server will need to be reconfigured. If you select this option, skip to Step 4: Select an existing HTTP Server (powered by Apache) to continue. 	
<p>Step 3: Create a new HTTP Server (powered by Apache)</p> <p>The new HTTP Server is created with all the necessary directives to be ready for production. The server is configured with the plug-in directives which routes URL requests from the HTTP Server to the application server. The name of the HTTP Server must be unique. If the name you type is in use by another HTTP Server, the wizard displays a warning.</p>	
<ul style="list-style-type: none"> • HTTP Server name: Specifies a name for the HTTP Server that identifies the server. The name can be any alphanumeric value no more than 10 characters long. Names cannot contain a blank or any of the following special characters: &!@%*:/\#;?;=^<> +' " () { } [] . The name must be unique. Example: <i>MYHTTPSRVR</i> 	

Form and field name	Write your value here
<ul style="list-style-type: none"> • HTTP Server description: Specifies a description for the HTTP Server. This field is optional and is only used to identify your HTTP Server throughout the IBM Web Administration for iSeries interface. The default for this field is <i>HTTP Server created by the Create Application Server Wizard</i>. Example: <i>This is the HTTP Server for WebSphere Portal.</i> 	
<ul style="list-style-type: none"> • IP address: Specifies the IP address the HTTP Server uses. Possible values include any valid IP address or All IP addresses. The default is All IP addresses. If All IP addresses is selected, the HTTP Server listens on the designated port for all IP addresses on the system. You can only specify the IP addresses that are currently configured to this iSeries. If you have multiple IP addresses configured for this machine (for example, one for your external requests and one for internal requests), you can limit this server to listen to only one of them by selecting the desired IP address from the list. 	
<ul style="list-style-type: none"> • Port: Specifies the TCP/IP port number the HTTP Server will be listening on. The port number is used to link the incoming data to the correct application service. The wizard reviews the list of ports being used on the system and recommends a port number currently not in use. The wizard starts looking at port number 10000 for free ports. Most browsers send request to port 80 by default. For production situations, it is recommended to use port number 80. Specify port 443 for secure HTTP (HTTPS) transactions. The wizard does not configure SSL. The wizard verifies that another server is not using any of the ports in the specified range. The wizard checks the list of active ports on the system. It also examines all the HTTP Servers that are configured, any ASF Tomcat servers that might be configured, and all WebSphere Application Servers configured on this system. If another server is configured using any of the ports in the specified range, an error message is displayed and you cannot proceed. 	
<p>Step 4: Select an existing HTTP Server (powered by Apache)</p> <p>Select an existing HTTP Server from the table. The HTTP Server is configured to route URL requests to the application server. The plug-in directives are reconfigured to point to the application server that is being created in this wizard.</p> <p>Note: If the HTTP Server is already configured for a different WebSphere Application Server, it is reconfigured for the new WebSphere Application Server. An HTTP Server can route requests to only one application server. If the HTTP Server is already configured for a different application server, it is reconfigured for the new application server. Any application server previously configured for an existing HTTP Server no longer receives HTTP requests.</p>	
<p>Step 5: Specify Internal Ports Used by the Application Server</p> <p>A range of consecutive is reserved for the application server internal services only. The number you specify is the first of a range of 12 to 13 consecutive numbers. For example, if you specify 10219, the numbers 10219 to 10230 or 10219 to 10231 are assigned. All of the consecutive ports must be free.</p>	

Form and field name	Write your value here
<ul style="list-style-type: none"> • First port in range: Specifies the range of port numbers your application server uses for internal services. The port number is used to link the incoming data to the correct application service. The default value is the first port number in the range of consecutive ports that are not currently in use by your system. Example: 10219 Note: Only one service can run on a port at a time. The wizard verifies that the port range you specify is correct and the port range is not in use. The wizard verifies that another server is not using any of the ports in the specified range. The wizard checks the list of active ports on the system. It also examines all the HTTP Servers that are configured, any ASF Tomcat servers that might be configured, and all WebSphere Application Servers configured on this system. If another server is configured using any of the ports in the specified range, an error message is displayed and you cannot proceed. 	
<p>Step 6: Create DB2 Database for Portal</p> <p>The WebSphere Portal server requires a database to store information about user identities, credentials, permissions for accessing portal resources, customized Web pages, and other customized portal information. The wizard creates a DB2 database on the local iSeries server or on a remote iSeries server.</p> <p>Choose to create a new collection, schema, or library on your local iSeries server (option A) or create a new collection, schema, or library on a remote iSeries (option B).</p>	
<ul style="list-style-type: none"> • Option A: Create a new collection on this local iSeries: Enables you to create a new database collection, schema, or library on a local iSeries server database. 	
<ul style="list-style-type: none"> • Collection, Schema or Library name: Specifies the name of your new collection, schema, or library. The name can be any alphanumeric value no more than 10 characters long. The name must be unique. The wizard recommends a unique default value. If a collection, schema, or library with the name specified currently exists, an error message is displayed and you cannot proceed. Example: <i>PortalDB</i> 	
<ul style="list-style-type: none"> • Option B: Create a new collection on a remote iSeries: Enables you to create a new database collection, schema, or library on a remote iSeries server database. If you select this option, skip to Step 8: Create a default URL path, portal path, and personalized path to continue. Note: You must have a user ID on the remote iSeries server that has *USER authority to the Change Job (CHGJOB) command to create a new collection, schema, or library. 	
<ul style="list-style-type: none"> • Collection, Schema or Library name: Specifies the name of your new collection, schema, or library. The name can be any alphanumeric value no more than 10 characters long. The name must be unique. The wizard recommends a unique default value. If a collection, schema, or library with the name specified currently exists, an error message is displayed and you cannot proceed. Example: <i>PortalDB</i> 	

Form and field name	Write your value here
<ul style="list-style-type: none"> • System: Specifies the IP address or the hostname of the remote iSeries server where the database is to be created. If the remote iSeries cannot be accessed, an error message is displayed and you cannot proceed. Example: <i>myiseries.com</i> 	
<ul style="list-style-type: none"> • iSeries userid: Specifies a valid iSeries user ID that can access the remote iSeries server and database. Note that the user ID is also the iSeries user profile that owns the remote iSeries database after it is created. It is recommended that you specify a special user ID whose only purpose is to own the remote database. This prevents the remote database from being associated with a specific user whose user ID might be removed in the future. The specified user ID must have a password associated with it. Do not change the user ID or password for the remote database at a later date. If the remote database owner user ID profile or password is edited, removed or disabled, your WebSphere Portal server stops working. It is recommended that you create a profile called <i>wpsdbuser</i>. 	
<ul style="list-style-type: none"> • Password: Specifies the password for the iSeries user ID typed in the iSeries userid field. 	
<p>Step 7: Specify User ID to Own the Portal Database</p> <p>The local database created for the WebSphere Portal server requires an iSeries user profile to own the database. It is recommended that you specify a special user ID whose only purpose is to own the database. This prevents the database from being associated with a specific user whose user ID might be removed in the future. The specified user ID must have a password associated with it. Do not change the user ID or password for the database at a later date. If the database owner user ID profile or password is edited, removed or disabled, your WebSphere Portal server stops working. It is recommended that you create a profile called <i>wpsdbuser</i>.</p> <p>Choose to use an existing user ID (option A) or create a new user ID (option B).</p>	
<ul style="list-style-type: none"> • Option A: Use an existing user ID on this local system: Enables you to specify an existing user ID and password. 	
<ul style="list-style-type: none"> • User name: Specifies an existing user ID. The name can be any alphanumeric value no more than 10 characters long. By default, <i>wpsadmin</i> is used. Note: You must provide a valid and existing iSeries user ID and password, or an error is issued and you cannot proceed. 	
<ul style="list-style-type: none"> • Password: Specifies the password for the iSeries user name typed in the User name field. 	

Form and field name	Write your value here
<ul style="list-style-type: none"> • Option B: Create a new user ID on this local system: Enables you to create a new user ID and password. The user ID is created with *USER class authority. Note: The create a new user ID on a local system option is available only if the user currently signed on to the GUI has *SECADM special authority. 	
<ul style="list-style-type: none"> • User name: Specifies the new user ID. The name can be any alphanumeric value no more than 10 characters long. By default, <i>wpsadmin</i> is used. 	
<ul style="list-style-type: none"> • Password: Specifies the password for the iSeries user name typed in the User name field. 	
<ul style="list-style-type: none"> • Confirm password: Specifies the new iSeries user name password a second time for confirmation. 	
Step 8: Create a default URL path, portal path, and personalized path	
<p>To access the WebSphere Portal server you need to provide a base URI for the default URL path. Provide a base URI for the default URL path to access the WebSphere Portal server. The values are customizable to better suit the needs of your organization. Any URL that starts with this URI is reserved for the WebSphere Portal server. In addition, the WebSphere Portal server default home path is used to access the WebSphere Portal home Web page. The personalized path is the path to the WebSphere Portal server user's customized pages. Use the forward slash (/) to distinguish different directory levels (example, <i>/myportal/users/home</i>). The wizard creates the directories for you. Note: When specifying the Default URL path (context root), do not specify a value that is the same as a directory existing in a portlet's WAR directory. For example, if you set the context root for WebSphere Portal to be <i>/images (/wps</i> is the default value) and there is also a portlet with the directory structure <i>/myPortlet.ear/myPortlet.war/images</i>, this could cause a conflict if the portlet encoded URI references to resources in its own <i>/images</i> directory. In this situation, the portlet would be unable to display images because WebSphere Portal would look for the image resources according to its own context root path instead of the directory path specified by the portlet's WAR file.</p>	
<ul style="list-style-type: none"> • Default URL path: Specifies the context root or base URI for the WebSphere Portal server. All URLs beginning with this path are reserved for the WebSphere Portal server. This value is part of the URL used to access WebSphere Portal from a browser. By default, <i>wps</i> is used. Example: <i>http://hostname.yourco.com:9081/wps/portal</i> 	
<ul style="list-style-type: none"> • Default home path: Specifies the default WebSphere Portal server home path. This URL is used to access the WebSphere Portal home page. This is the Web page for users who are not logged in. This value is part of the URL used to access WebSphere Portal from a browser. By default, <i>portal</i> is used. Example: <i>http://hostname.yourco.com:9081/wps/portal</i> 	
<ul style="list-style-type: none"> • Personalized path: Specifies the personalized path URL. The personalized path is the URL for user's customized pages. This is the portal Web page for users who have already logged into the portal. This page cannot be accessed by anonymous users. This value is part of the URL used to access WebSphere Portal from a browser. By default, <i>myportal</i> is used. Example: <i>http://hostname.yourco.com:9081/wps/myportal</i> 	


Form and field name	Write your value here
<p>Step 9: Configure Proxy Information</p> <p>Some intranet security configurations require a proxy server for accessing content outside the Intranet. A proxy server is used to enable users to access Web sites outside of firewalls, to provide better access time by using caching, to provide logging and tracking of internet usage, or to limit port access. If you are unsure whether your company uses a proxy server, contact your network administrator. If you specify to use a proxy server, all portlets that use the Content Access Service (CAS) to retrieve data from sources that reside outside your local network use this proxy server to retrieve data.</p> <p>WebSphere Portal comprises a highly configurable framework of services to accommodate the different scenarios that portals need to address. The framework enables convenient replacement of service implementations as well as modification to the configuration of each service. Content Access Service is one of the services provided by the configurable framework of services.</p> <p>Portlets can access content from a remote system that is located on the other side of a firewall by invoking CAS. Not all portlets use CAS. You may need to modify portlets individually to use your proxy server to obtain external data. Some portlets included with the Business portlets, such as the Pinnacor portlets (My News, My Weather, My Stocks, etc.) and the Financial Times portlets, do not use CAS. Refer to the "Troubleshooting Portlets" section of the WebSphere Portal InfoCenter  for instructions on configuring these portlets to use a proxy. For more information about the Content Access Service, refer to the sections entitled "Accessing remote systems" and "Portal service configuration" in the WebSphere Portal InfoCenter . If you are unsure if your company uses a proxy server, contact your network administrator.</p> <p>Choose to not use a proxy (option A) or use a proxy (option B).</p>	
<ul style="list-style-type: none"> • Option A: Do not use proxy: Enables you to use the WebSphere Portal server without proxy. 	
<ul style="list-style-type: none"> • Option B: Use proxy: Configures WebSphere Portal server for portlets that use Content Access Service. 	
<ul style="list-style-type: none"> • Hostname: Specifies the hostname of your proxy server. A hostname is a fully qualified Domain Name System (DNS) domain name that can be resolved to one or more IP addresses via the DNS domain name service. It represents a logical host and must be resolvable to at least one fully qualified Internet address in numeric (dotted quad) form or often to a list of hosts with different IP addresses. The wizard attempts to verify that the hostname and port are valid by attempting to connect to the specified proxy server. If the wizard is unable to connect, a warning is issued. Before the portal server can use the proxy, you must make sure the proxy server is running. Example: <i>myiseries.proxy.domain</i> 	
<ul style="list-style-type: none"> • Port: Specifies the TCP/IP port number on which the proxy server is listening. Example: 78 	

Step 10: Deploy Default Portlets

WebSphere Portal includes a variety of portlets that you can deploy and use in your portal pages. The Administrative portlets and Themes and Skins are deployed by default. Specify the portlets that you want to deploy to your WebSphere Portal server.

<ul style="list-style-type: none"> • Administrative portlets: These portlets allow you the ability to easily and efficiently administer the content and resources of your portal server, deploy new portlets, and control who has access to the portal server.
<ul style="list-style-type: none"> • Themes and skins: These portlets easily customize the look and feel of your portal.
<ul style="list-style-type: none"> • Business portlets: These portlets provide connectivity and integration to allow access to enterprise data, external newsfeeds, weather, newsgroups, and other applications.
<ul style="list-style-type: none"> • Portal Document Manager (PDM) portlets: These portlets streamline complicated document management processes by providing a centralized location for documents and built-in methods for tracking changes and comments from members of the work team. They provide protection for the documents by controlling who has read or write access.
<ul style="list-style-type: none"> • Lotus Collaborative portlets: These portlets provide access to a variety of Lotus Notes-based applications, such as e-mail, calender, to-do list, discussion, team room, contacts, and notebooks, and other Lotus applications such as knowledge management, instant messaging, quickplaces, and document management.
<ul style="list-style-type: none"> • iSeries Access portlets: These portlets allow you to access information on your iSeries servers through a Web browser. This option is disabled if you do not have the following 5722-XH2 PTFs applied: <ul style="list-style-type: none"> – SI11914 (5722XH2) iSeries Access portlets include the following: <ul style="list-style-type: none"> – 5250 portlet: <ul style="list-style-type: none"> - Run commands and access full-screen 5250 character-based applications. – IFrame portlet: <ul style="list-style-type: none"> - Access any of the iSeries Access for Web servlets using the IFrame portlet. – Integrated file system browsing portlets: <ul style="list-style-type: none"> - Browse the iSeries integrated file system. - View, edit, upload and download files. – Printers, printer output, and output queues portlets: <ul style="list-style-type: none"> - View printer status, start and stop the writer job associated with a printer. - Hold, release, print, delete and view printer output files. - Move printer output files to another output queue or printer. - Hold and release output queues. – Database tables and SQL portlets: <ul style="list-style-type: none"> - View database tables, add and update records. - View query results, customize format of results. - Run SQL statements dynamically. – Command portlets: <ul style="list-style-type: none"> - Run CL commands.

- **Lotus Collaboration Center:** These portlets provide users immediate access to a searchable directory of people that is integrated with their workplaces and their e-meetings within the collaborative portal. Users can find people in the directory, see their online status, and interact with them using instant messaging and other actions provided by people links. In addition to search features, the People Finder provides views of each person's directory record and his or her place in the organizational context. This option is disabled if you do not have WebSphere Portal - Express Plus, or if you have WebSphere Portal - Express Plus, but chose not to install the Lotus Collaboration Center.

Note: After the portlets have been deployed, they may require user configuration before they are functional. Refer to the WebSphere Portal InfoCenter  for documentation regarding specific portlets.

If you select to deploy **Lotus Collaborative portlets** or **Lotus Collaboration Center**, continue to Step 11: Configure Lotus Collaborative Components . Otherwise, to continue, skip to **Step 12: Secure server using LDAP**.

Step 11: Configure Lotus Collaborative Components

Lotus Collaborative Components provide the building blocks for integrating the functionality of Lotus Domino, Lotus Sametime®, and Lotus QuickPlace® into portals and portlets. To use Lotus Collaborative Components, you must use the Lotus companion products. In addition, you can configure Lotus Collaborative Components to use Domino Directory as the LDAP server. Specify the Lotus Collaborative Components that you want to deploy to your WebSphere Portal server. The hostname and port number are required. The default port number for Lotus Sametime is 1533. The default port number for Lotus QuickPlace is 80.

Note: The port number is used by the Domino HTTP server to serve the Lotus companion products. Contact your Domino administrator to determine what port number should be used for each Lotus companion product.


Lotus Collaborative Components are Java APIs that provide the building blocks for integrating the functionality of Domino, Lotus Sametime, Lotus QuickPlace, and Lotus Discovery Server™ into portals and portlets. Using Lotus Collaborative Components, application developers can design and implement user interface extensions in portals and portlets that extend the functionality of Lotus Software collaboration products. The primary goal of Lotus Collaborative Components is to provide the data for portlets' user interface and to enable developers to execute actions on installed Lotus products. Lotus Collaborative Components include no platform-specific code, hide the configuration details of Lotus Software products installed in the enterprise, and (except for the PeopleService tags) are user interface neutral. Application developers can add collaborative functionality to a portlet without having to know the details of server configuration and with total control of user interface design and implementation. These benefits make Lotus Collaborative Components effective for implementing mobile applications.

Step 12: Secure server using LDAP


Determine if your WebSphere Portal server requires security. If you want to control who has access to the WebSphere Portal server environment, configure LDAP. The wizard enables global security for the application server. Global security enables the LDAP server to provide the WebSphere Portal server with user information. The LDAP server must be active and you must have the administrator user's Distinguished Name (DN) and password. The wizard updates the LDAP server with the required application server and WebSphere Portal server information such as the WebSphere Portal administrator user and group. If you plan to use LDAP with WebSphere Portal server, it is advised that you use the wizard now to configure LDAP for you. The wizard configures Single Signon (SSO) for the application server. SSO allows the user to enter one name and password and access to more than one application.

Choose to not secure your WebSphere Portal server with LDAP (option A) or secure your WebSphere Portal server with LDAP (option B).

- **Option A: No, do not secure this server:** Enables you to use WebSphere Portal server without LDAP (not secured).
- If you select this option, skip to Step 16: Portal Administrative Group and Administrative User to continue.
Note: If you plan on using LDAP with WebSphere Portal server, it is advised you use the wizard now to configure LDAP for you.
- **Option B: Yes, secure this server using LDAP:** Enables you to use WebSphere Portal server with LDAP (secured server).

<ul style="list-style-type: none"> • LDAP server host name: Specifies the LDAP hostname. The LDAP server can be located either on this iSeries or on a remote system. Specify the hostname for the local or remote LDAP server that you want to use. By default, the wizard retrieves the hostname for the LDAP server for this iSeries. Edit the hostname if you wish to use another LDAP server within your enterprise. The LDAP server specified here must be running. The wizard attempts to connect to the LDAP server. If a connection is not made, an error is issued and you cannot proceed. Example: <i>MYSERVER.COM</i> Note: Only certain types of LDAP servers are supported by WebSphere Portal. Refer to the WebSphere Portal InfoCenter  for more information about LDAP servers. 	
<ul style="list-style-type: none"> • LDAP port: Specifies the TCP/IP port number on which the LDAP server is listening. The wizard initializes the port to the port being used by the IBM Directory Server if it is configured or to the default port number. The default value is port number 389 for TCP/IP connections and 636 for SSL connections to the LDAP server. Edit the port number if your LDAP server does not listen on port number 389 by default. Check your LDAP server configuration to determine the correct port number. 	
<p>Step 13: Specify LDAP user and password</p> <p>The wizard connects to your LDAP server and attempts to gather the proper configuration parameters for you. The configuration for LDAP cannot continue without a valid LDAP administrator user and password for this connection. The wizard verifies that the LDAP administrator Distinguished Name (DN) and password values correspond to a valid LDAP entry. If the entry does not exist or an incorrect password is entered, an error is issued and you cannot proceed.</p>	
<ul style="list-style-type: none"> • LDAP administrator DN: Specifies the administration DN for the LDAP server. The DN uniquely identifies an entry in the LDAP server. The wizard tries to retrieve the administrator user DN from the LDAP server. If the administrator DN is found, the value is entered into this field. If the value is not found, this field is left blank and you must enter the correct value in this field. Edit the value if you want to change the administrator DN. Example: <i>cn=administrator</i> 	
<ul style="list-style-type: none"> • LDAP administrator password: Specifies the password for the administrator DN. 	
<p>Step 14: LDAP Configuration Parameters</p> <p>The WebSphere Portal server uses LDAP to store user information for authentication purposes. Specify where the administrator user and group reside in your LDAP directory. This is the same location that the repository of users authorized to access the Portal Server resides. This can be existing list of users that have been defined in your enterprise. The wizard connects to your LDAP server and attempts to gather the proper configuration parameters for you.</p>	
<ul style="list-style-type: none"> • Information describing user entries 	

<ul style="list-style-type: none"> • Parent DN: Specifies the user parent DN for the LDAP server. This is the container where the Portal administrator user (<i>wpsadmin</i>) and where all new Portal users will reside. If a <i>wpsadmin</i> entry already exists in this LDAP server, it is recommended that you specify the user parent DN (container) where that <i>wpsadmin</i> entry is located. If no <i>wpsadmin</i> Portal administrator exists in this LDAP server, the wizard creates a new <i>wpsadmin</i> entry in the user parent DN container that you specify. All new Portal users are created in this user parent DN container. If your LDAP server has a user parent DN container that is already populated with LDAP users that are to have access to Portal and if no <i>wpsadmin</i> Portal administrator exists in this LDAP server, specify this populated user parent DN container. The users already in this user parent DN container will now have access to Portal. All new Portal users that are registered via Portal will be created in this user parent DN container as well. The wizard searches for the default user parent DN <i>cn=users,DC=myserver,DC=area,DC=co,DC=com</i>. If it is found, the value is entered into this field. If the value is not found, select the user parent DN with Browse. Edit the value if you want to change the user parent DN. Example: <i>cn=users,DC=myserver,DC=area,DC=co,DC=com</i> Note: For IBM Directory Server, when using the default value (<i>cn=users,DC=myserver,DC=area,DC=co,DC=com</i>), the wizard creates the <i>cn=users</i> container for you if it does not exist. If you require a special user parent DN, create the special user parent DN in your LDAP server before proceeding. Refer to the WebSphere Portal InfoCenter for more information on LDAP servers and their recommended configurations. The Browse button can be used to select a location within your directory where user information is stored. This could be a container that already holds user information. The directory browser is provided as an aid for specifying the location in the directory where user information will be stored. Users are typically stored under suffixes that begin with <i>c=</i>, <i>o=</i>, <i>ou=</i>, or <i>dc=</i>. For this reason, the directory browser only displays suffixes that begin with these values. 	
<ul style="list-style-type: none"> • Object class: Select an object class that inherits, directly or indirectly from the person class. A list of possible classes is retrieved by the wizard. 	
<ul style="list-style-type: none"> • Naming attribute: Select a naming attribute. The possible values are retrieved by the wizard based on the value of the object class. 	
<ul style="list-style-type: none"> • Information describing the administrative group entry 	

<ul style="list-style-type: none"> • Parent DN: Specifies the group parent DN for the LDAP server. This is the container in which the Portal administrator group (<i>wpsadmins</i>) must exist. If a <i>wpsadmins</i> group entry already exists in this LDAP server, it is recommended that you specify the group parent DN (container) where that <i>wpsadmins</i> entry is located. If no <i>wpsadmins</i> Portal administrator group exists in this LDAP server, the wizard creates a new <i>wpsadmins</i> entry in the group parent DN container that you specify. The wizard makes the <i>wpsadmin</i> administrator a member of that group. Unless you are using an IBM Directory Server and have chosen <i>cn=groups,DC=myserver,DC=area,DC=co,DC=com</i>, the group parent DN container must exist before continuing (it is created for you in an IBM Directory Server). If you want to create a special group parent DN, you need to create that container first before continuing. The wizard searches for the default group parent DN <i>cn=groups,DC=myserver,DC=area,DC=co,DC=com</i>. If it is found, the value is entered into this field. If the value is not found, select the group parent DN with Browse. Edit the value if you want to change the group parent DN. Example: <i>cn=groups,DC=myserver,DC=area,DC=co,DC=com</i> Note: For IBM Directory Server, when using the default value (<i>cn=groups,DC=myserver,DC=area,DC=co,DC=com</i>), the wizard creates the <i>cn=groupscontainer</i> for you if it does not already exist. Also, for Domino LDAP servers, the use of group Parent DNs is not required. Therefore, if you configure WebSphere Portal to use a Domino LDAP server, the wizard provides a special value, *ROOT, that you may use for your group Parent DN. This value indicates that the <i>wpsadmins</i> Portal administrator group is to be located in the root of the Domino LDAP server. This is the only LDAP server type for which this additional support is provided. (You are not required to use the *ROOT special value when configuring a Domino server. You can specify to have the <i>wpsadmins</i> Portal administrator group located in a group parent DN container.) Refer to the WebSphere Portal InfoCenter  for more information about what common group parent DNs are used based on your LDAP server type. Some LDAP servers do not use group Parent DNs. 	
<ul style="list-style-type: none"> • Object class: Select an object class that contains the member, uniqueMember, or memberURL class. A list of possible classes is retrieved by the wizard. 	
<ul style="list-style-type: none"> • Naming attribute: Select a naming attribute. The possible values are retrieved by the wizard based on the value of the object class. 	
<ul style="list-style-type: none"> • Member attribute: The member attribute is retrieved by the wizard based on the value of the object class. 	

<p>Step 15: LDAP Administrative Group and Administrative User</p> <p>The WebSphere Portal server requires an administrative group and user entry in the LDAP directory. The wizard defaults to wpsadmins and wpsadmin, respectively. The wizard creates the wpsadmins and wpsadmin LDAP entries if they do not exist, and then adds wpsadmin to the wpsadmins group in your LDAP directory. The values shown are used to create the LDAP Distinguished Name (DN) for these entries. Provide a password that can be used to access the administrative functions of the installed applications in the future.</p> <p>Note: wpsadmin is the WebSphere Portal administrator, the WebSphere Application Server administrator, and the LDAP authentication user.</p>	
<ul style="list-style-type: none"> • Administrative group name: The administrative group name is retrieved by the wizard and cannot be edited. 	
<ul style="list-style-type: none"> • Administrator name: The administrator name is retrieved by the wizard and cannot be edited. 	
<ul style="list-style-type: none"> • Password: Specifies a new password for the portal administrator ID. Important: Keep the password for future reference. If the administrator user is already in the selected LDAP directory, you need to provide the password for that user to proceed with the installation of the WebSphere Portal server. 	
<ul style="list-style-type: none"> • Confirm password: Specifies the password for the portal administrator ID a second time for confirmation. Note: Some LDAP servers, such as Microsoft Active Directory, require a Secure Socket Layer (SSL) connection to create users. If you are securing your WebSphere Portal server with such an LDAP server and wpsadmin or wpsadmins do not already exist, create a wpsadmin Portal administrator entry in the directory and assign the necessary authority, because the Create Portal wizard is not able to create the entry. Next, create a wpsadmins Portal administrative group entry in the directory and add wpsadm into this group. After you have successfully created the two entries, enter the password twice. 	
<p>Step 16: Portal Administrative Group and Administrative User</p> <p>If you determine your WebSphere Portal does not require security (for example, you are not securing this server with an LDAP server via the wizard), a WebSphere Portal server administrative group, administrator user, and password are required when using the WebSphere Portal administration tools.</p>	
<ul style="list-style-type: none"> • Administrative group name: The administrative group name is retrieved by the wizard and cannot be edited. 	
<ul style="list-style-type: none"> • Administrator name: The administrator name is retrieved by the wizard and cannot be edited. 	
<ul style="list-style-type: none"> • Password: Specify the password for the administrative group and administrative name. 	
<ul style="list-style-type: none"> • Confirm Password: Specify the new user ID password a second time for confirmation. 	
<p>Step 17: Web Server Single Signon (SSO) Configuration Parameters</p> <p>The wizard configures single signon (SSO) to allow a user to access multiple Web servers without the need to re-authenticate. Specify the SSO domain:</p>	
<ul style="list-style-type: none"> • Option A: Limit SSO domain to this Web server's hostname: Does not enable your WebSphere Portal to acquire information from other Web servers without additional authentication. To continue, skip to Step 21: Configure Look-Aside Database. 	

<ul style="list-style-type: none"> • Option B: Include other Web servers in your SSO environment: Enables your WebSphere Portal to acquire information from other Web servers without additional authentication. 	
<ul style="list-style-type: none"> • SSO domain name: Specifies the DNS hostname that is shared by all Web servers participating in your SSO environment. The SSO domain name is case sensitive. Example: <i>domain.com</i> 	
<p>Step 18: Configure Lightweight Third Party Authentication (LTPA) for Web Server Single Signon (SSO) Environment</p> <p>Lightweight Third Party Authentication (LTPA) is the mechanism used to implement Web Server SSO. LTPA is a set of tokens or cookies which provide a means to share authentication and access control information between Web servers. The information in these tokens uniquely identifies the user. The application server creates the keys. The keys must be exported from this application server to other Web servers in your SSO environment. Create a password to encrypt and decrypt the LTPA keys.</p>	
<ul style="list-style-type: none"> • LTPA password: Specifies a new password for LTPA. Important: Keep the password for future reference. 	
<ul style="list-style-type: none"> • Confirm password: Specifies the password for LTPA a second time for confirmation. 	
<p>Step 19: Configure Identity Token SSO for Web to iSeries Access</p> <p>The wizard configures Identity Token support to enable single signon (SSO) capabilities. This allows authenticated WebSphere users to access iSeries server data and resources through an Enterprise Identity Mapping (EIM) mapping operation.</p> <p>Choose to not configure identity tokens (option A) or to configure identity tokens (option B).</p>	
<ul style="list-style-type: none"> • Option A: Do not configure Identity Tokens: If you select this option, the wizard does not install the EIM Identity Token Connector resource adapter plug-in into your WebSphere Application Server instance. Web applications that normally support using the WebSphere authenticated user and identity tokens to access iSeries resources are not able to provide this type of single signon capability. Select this option if you do not want web applications to provide this function. Identity Token support can be added at a later time by using the WebSphere Application Server administrative console to install the resource adapter. • If you select this option, skip to Step 21: Configure Look-Aside Database. 	
<ul style="list-style-type: none"> • Option B: Configure Identity Tokens: If you select this option, the wizard installs the EIM Identity Token Connector resource adapter plug-in into your WebSphere Application Server instance. An EIM domain must be configured and running on an LDAP server. In addition, a user registry definition in EIM must be created that represents the source WebSphere registry being used for authentication, such as an LDAP user registry. For more information on configuring EIM and Identity Tokens, see Enterprise Identity Mapping (EIM) in the iSeries Information Center. 	

<ul style="list-style-type: none"> • LDAP server host name:Specifies the fully qualified TCP/IP host name of the LDAP server hosting the EIM domain controller. This value was set by the administrator when the EIM Configuration Wizard in iSeries Navigator was run to configure an iSeries server for EIM. Example: myhost.mycompany.com. 	
<ul style="list-style-type: none"> • LDAP port:Specifies the TCP/IP port number on which the LDAP server hosting the EIM domain controller is listening. By default, an LDAP directory server using an unsecured connection listens on port 389. This value was set by the administrator when the EIM Configuration Wizard in iSeries Navigator was run to configure an iSeries server for EIM. 	
<ul style="list-style-type: none"> • LDAP administrator DN:The (Distinguished Name) DN of the user who is capable of performing searches on the LDAP directory. If the LDAP directory contents are restricted to certain users, you need to specify the DN of an authorized user. This value was set by the administrator when the EIM Configuration Wizard in iSeries Navigator was run to configure an iSeries server for EIM. Example: cn=administrator. 	
<ul style="list-style-type: none"> • LDAP administrator password:The password corresponding to the LDAP administrator DN field. This field is case sensitive. This value was set by the administrator when the EIM Configuration Wizard in iSeries Navigator was run to configure an iSeries server for EIM. 	
<p>Step 20: Configure Identity Token EIM Domain information</p> <p>The wizard has signed into specified LDAP server which is an EIM Domain Controller. Select the correct EIM domain and source registry name that contains the repository of identity information.</p>	
<ul style="list-style-type: none"> • EIM domain name:The name of the EIM domain to use for Identity Token EIM mapping operations. The wizard retrieves the names of the EIM domains currently defined on the EIM domain controller. This value was set by the administrator when the EIM Configuration Wizard in iSeries Navigator was run to configure an iSeries server for EIM. Example: EIM. 	
<ul style="list-style-type: none"> • Source registry name:The name of the EIM user registry that contains WebSphere user identity source associations. The wizard retrieves the names of the EIM user registries currently defined for the selected EIM domain. This value was set by the Add a new system registry request in the iSeries Navigator. For more information, see Add a system registry definition in the iSeries Information Center. Example: WebSphereLdapUserRegistry. 	

Step 21: Configure Look-Aside Database

The WebSphere Portal server is configured to use an LDAP directory to store the user account information for security. The information is referred to as user registry information and includes user and password information. The LDAP directory can also be extended to contain additional user preference information such as e-mail addresses and telephone numbers. The additional information is referred to as user repository information. If your LDAP directory can be extended to contain both the user registry and user repository information, a look-aside database is not required. However, if the LDAP directory cannot store all the profile information, the WebSphere Portal database can be used as a look-aside database for storing the additional profile information. If you think that you need the look-aside database, configure it now. If you choose to use the look-aside database at a later time, you need to reconfigure your LDAP security configuration.

Choose to not configure the look-aside database (option A) or configure the look-aside database now (option B).

• **Option A: No, do not configure the look-aside database:**
Disables the look-aside database.

• **Option B: Yes, configure the look-aside database:** Enables the look-aside database to extend your LDAP directory.

Step 22: Summary

The wizard summary page details information you specified for the WebSphere Application Server, the HTTP Server (powered by Apache), the WebSphere Portal server, the WebSphere Portal server security, the deployed portlets, and the WebSphere Portal server properties. It is recommend that you click the **Printable Summary** button for a complete summary of the options and values you provided and that you print this page for future reference.

Step 23: Creation and Configuration

The wizard now configures and deploys your WebSphere Portal server. Note creation and configuration of your WebSphere Portal server and its components may take some time to complete. You can monitor the progress from the WebSphere Portal server introduction page that displays after the **Finish** button has been clicked.

To access your new WebSphere Portal on the Web, enter `http://[iSeries]:[port]/wps/portal` in a Web browser, where *iSeries* is the name of your iSeries and *port* is the port number for your HTTP Server (powered by Apache).

Virtual host tasks

This topic provides step-by-step tasks for virtual hosts.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Related information

“Virtual hosts on HTTP Server” on page 65

This topic provides information about virtual host types.

Set up virtual hosts on HTTP Server (powered by Apache)

This topic provides information about how to set up virtual hosts on your HTTP Server with the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 65 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 (powered by Apache) 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 (powered by Apache) 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 IBM Web Administration for iSeries interface provides the IP addresses used by your iSeries 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 (powered by Apache) 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 powered by Apache) 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	
<code>%%</code>	inserts a %
<code>%p</code>	inserts the port number of the virtual host
<code>%N.M</code>	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
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`.

Programming

This topic provides information on CGI, directives, APIs, and other programming topics.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Note: As of V5R3, the Web Programming Guide is now in HTML format. The contents of the guide are found under this topic.

Application Programming Interface

This topic provides information on application programming interfaces (APIs).

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Application programming interfaces (APIs) for HTTP Server

This topic provides information about HTTP Server APIs for CGI applications, configuration APIs, server instance APIs, group APIs, and triggered cache manager APIs.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

HTTP Server supports the APIs listed below in C++, Java, 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.

CGI application APIs: To use these APIs in a CGI application, you must bind the CGI program to `*SRVPGM QZHBTCGI` in library `QHTTPSVR`. 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 224
- “Put Environment Variable (QtmhPutEnv) API” on page 226
- “Read from Stdin (QtmhRdStin) API” on page 227
- “Write to Stdout (QtmhWrStout) API” on page 228
- “Convert to DB (QtmhCvtDB) API” on page 222
- “Parse QUERY_STRING Environment Variable or Post stdin data (QzhhCgiParse) API” on page 231
- “Produce Full HTTP Response (QzhhCgiUtils) API” on page 237
- “Send or Save CGI Stateful Data (QzhhCgiSendState) API” on page 236
- “Receive CGI Stateful Data (QzhhCgiRecvState) API” on page 235

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 427
- “Change Config Object Value (QzuiChangeConfigObject) API” on page 429
- “Close Apache Config File (QzuiCloseConfig) API” on page 432
- “Find Config Object (QzuiFindConfigObject) API” on page 435
- “Open Apache Config File (QzuiOpenConfig) API” on page 440
- “Remove Config Object (QzuiRemoveConfigObject) API” on page 442

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 430
- “Create Apache Server Instance (QzuiCreateInstance) API” on page 433
- “Get Apache Server Instance Data (QzuiGetInstanceData) API” on page 437
- “Get Server Instance Names (QzuiGetInstanceNames) API” on page 438
- “Get Instance Type (QzuiGetInstanceType) API” on page 440

Group file APIs: The group file APIs are in *SRVPGM QZHBCONF in library QHTTSPVR. ILE C programs must include header file QHTTSPVR/H(QZHBCONF).

Note: Group file APIs are independent of HTTP Server type.

- “Create a new Group File (QzhbCreateGroupList) API” on page 239
- “Read a Group File into Memory (QzhbOpenGroupList) API” on page 246
- “Free Group File from Memory (QzhbCloseGroupList) API” on page 238
- “Retrieve the next Group in the Group List (QzhbGetNextGroup) API” on page 243
- “Locate a named group in a Group List (QzhbFindGroupInList) API” on page 240
- “Retrieve the Name of a Group (QzhbGetGroupName) API” on page 242
- “Add a new Group to the end of a Group List (QzhbAddGroupToList) API” on page 229
- “Remove a Group from a Group List (QzhbRemoveGroupFromList) API” on page 248
- “Retrieve the next User in the Group (QzhbGetNextUser) API” on page 244
- “Locate a User in a Group (QzhbFindUserInGroup) API” on page 241
- “Retrieve the Name of a User (QzhbGetUserString) API” on page 245
- “Add a new user to the end of a Group (QzhbAddUserToGroup) API” on page 230
- “Remove a User or Element from a Group (QzhbRemoveUserFromGroup) API” on page 248

TCM server operation APIs: The trigger cache manager server operation APIs may be used to start and stop trigger cache manager servers. The APIs are callable service implemented as an ILE entry point within the QZHTINOPS *SVRPGM in the QTCM *LIB. The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for these APIs.

- “Start Triggered Cache Manager Server (QzhtStrTCMServer) API” on page 426
- “End Triggered Cache Manager Server (QzhtEndTCMServer) API” on page 367

TCM basic configuration APIs: The trigger cache manager basic configuration APIs are a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB. The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for these APIs.

- “Create Triggered Cache Manager Server (QzhtCrtTCMServer) API” on page 362
- “Get Triggered Cache Manager Server Status (QzhtGetTCMServerStatus) API” on page 368
- “Delete Triggered Cache Manager Server Status (QzhtDltTCMServer) API” on page 365
- “Retrieve Triggered Cache Manager Basic Configuration (QzhtRtvTCMBasicConfig) API” on page 390
- “Change Triggered Cache Manager Basic Configuration (QzhtChgTCMBasicConfig) API” on page 305

TCM trigger handler configuration APIs: The trigger cache manager trigger handler configuration APIs are a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB. The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for these APIs.

- “Add Triggered Cache Manager Trigger Handler Description (QzhtAddTCMTriggerHandlerDesc) API” on page 288
- “Retrieve Triggered Cache Manager Trigger Handler Description (QzhtRtvTCMTriggerHandlerDesc) API” on page 419
- “Change Triggered Cache Manager Trigger Handler Description (QzhtChgTCMTriggerHandlerDesc) API” on page 353
- “Remove Triggered Cache Manager Trigger Handler Description (QzhtRmvTCMTriggerHandlerDesc) API” on page 383

TCM data source configuration APIs: The trigger cache manager data source configuration APIs are a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB. The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for these APIs.

- “Add Triggered Cache Manager Data Source Description (QzhtAddTCMDataSourceDesc) API” on page 270
- “Retrieve Triggered Cache Manager Data Source Description (QzhtRtvTCMDataSourceDesc) API” on page 402

- “Change Triggered Cache Manager Data Source Description (QzhtChgTCMDataSourceDesc) API” on page 329
- “Remove Triggered Cache Manager Data Source Description (QzhtRmvTCMDataSourceDesc) API” on page 374

TCM cache target configuration APIs: The trigger cache manager cache target configuration APIs are a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB. The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for these APIs.

- “Add Triggered Cache Manager Cache Target Description (QzhtAddTCMCacheTargetDesc) API” on page 256
- “Retrieve Triggered Cache Manager Cache Target Description (QzhtRtvTCMCacheTargetDesc) API” on page 393
- “Change Triggered Cache Manager Cache Target Description (QzhtChgTCMCacheTargetDesc) API” on page 308
- “Remove Triggered Cache Manager Cache Target Description (QzhtRmvTCMCacheTargetDesc) API” on page 372

TCM acknowledgement target configuration APIs: The triggered cache manager acknowledgement target configuration APIs are a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB. The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for these APIs.

- “Add Triggered Cache Manager Acknowledgment Target Description (QzhtAddTCMAckTargetDesc) API” on page 249
- “Retrieve Triggered Cache Manager Acknowledgment Target Description (QzhtRtvTCMAckTargetDesc) API” on page 385
- “Change Triggered Cache Manager Acknowledgment Target Description (QzhtChgTCMAckTargetDesc) API” on page 297
- “Remove Triggered Cache Manager Acknowledgment Target Description (QzhtRmvTCMAckTargetDesc) API” on page 371

TCM host configuration APIs: The triggered cache manager host configuration APIs are a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB. The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for these APIs.

- “Add Triggered Cache Manager Host Description (QzhtAddTCMHostDesc) API” on page 277
- “Retrieve Triggered Cache Manager Host Description (QzhtRtvTCMHostDesc) API” on page 407
- “Change Triggered Cache Manager Host Description (QzhtChgTCMHostDesc) API” on page 338
- “Remove Triggered Cache Manager Host Description (QzhtRmvTCMHostDesc) API” on page 376

TCM object dependency graph configuration APIs: The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for these API.

- “Add Triggered Cache Manager Object Dependency Graph Description (QzhtAddTCMODGDesc) API” on page 279
- “Retrieve Triggered Cache Manager Object Dependency Graph Description (QzhtRtvTCMODGDesc) API” on page 410
- “Change Triggered Cache Manager Object Dependency Graph Description (QzhtChgTCMODGDesc) API” on page 341
- “Remove Triggered Cache Manager Object Dependency Graph Description (QzhtRmvTCMODGDesc) API” on page 378

TCM rule set configuration APIs: The triggered cache manager rule set configuration APIs are a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB. The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for these APIs.

- “Add Triggered Cache Manager Rule Set (QzhtAddTCMRuleSet) API” on page 286
- “Retrieve Triggered Cache Manager Rule Set (QzhtRtvTCMRuleSet) API” on page 416
- “Change Triggered Cache Manager Rule Set (QzhtChgTCMRuleSet) API” on page 350
- “Remove Triggered Cache Manager Rule Set (QzhtRmvTCMRuleSet) API” on page 381

TCM publishing rule configuration APIs: The triggered cache manager publishing rule configuration APIs are a callable service implemented as an ILE entry point within QZHTINCONF *SRVPGM in QTCM *LIB. The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for these APIs.

- “Add Triggered Cache Manager Publishing Rule (QzhtAddTCMPublishingRule) API” on page 282
- “Retrieve Triggered Cache Manager Publishing Rule (QzhtRtvTCMPublishingRule) API” on page 413
- “Change Triggered Cache Manager Publishing Rule (QzhtChgTCMPublishingRule) API” on page 345
- “Remove Triggered Cache Manager Publishing Rule (QzhtRmvTCMPublishingRule) API” on page 380

Convert to DB (QtmhCvtDB) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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(*)

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. 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 iSeries 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:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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(*)

The *QtmhGetEnv* API allows you to get the value set by the server for a particular HTTP environment variable.

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:BINAR(4)

The input variable containing the length of the space provided to receive the environment variable value.

Length of response

OUTPUT:BINAR(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:BINAR(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 iSeries 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.

Put Environment Variable (QtmhPutEnv) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 Environment string	Input	Char(*)
2 Length of environment string	Input	Binary(4)
3 Error Code	I/O	Char(*)

The QtmhPutEnv API allows you to set or create a job-level environment variable. This is useful for communication between programs running in the same job, such as your program and the Net.Data language environment DTW_SYSTEM.

Required parameter group:

Environment string

INPUT:CHAR(*)

The input string of the form: $\Delta envVar = value \Delta$. Where $\Delta envVar \Delta$ is the name of the new or existing environment variable, and $\Delta value \Delta$ 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 $\Delta envVar = value \Delta$ 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 iSeries 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.

Read from Stdin (QtmhRdStin) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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(*)

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.

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:BINAR(4)

The input variable containing the number of bytes that are to be read from `stdin`.

Length or response available

OUTPUT:BINAR(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 iSeries 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:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 Data variable	Input	Char(*)
2 Length of data variable	Input	Binary(4)
3 Error Code	I/O	Char(*)

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:***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 iSeries 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.

Add a new Group to the end of a Group List (QzhbAddGroupToList) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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

Use the *QzhbAddGroupToList* API to add a new group to an in-memory group list.

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:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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

Use the *QzhbAddUserToGroup* API to add a new user to an in-memory group.

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.

Parse QUERY_STRING Environment Variable or Post stdin data (QzhbCgiParse) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 Command string	Input	Char(*)
2 Output format	Input	Char(8)
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(*)

Use *QzhbCgiParse* API 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.

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(20)

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[*gain*] *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 CGII0200 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 Δ FORM_ Δ .

-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 iSeries Information Center.

CGII0200 Format:

Offset Decimal	Offset Hexadecimal	Type	Field
0	0	Binary(4)	Bytes returned
4	4	Binary(4)	Bytes available
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: For further information on errors, the joblog for the CGI job may contain CPF9898 messages (with all English text) describing the error in more detail.

Receive CGI Stateful Data (QzhbCgiRecvState) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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(*)

Use the *QzhbCgiRecvState* API with HA CGI programs to receive the CGI stateful data. 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*.

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 this 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 API Error Reporting in the iSeries 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 QzhbCgiRecvState() after it had already received the entire state.

HTP4006 E

QzhbCgiRecvState() was called when there was no state.

Send or Save CGI Stateful Data (QzhbCgiSendState) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 CGI's state string	Input	Void
2 Length of the string	Input	Binary(4)
3 Error Code	I/O	Char(*)

Use the *QzhbCgiSendState* API with HA CGI programs to send or save the CGI stateful data. 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:VOID

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 API Error Reporting in the iSeries 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:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1	Command string	Input	Char(*)
2	Error code	I/O	Char(*)

Use *QzhhCgiUtils* API 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:

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/plain.

-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 iSeries 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.

Free Group File from Memory (QzhbCloseGroupList) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1	grplist	Input	Binary(4)
2	write	Input	Binary(4)
3	errcode	I/O	Char(*)

Threadsafe: Yes

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.

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:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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

Use the *QzhbCreateGroupList* API to create a new empty group file, and return a handle to that empty in-memory version of the file. 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 246.

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:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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

Use the *QzhbFindGroupInList* API to search an in-memory group list for a named group.

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:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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

Use the *QzhbFindUserInGroup* API to search an in-memory group for a specific user.

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:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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

Use the *QzhbGetGroupName* API to retrieve the name of a group using the group handle.

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:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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

Use the *QzhbGetNextGroup* API to retrieve the next group from an in-memory group list.

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:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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

Use the *QzhbGetNextUser* API to retrieve the next user from a group.

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:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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

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.

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:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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
Threadsafe: Yes

I/O

Char(*)

Use the *QzHbOpenGroupList* API to read in an existing group file, and return a handle to an in-memory version of the file. See “Free Group File from Memory (*QzHbCloseGroupList*) API” on page 238 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:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1	grplist	Input	Binary(4)
2	grp	Input	Binary(4)
3	errcode	I/O	Char(*)

Threadsafe: Yes

Use the *QzhbRemoveGroupFromList* API to remove a named group, and all the users in that group, from an in-memory group list.

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:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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

Use the *QzhbRemoveUserFromGroup* API to remove a user from an in-memory group.

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 Triggered Cache Manager Acknowledgment Target Description (QzhtAddTCMAckTargetDesc) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1	request variable	Input	Char(*)
2	length or request variable	Input	Binary(4)
3	request variable format	Input	Char(8)
4	error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

Use the QzhtAddTCMAckTargetDesc API to add new acknowledgment target descriptions to the configurations of triggered cache manager servers. New acknowledgment target descriptions are referenced subsequently, by name, from trigger handler descriptions associated with the same server. New acknowledgment target descriptions are utilized by all trigger handler descriptions, referencing them the next time the servers are restarted. The API is a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

request variable

INPUT: CHAR(*)

The variable used to pass the information used to add a new acknowledgment target description. See Acknowledgment target description formats for more information.

length of request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for request variable.

request variable format

INPUT: CHAR(8)

The format name of the Request variable data. The following values must be used:

- ATDP0100: Basic information format for an acknowledgment target description.
- ATDP0200: Detailed information format for an *HTTP1 type acknowledgment target description.
- ATDP0210: Detailed information format for an *HTTP2 type acknowledgment target description.

error code

I/O: CHAR(*)

The structure in which to return error information.

ATDP0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Char(32)	Acknowledgement target description name
64	40	Char(10)	Acknowledgement target type
74	4A	Char(2)	Reserved
76	4C	Binary(4)	Default property
80	50	Binary(4)	Number of threads
84	54	Binary(4)	Initial state

ATDP0100 format field descriptions:

Note: Acknowledgment target descriptions are added using default values for all unspecified values according to the type specified by Acknowledgment target type). See other acknowledgment target description formats for details regarding these default values.

Acknowledgement target description name

The name used by the new acknowledgment target description (left justified and padded with blanks if necessary).

Note: Acknowledgment target description names must be unique for each triggered cache manager server. They are referenced, by name, from trigger handler descriptions associated with the same server.

Acknowledgement target type

The type of acknowledgment target description that is added (left justified and padded with blanks if necessary). The value must be one of the special values described below.

Special values and their meanings are as follows:

*HTTP1

QZHT_HTTP_TYPE1: An *HTTP1 type is added.

*HTTP2

QZHT_HTTP_TYPE2: An *HTTP2 type is added.

Default property

Specifies if the new description is the default acknowledgment target description for the triggered cache manager server specified in Server name. The value must equal one of the special values described below. The default value is 0 (QZHT_NO).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The description is the default cache target description for this server.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The description is not the default cache target description.
-1	QZHT_DEFAULT: The default value is used.

Note: Multiple default acknowledgment target descriptions are allowed. Trigger handler descriptions, added or changed using the *DEFAULT special value, reference all acknowledgment target descriptions designated as default at the time the trigger handler descriptions are added or changed.

Initial state

Specifies the state that the triggered cache manager server request processor, for this acknowledgment target, is in at server startup. The value must equal one of the special values described below. The default value is 1 (QZHT_ENABLED).

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The request processor is enabled at server startup.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The request processor is disabled at server startup.
-1	QZHT_DEFAULT: The default value is used.

Note: The state of a acknowledgment target request processor can be changed while the triggered cache manager server is active by using the *-chack* command in a trigger message.

Number of threads

The number of concurrent threads the triggered cache manager server spawns when processing requests sent to this acknowledgment target. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below. The default value is 5.

Special values and their meanings are as follows:

-1	QZHT_DEFAULT: The default value is used.
----	--

Server name

The name used to identify the triggered cache manager server for which the new description is associated (left justified and padded with blanks if necessary).

ATDP0200 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(256)	Everything from ATDP0100 format
88	58	Binary(4)	HTTP IP interface
344	158	Binary(4)	HTTP TCP port
348	15C	Binary(4)	Offset to HTTP URI root
352	160	Binary(4)	Length of HTTP URI root
356	164	Binary(4)	HTTP keepalive
360	168	Binary(4)	Timeout
		Char(*)	HTTP URI root

ATDP0200 format field descriptions:

HTTP IP interface

The IP host name or address of the system hosting an HTTP server that accepts completion messages (left justified and padded with blanks if necessary). The value must be a host name (for example, server.mycompany.com), dotted address (for example, 192.168.3.57), or one of the special values described below. If a host name is specified, it must use proper naming conventions as defined by RFC 1034, Domain Names - Concepts and Facilities. If a dotted address is specified, it must use proper IP version 4 address conventions as defined by RFC 791, Internet Protocol. The default value is 127.0.0.1, the local system loopback interface.

Special values and their meanings are as follows:

*DEFAULT

QZHT_DEFAULT_CHAR: The default value is used.

HTTP keepalive

Specifies if connections to HTTP Server are kept open for reuse after data is transferred. The value must equal one of the special values described below. The default value is 0 (QZHT_DISABLED).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The connection is kept open after data transfer.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The connection is closed after data transfer.
-1	QZHT_DEFAULT: The default value is used.

Note: HTTP Server must support keepalive for this option to work properly.

HTTP TCP port

The TCP port number upon which HTTP Server listens for incoming requests. The value must be greater than 0 and less than 65536, or equal to one of the special values described below. The default value is 80.

Special values and their meanings are as follows:

-1	QZHT_DEFAULT: The default value is used.
----	--

Note: The TCP port number is used in combination with the IP host name or address in HTTP IP interface to establish communication with HTTP Server acknowledgment target.

HTTP URI root

The path of HTTP Server URI that is the root of this acknowledgment target (left justified and padded with blanks if necessary). The value must be a path acceptable to HTTP Server. The default path is / .

Note: All completion message requests sent to the acknowledgment target are prepended with this path. If the value is null, Offset to HTTP URI root must equal 0 (QZHT_NONE), or -1 (QZHT_DEFAULT). See Offset to HTTP URI root for more details.

Length of HTTP URI root

The length of the information for the HTTP URI root entry.

Note: If Offset to HTTP URI root equals 0 (QZHT_NONE), or -1 (QZHT_DEFAULT), this value must equal 0.

Offset to HTTP URI root

The offset from the beginning of the request variable to the HTTP URI root data, in bytes. The value must be greater than 0, or equal to one of the special values defined below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
-1	QZHT_DEFAULT: The default value is used for HTTP URI root.

Timeout

The number of seconds the triggered cache manager server waits before canceling a read operation on sockets and ending transactions, with the HTTP host, in error. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below. The default value is 0 (QZHT_DISABLED).

Special values and their meanings are as follows:

0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The time out operation is disabled. The triggered cache manager server should never cancel a read operation.
-1	QZHT_DEFAULT: The default value is used.

ATDP0210 format:

Offset		Type	Field
Dec	Hex		
0	0		Everything from ATDP0100 format
88	58	Char(32)	HTTP host
120	78	Binary(4)	HTTP TCP port
124	7C	Binary(4)	Offset to HTTP URI root
128	80	Binary(4)	Length of HTTP URI root
132	84	Binary(4)	HTTP keepalive
136	88	Binary(4)	Timeout
		Char(*)	HTTP URI root

ATDP0210 format field descriptions:

HTTP host

The name of a host description associated with the triggered cache manager server that is referenced by the new acknowledgment target description and used later, at server startup, to obtain information about the system hosting an HTTP server accepting completion messages. The value must be a host description name, or one of the special values described below (left justified and padded with blanks if necessary). The default value is to reference the description currently designated as the default host description for the triggered cache manager server.

Special values and their meanings are as follows:

*DEFAULT

QZHT_DEFAULT_CHAR: The default value is referenced.

Note: See HTTP TCP port for more information. An escape message is sent if the referenced description does not currently exist, or if *DEFAULT is specified and there is currently no default host description for the server.

HTTP keepalive

Specifies if connections to HTTP Server are kept open for reuse after completion messages are sent. The value must equal one of the special values described below. The default value is 0 (QZHT_DISABLED).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The connection is kept open after messages are sent.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The connection is closed after messages are sent.
-1	QZHT_DEFAULT: The default value is used.

Note: HTTP Server must support keepalive for this option to work properly.

HTTP TCP port

The TCP port number upon which HTTP Server listens for incoming completion messages. The value must be greater than 0 and less than 65536, or equal to one of the special values described below. The default value is 80.

Special values and their meanings are as follows:

-1	QZHT_DEFAULT: The default value is used.
----	--

Note: The TCP port number is used in combination with information obtained at server startup from the host description specified in HTTP host to establish communication with HTTP Server acknowledgment target.

HTTP URI root

The path of HTTP Server URI that is the root of this acknowledgment target (left justified and padded with blanks if necessary). The value must be a path acceptable to HTTP Server. The default path is / .

Note: All requests to send completion messages to this acknowledgment target are prepended with this path. If the value is null, Offset to HTTP URI root must equal 0 (QZHT_NONE), or -1 (QZHT_DEFAULT). See Offset to HTTP URI root for more details.

Length of HTTP URI root

The length of the information for the HTTP URI root entry.

Note: If Offset to HTTP URI root equals 0 (QZHT_NONE), or -1 (QZHT_DEFAULT), this value must equal 0.

Offset to HTTP URI root

The offset from the beginning of the request variable to the HTTP URI root data, in bytes. The value must be greater than 0, or equal to one of the special values defined below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
-1	QZHT_DEFAULT: The default path is used for HTTP URI root.

Timeout

The number of seconds the triggered cache manager server waits before canceling a read operation on sockets and ending transactions, with the HTTP host, in error. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below. The default value is 0 (QZHT_DISABLED).

Special values and their meanings are as follows:

0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The time out operation is disabled. The triggered cache manager server should never cancel a read operation.
-1	QZHT_DEFAULT: The default value is used.

Error messages:

TCM7001 E

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7031 E

Request variable format is not valid.

TCM7033 E

Length of request variable is not valid for the specified format.

TCM703E E

Error code parameter is not valid.

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 E

A configuration file for triggered cache manager server &1 was not found.

TCM7222 E

A default &1 is not designated for triggered cache manager server &2.

TCM7293 E

A &1 using the name &2 already exists for triggered cache manager server &3.

TCM72C0 E

Triggered cache manager &1 name is not valid.

TCM72C1 E

Triggered cache manager &1 type is not valid.

TCM72C2 E

Triggered cache manager description type &1 cannot be specified when using data format &2.

TCM72C3 E

Value passed to offset &1 is not valid.

TCM74C0 E

Triggered cache manager server name is not valid.

Add Triggered Cache Manager Cache Target Description (QzhtAddTCMCacheTargetDesc) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 request variable	Input	Char(*)
2 length of request variable	Input	Binary(4)
3 request variable format	Input	Char(8)
4 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

Use the QzhtAddTCMCacheTargetDesc API to add new cache target descriptions to the configurations of triggered cache manager servers. New cache target descriptions are referenced subsequently, by name, from trigger handler descriptions associated with the same server. New cache target descriptions are utilized by all trigger handler descriptions, referencing then the next time the servers are restarted. The API is a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

request variable

INPUT: CHAR(*)

The variable used to pass the information used to add a new cache target description. See Cache target description formats for more information.

length of request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for request variable.

request variable format

INPUT: CHAR(8)

The format name of the request variable data. The following values must be used:

- CTDP0100: Basic information format for a cache target description.
- CTDP0200: Detailed information format for an *IFS type cache target description.
- CTDP0300: Detailed information format for a *HTTP1 type cache target description.
- CTDP0310: Detailed information format for a *HTTP2 type cache target description.
- CTDP0500: Detailed information format for a *ECCP1 type cache target description.
- CTDP0510: Detailed information format for a *ECCP2 type cache target description.

error code

I/O: CHAR(*)

The structure in which to return error information.

CTDP0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Char(32)	Cache target description name
64	40	Char(10)	Cache target type
74	4A	Char(2)	Reserved
76	4C	Binary(4)	Default property
80	50	Binary(4)	Number of threads
84	54	Binary(4)	Initial state

CTDP0100 format field descriptions:

Note: Cache target descriptions are added using default values for all unspecified values according to the type specified by Cache target type). See other cache target description formats for details regarding these default values. This format cannot be used to add cache target descriptions of type *ECCP1 or *ECCP2.

Cache target description name

The name used by the new cache target description (left justified and padded with blanks if necessary).

Note: Cache target description names must be unique for each triggered cache manager server. They are referenced, by name, from trigger handler descriptions associated with the same server.

Cache target type

The type of cache target description that is added (left justified and padded with blanks if necessary). The value must be one of the special values described below.

Special values and their meanings are as follows:

***IFS** QZHT_IFS_TYPE: An *IFS type is added.

***HTTP1**

QZHT_HTTP_TYPE1: An *HTTP1 type is added.

***HTTP2**

QZHT_HTTP_TYPE2: An *HTTP2 type is added.

Note: Cache target description types of *ECCP1 and *ECCP2 must be added using the CTDP0500 and CTDP0510 formats, respectively.

Default property

Specifies if the new description will become the default cache target description for the triggered cache manager server specified in Server name. The value must equal one of the special values described below. The default value is 0 (QZHT_NO).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The description becomes the default cache target description for this server.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The description is not the default cache target description.
-1	QZHT_DEFAULT: The default value is used.

Note: Multiple default cache target descriptions are allowed. Trigger handler descriptions added or changed using the *DEFAULT special value reference all cache target descriptions designated as default at the time the trigger handler descriptions are added or changed.

Initial state

Specifies the state that the triggered cache manager server request processor, for this cache target, is in at server startup. The value must equal one of the special values described below. The default value is 1 (QZHT_ENABLED).

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The request processor is enabled at server startup.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The request processor is disabled at server startup.
-1	QZHT_DEFAULT: The default value is used.

Note: The state of a cache target request processor can be changed while the triggered cache manager server is active by using the *-chsink* command in a trigger message.

Number of threads

The number of concurrent threads the triggered cache manager server spawns when processing requests sent to this cache target. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below. The default value is 5.

Special values and their meanings are as follows:

-1	QZHT_DEFAULT: The default value is used.
----	--

Server name

The name used to identify the triggered cache manager server for which the new description is associated (left justified and padded with blanks if necessary).

CTDP0200 format:

Offset		Type	Field
Dec	Hex		
0	0		Everything from CTDP0100 format
88	58	Binary(4)	Offset to local directory root
92	5C	Binary(4)	Length of local directory root
		Char(*)	Local directory root

CTDP0200 format field descriptions:

Length of local directory root

The length of the information for the Local directory root entry.

Note: If Offset to local directory root equals 0 (QZHT_NONE), or -1 (QZHT_DEFAULT), this value must equal 0.

Local directory root

The path to the local file system directory that is the root of this cache target (left justified and padded with blanks if necessary). The value must be a path acceptable to the root (/) file system of the local iSeries system. If a path is provided, but does not specify an absolute path (does not start with /), the path is prepended with the default path. The default path is /QIBM/UserData/TCM/{Server name} where {Server name} is the name of the triggered cache manager server as defined by Server name.

Note: All requests for files from the cache target must have this path prepended to the file name, even if an absolute file path is specified. If the value is null, Offset to local directory root must equal 0 (QZHT_NONE) or -1 (QZHT_DEFAULT). See Offset to local directory root for more details.

Offset to local directory root

The offset from the beginning of the request variable to the Local directory root data, in bytes. The value must be greater than 0, or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
-1	QZHT_DEFAULT: The default path is used for Local directory root.

CTDP0300 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(256)	Everything from CTDP0100 format
88	58	Binary(4)	HTTP IP interface
344	158	Binary(4)	HTTP TCP port

348	15C	Binary(4)	Offset to HTTP URI root
352	160	Binary(4)	Length of HTTP URI root
356	164	Binary(4)	HTTP keepalive
360	168	Binary(4)	Timeout
		Char(*)	HTTP URI root

CTDP0300 format field descriptions:

HTTP IP interface

The IP host name or address of the system hosting an HTTP server cache target (left justified and padded with blanks if necessary). The value must be a host name (for example, server.mycompany.com), dotted address (for example, 192.168.3.57), or one of the special values described below. If a host name is specified, it must use proper naming conventions as defined by RFC 1034, Domain Names - Concepts and Facilities. If a dotted address is specified, it must use proper IP version 4 address conventions as defined by RFC 791, Internet Protocol. The default value is 127.0.0.1, the local system loopback interface.

Special values and their meanings are as follows:

*DEFAULT

QZHT_DEFAULT_CHAR: The default value is used.

HTTP keepalive

Specifies if connections to HTTP Server are kept open for reuse after data is transferred. The value must equal one of the special values described below. The default value is 0 (QZHT_DISABLED).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The connection is kept open after data transfer.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The connection is closed after data transfer.
-1	QZHT_DEFAULT: The default value is used.

Note: HTTP Server must support keepalive for this option to work properly.

HTTP TCP port

The TCP port number upon which HTTP Server listens for incoming requests. The value must be greater than 0 and less than 65536, or equal to one of the special values described below. The default value is 80.

Special values and their meanings are as follows:

-1	QZHT_DEFAULT: The default value is used.
----	--

Note: The TCP port number is used in combination with the IP host name or address in HTTP IP interface to establish communication with HTTP Server cache target.

HTTP URI root

The path of HTTP Server URI that is the root of this cache target (left justified and padded with blanks if necessary). The value must be a path acceptable to HTTP Server. The default path is / .

Note: All file requests from this cache target have this path inserted into the URI, even if an absolute path for the file is specified. If the value is null, Offset to HTTP URI root must equal 0 (QZHT_NONE), or -1 (QZHT_DEFAULT). See Offset to HTTP URI root for more details.

Length of HTTP URI root

The length of the information for the HTTP URI root entry.

Note: If Offset to HTTP URI root equals 0 (QZHT_NONE), or -1 (QZHT_DEFAULT), this value must equal 0.

Offset to HTTP URI root

The offset from the beginning of the request variable to the HTTP URI root data, in bytes. The value must be greater than 0, or equal to one of the special values defined below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
-1	QZHT_DEFAULT: The default value is used for HTTP URI root.

Timeout

The number of seconds the triggered cache manager server waits before canceling a read operation on sockets and ending transactions, with the HTTP host, in error. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below. The default value is 0 (QZHT_DISABLED).

Special values and their meanings are as follows:

0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The time out operation is disabled. The triggered cache manager server should never cancel a read operation.
-1	QZHT_DEFAULT: The default value is used.

CTDP0310 format:

Offset		Type	Field
Dec	Hex		
0	0		Everything from CTDP0100 format
88	58	Char(32)	HTTP host
120	78	Binary(4)	HTTP TCP port
124	7C	Binary(4)	Offset to HTTP URI root
128	80	Binary(4)	Length of HTTP URI
132	84	Binary(4)	HTTP keepalive
136	88	Binary(4)	Timeout
		Char(*)	HTTP URI root

CTDP0310 format field descriptions:

HTTP host

The name of a host description associated with the triggered cache manager server that is referenced by the new cache target description and used later, at server startup, to obtain information about the system hosting an HTTP server cache target. The value must be a host description name, or one of the special values described below (left justified and padded with

blanks if necessary). The default value is to reference the description currently designated as the default host description for the triggered cache manager server.

Special values and their meanings are as follows:

***DEFAULT**

QZHT_DEFAULT_CHAR: The default value is referenced.

Note: See HTTP TCP port for more information. An escape message is sent if the referenced description does not currently exist, or if *DEFAULT is specified and there is currently no default host description for the server.

HTTP keepalive

Specifies if connections to HTTP Server are kept open for reuse after data is transferred. The value must equal one of the special values described below. The default value is 0 (QZHT_DISABLED).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The connection is kept open after data transfer.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The connection is closed after data transfer.
-1	QZHT_DEFAULT: The default value is used.

Note: HTTP Server must support keepalive for this option to work properly.

HTTP TCP port

The TCP port number upon which HTTP Server listens for incoming requests. The value must be greater than 0 and less than 65536, or equal to one of the special values described below. The default value is 80.

Special values and their meanings are as follows:

-1	QZHT_DEFAULT: The default value is used.
----	--

Note: The TCP port number is used in combination with information obtained at server startup from the host description specified in HTTP host to establish communication with HTTP Server cache target.

HTTP URI root

The path of HTTP Server URI that is the root of this cache target (left justified and padded with blanks if necessary). The value must be a path acceptable to HTTP Server. The default path is / .

Note: All requests to send files to this cache target have this path inserted into the URI, even if an absolute file path is specified. If the value is null, Offset to HTTP URI root must equal 0 (QZHT_NONE), or -1 (QZHT_DEFAULT). See Offset to HTTP URI root for more details.

Length of HTTP URI root

The length of the information for the HTTP URI root entry.

Note: If Offset to HTTP URI root equals 0 (QZHT_NONE), or -1 (QZHT_DEFAULT), this value must equal 0.

Offset to HTTP URI root

The offset from the beginning of the request variable to the HTTP URI root data, in bytes. The value must be greater than 0, or equal to one of the special values defined below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
-1	QZHT_DEFAULT: The default path is used for HTTP URI root.

Timeout

The number of seconds the triggered cache manager server waits before canceling a read operation on sockets and ending transactions, with the HTTP host, in error. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below. The default value is 0 (QZHT_DISABLED).

Special values and their meanings are as follows:

0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The time out operation is disabled. The triggered cache manager server should never cancel a read operation.
-1	QZHT_DEFAULT: The default value is used.

CTDP0500 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Char(32)	Cache target description name
64	40	Char(10)	Cache target type
74	4A	Char(2)	Reserved
76	4C	Binary(4)	Default property
80	50	Binary(4)	Number of threads
84	54	Binary(4)	Initial state
88	58	Char(256)	ECCP IP interface
344	158	Binary(4)	ECCP TCP port
348	15C	Char(32)	HTTP cluster IP interface
604	260	Binary(4)	HTTP cluster TCP port
608	264	Binary(4)	Offset to HTTP cluster URI root
612	268	Binary(4)	Length of HTTP cluster URI root
6161	26C	Binary(4)	ECCP keepalive
		Char(*)	HTTP cluster URI root

CTDP0500 format field descriptions:

Cache target description name

The name used by the new cache target description (left justified and padded with blanks if necessary).

Note: Cache target description names must be unique for each triggered cache manager server. They are referenced, by name, from trigger handler descriptions associated with the same server.

Cache target type

The added cache target descriptions type (left justified and padded with blanks if necessary). The value must be one of the special values described below.

Special values and their meanings are as follows:

*ECCP1

QZHT_ECCP_TYPE1: An *ECCP1 type is added.

Default property

Specifies if the new description is a default cache target description for the triggered cache manager server specified in Server name. The value must equal one of the special values described below. The default value is 0 (QZHT_NO).

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The description is a default cache target description for this server.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The description is not a default cache target description.
-1	QZHT_DEFAULT: The default value is specified.

Note: Multiple default cache target descriptions are allowed. Trigger handler descriptions added or changed, using the *DEFAULT special value, reference all cache target descriptions designated as default at the time they are added or changed.

ECCP IP interface

The IP host name or address of the backend IP interface to the network router hosting a web server cluster cache target (left justified and padded with blanks if necessary). The value must be a host name (for example, server.mycompany.com) or dotted address (for example, 192.168.3.57). If a host name is specified, it must use proper naming conventions as defined by RFC 1034, Domain Names - Concepts and Facilities. If a dotted address is specified, it must use proper IP version 4 address conventions as defined by RFC 791, Internet Protocol. There is no default value for this entry.

Note: An exception occurs if an IP host name or address is not specified. See ECCP TCP port for more information.

ECCP keepalive

Specifies if the connection to the network router is kept open for reuse after data is transferred. The value must equal one of the special values described below. The default value is 0 (QZHT_DISABLED).

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The connection is kept open after data transfer.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The connection is closed after data transfer.
-1	QZHT_DEFAULT: The default value is used.

Note: The network router must support keepalive for this option to function properly.

ECCP TCP port

The TCP port number upon which the network router listens for incoming requests. The value must be greater than 0 and less than 65536. There is no default value for this entry.

Note: The TCP port number is used in combination with the IP host name or address, in ECCP IP interface, to establish communications with the network router cache target.

HTTP cluster IP interface

The IP host name or address of the web server cluster from which the network router is hosting a cache (left justified and padded with blanks if necessary). The value must be a host name (for example, server.mycompany.com), dotted address (for example, 192.168.3.57), or one of the special values described below. If a host name is specified, it must use proper naming conventions as defined by RFC 1034, Domain Names - Concepts and Facilities. If a dotted address is specified, it must use proper IP version 4 address conventions as defined by RFC 791, Internet Protocol. There is no default value for this entry.

Note: An IP host name or address must be specified. See HTTP cluster TCP port for more information.

HTTP cluster TCP port

The web server cluster URI path that is used to define the root for this cache target (left justified and padded with blanks if necessary). The value must be a path acceptable to the web server cluster (from which the network router is hosting a cache). The default path is /.

Note: All file requests sent to this cache target have this path inserted into the URI, even if an absolute file path is specified. If the value is null, Offset to HTTP cluster URI root must equal 0 (QZHT_NONE), or -1 (QZHT_DEFAULT). See Offset to HTTP URI root for more details.

HTTP cluster URI root

The web server cluster URI path that is used to define the root of this cache target (left justified and padded with blanks if necessary). The value must be a path acceptable to the web server cluster from which the network router is hosting a cache. The default path is /.

Note: All file requests sent to this cache target have this path inserted into the URI, even if an absolute file path is specified. If the value is null, Offset to HTTP cluster URI root must equal 0 (QZHT_NONE), or -1 (QZHT_DEFAULT). See Offset to HTTP URI root for more details.

Initial state

Specifies the state that the triggered cache manager server request processor, for this cache target, is in at server startup. The value must equal one of the special values described below. The default value is 1 (QZHT_ENABLED).

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The request processor is enabled at server startup.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The request processor is disabled at server startup.
-1	QZHT_DEFAULT: The default value is used.

Note: The state of a cache target request processor can be changed while the triggered cache manager server is active by using the *-chsink* command in a trigger message.

Length of HTTP cluster URI root

The length of the information for the HTTP cluster URI root entry.

Note: If Offset to HTTP cluster URI root equals 0 (QZHT_NONE), or -1 (QZHT_DEFAULT), this value must equal 0.

Number of threads

The number of concurrent threads the triggered cache manager server spawns when processing requests sent to this cache target. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below. The default value is 5.

Special values and their meanings are as follows:

-1	QZHT_DEFAULT: The default value is used.
----	--

Offset to HTTP URI root

The offset from the beginning of the request variable to the HTTP cluster URI root data, in bytes. The value must be greater than 0, or equal to one of the special values defined below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
-1	QZHT_DEFAULT: The default path is used for HTTP cluster URI root.

Server name

The name used to identify the triggered cache manager server for which the new description is associated (left justified and padded with blanks if necessary).

CTDP0510 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Char(32)	Cache target description name
64	40	Char(10)	Cache target type
74	4A	Char(2)	Reserved
76	4C	Binary(4)	Default property
80	50	Binary(4)	Number of threads
84	54	Binary(4)	Initial state
88	58	Char(32)	ECCP host
120	78	Binary(4)	ECCP TCP port
124	7C	Char(256)	HTTP cluster IP interface
380	17C	Binary(4)	HTTP cluster TCP port
384	180	Binary(4)	Offset to HTTP cluster URI root
388	184	Binary(4)	Length of HTTP cluster URI root
392	188	Binary(4)	ECCP keepalive
		Char(*)	HTTP cluster URI root

CTDP0510 format field descriptions:

Cache target description name

The name used by the new cache target description (left justified and padded with blanks if necessary).

Note: Cache target description names must be unique for each triggered cache manager server. They are referenced, by name, from trigger handler descriptions associated with the same server.

Cache target type

The added cache target description type (left justified and padded with blanks if necessary). The value must be one of the special values described below.

Special values and their meanings are as follows:

*ECCP1

QZHT_ECCP_TYPE2: An *ECCP2 type is added.

Default property

Specifies if the new description is a default cache target description for the triggered cache manager server specified in Server name. The value must equal one of the special values described below. The default value is 0 (QZHT_NO).

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The description is a default cache target description for this server.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The description is not a default cache target description.
-1	QZHT_DEFAULT: The default value is specified.

Note: Multiple default cache target descriptions are allowed. Trigger handler descriptions added or changed, using the *DEFAULT special value, reference all cache target descriptions designated as default at the time they are added or changed.

ECCP host

The name of a host description associated with the triggered cache manager server referenced by the new cache target description and used later, at server startup, to obtain information about the network router hosting a web server cluster cache target. The value must be a host description name, or one of the special values described below (left justified and padded with blanks if necessary). The default setting references the description currently designated as the default host description for the triggered cache manager server.

Special values and their meanings are as follows:

*DEFAULT

QZHT_DEFAULT_CHAR: The default is referenced.

Note: See ECCP TCP port for more information. An escape message is sent if the referenced description does not currently exist, or if *DEFAULT is specified and there is currently no default host description for the server.

ECCP keepalive

Specifies if the connection to the network router is kept open for reuse after data is transferred. The value must equal one of the special values described below. The default value is 0 (QZHT_DISABLED).

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The connection is kept open after data transfer.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The connection is closed after data transfer.
-1	QZHT_DEFAULT: The default value is used.

Note: The network router must support keepalive for this option to function properly.

ECCP TCP port

The TCP port number upon which the network router listens for incoming requests. The value must be greater than 0 and less than 65536.

Note: There is no default value for this entry. A TCP port number must be specified. The TCP port number is used in combination with the IP host name or address, obtained at server startup, from the host description (specified in ECCP host) to establish communication with the network router cache target.

HTTP cluster IP interface

The IP host name or address of the web server cluster from which the network router is hosting a cache (left justified and padded with blanks if necessary). The value must be a host name (for example, server.mycompany.com), dotted address (for example, 192.168.3.57), or one of the special values described below. If a host name is specified, it must use proper naming conventions as defined by RFC 1034, Domain Names - Concepts and Facilities. If a dotted address is specified, it must use proper IP version 4 address conventions as defined by RFC 791, Internet Protocol. There is no default value for this entry.

Note: There is no default value for this entry. An IP host name or address must be specified. See HTTP cluster TCP port for more information.

HTTP cluster TCP port

The number of the TCP port associated with the IP host name or address described by HTTP cluster IP interface. The value must be greater than 0 and less than 65536.

Note: There is no default value for this entry. A TCP port number must be specified. The network router using the address described by ECCP host may host multiple web server caches. The TCP port number specified here is used in combination with the IP host name or address in HTTP cluster IP interface to identify a particular web server cache when communicating with the network router.

HTTP cluster URI root

The web server cluster URI path that is used to define the root of this cache target (left justified and padded with blanks if necessary). The value must be a path acceptable to the web server cluster from which the network router is hosting a cache. The default path is /.

Note: All file requests sent to this cache target have this path inserted into the URI, even if an absolute file path is specified. If the value is null, Offset to HTTP cluster URI root must equal 0 (QZHT_NONE), or -1 (QZHT_DEFAULT). See Offset to HTTP URI root for more details.

Initial state

Specifies the state that the triggered cache manager server request processor, for this cache target, is in at server startup. The value must equal one of the special values described below. The default value is 1 (QZHT_ENABLED).

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The request processor is enabled at server startup.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The request processor is disabled at server startup.
-1	QZHT_DEFAULT: The default value is used.

Note: The state of a cache target request processor can be changed while the triggered cache manager server is active by using the *-chsink* command in a trigger message.

Length of HTTP cluster URI root

The length of the information for the HTTP cluster URI root entry.

Note:

Number of threads

The number of concurrent threads the triggered cache manager server spawns when processing

requests sent to this cache target. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below. The default value is 5.

Special values and their meanings are as follows:

-1	QZHT_DEFAULT: The default value is used.
----	--

Offset to HTTP URI root

The offset from the beginning of the request variable to the HTTP cluster URI root data, in bytes. The value must be greater than 0, or equal to one of the special values defined below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
-1	QZHT_DEFAULT: The default value is used.

Server name

The name used to identify the triggered cache manager server for which the new description is associated (left justified and padded with blanks if necessary).

Error messages:

TCM7001 E

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7031 E

Request variable format is not valid.

TCM7033 E

Length of request variable is not valid for the specified format.

TCM703E E

Error code parameter is not valid

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 E

A configuration file for triggered cache manager server &1 was not found.

TCM7222 E

A default &1 is not designated for triggered cache manager server &2.

TCM7293 E

A &1 using the name &2 already exists for triggered cache manager server &3.

TCM72C0 E

Triggered cache manager &1 name is not valid.

TCM72C1 E

Triggered cache manager &1 type is not valid.

TCM72C2 E

Triggered cache manager description type &1 cannot be specified when using data format &2.

TCM72C3 E

Value passed to offset &1 is not valid.

TCM74C0 E

Triggered cache manager server name is not valid.

Add Triggered Cache Manager Data Source Description (QzhtAddTCMDataSourceDesc) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 request variable	Input	Char(*)
2 length of request variable	Input	Binary(4)
3 request variable format	Input	Char(8)
4 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

Use the QzhtAddTCMDataSourceDesc API to add new data source descriptions to the configurations of triggered cache manager servers. New data source descriptions are referenced subsequently, by name, from trigger handler descriptions associated with the same server. New data source descriptions are utilized by all trigger handler descriptions, referencing then the next time the servers are restarted. The API is a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

request variable

INPUT: CHAR(*)

The variable used to pass the information used to add a new data source description.

length of request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for request variable.

request variable format

INPUT: CHAR(8)

The format name of the Request variable data. The following values must be used:

- DSDP0100: Basic information format for a data source description.
- DSDP0200: Detailed information format for an *IFS type data source description.
- DSDP0300: Detailed information format for a *HTTP1 type data source description.
- DSDP0310: Detailed information format for a *HTTP2 type data source description.

error code

I/O: CHAR(*)

The structure in which to return error information.

DSDP0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Char(32)	Data source description name
64	40	Char(10)	Data source type
74	4A	Char(2)	Reserved
76	4C	Binary(4)	Default property
80	50	Binary(4)	Number of threads

DSDP0100 format field descriptions:

Note: Data source descriptions are added using default values for all unspecified values according to the type specified by Data source type). See other data source description formats for details regarding these default values.

Data source description name

The name used by the new data source description (left justified and padded with blanks if necessary).

Note: Data source description names must be unique for each triggered cache manager server. They are referenced, by name, from trigger handler descriptions associated with the same server.

Data source type

The type of data source description that is added (left justified and padded with blanks if necessary). The value must be one of the special values described below.

Special values and their meanings are as follows:

***IFS** QZHT_IFS_TYPE: An *IFS type is added.

***HTTP1**
QZHT_HTTP_TYPE1: An *HTTP1 type is added.

***HTTP2**
QZHT_HTTP_TYPE2: An *HTTP2 type is added.

Default property

Specifies if the new description will become the default data source description for the triggered cache manager server specified in Server name. The value must equal one of the special values described below. If the value equals 1 (QZHT_YES), the default property on the current default data source description is set to 0 (QZHT_NO). This description takes over the default role. The default value is 0 (QZHT_NO).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The description becomes the default data source description for this server.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The description is not the default data source description.
-1	QZHT_DEFAULT: The default value is used.

Note: Trigger handler descriptions added or changed using the *DEFAULT special value are referenced from this data source description designated as default at the time the trigger handler descriptions are added or changed.

Number of threads

The number of concurrent threads the triggered cache manager server spawns when processing triggers sent to this trigger handler. The value must be greater than 0 and less than 2^{31} (or 2.147x109), or equal to one of the special values described below. The default value is 5.

Special values and their meanings are as follows:

-1	QZHT_DEFAULT: The default value is used.
----	--

Server name

The name used to identify the triggered cache manager server for which the description is associated (left justified and padded with blanks if necessary).

DSDP0200 format:

Offset		Type	Field
Dec	Hex		
0	0		Everything from DSDP0100 format
84	54	Binary(4)	Offset to local directory root
88	58	Binary(4)	Length of local directory root
		Char(*)	Local root directory

DSDP0200 format field descriptions:

Length of local directory root

The length of the information for the Local directory root entry.

Note: If Offset to local directory root equals 0 (QZHT_NONE) or -1 (QZHT_DEFAULT), this value must equal 0.

Local directory root

The path to the local file system directory that is the root of this data source (left justified and padded with blanks if necessary). The value must be a path acceptable to the root (/) file system of the local iSeries system. If a path is provided, but does not specify an absolute path (does not start with /), the path is prepended with the default path. The default path is /QIBM/UserData/TCM/{Server name} where {Server name} is the name of the triggered cache manager server as defined by Server name.

Note: All requests for files from the data source must have this path prepended to the file name, even if an absolute path is specified for the file. If the value is null, Offset to local directory root must equal 0 (QZHT_NONE) or -1 (QZHT_DEFAULT).

Offset to local directory root

The offset from the beginning of the request variable to the Local directory root data, in bytes. The value must be greater than 0, or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
---	---------------------------------------

-1	QZHT_DEFAULT: The default path is used.
----	---

DSDP0300 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(256)	Everything from DSDP0100 format
88	58	Binary(4)	HTTP IP interface
340	154	Binary(4)	HTTP TCP port
344	158	Binary(4)	Offset to HTTP URI root
348	15C	Binary(4)	Length of HTTP URI root
352	160	Binary(4)	HTTP keepalive
356	164	Binary(4)	Timeout
		Char(*)	HTTP URI root

DSDP0300 format field descriptions:

HTTP IP interface

The IP host name or address of the system hosting an HTTP server data source (left justified and padded with blanks if necessary). The value must be a host name (for example, server.mycompany.com), dotted address (for example, 192.168.3.57), or one of the special values described below. If a host name is specified, it must use proper naming conventions as defined by RFC 1034, Domain Names - Concepts and Facilities. If a dotted address is specified, it must use proper IP version 4 address conventions as defined by RFC 791, Internet Protocol. The default value is 127.0.0.1, the local system loopback interface.

Special values and their meanings are as follows:

*DEFAULT

QZHT_DEFAULT_CHAR: The default value is used.

Note: See HTTP TCP port for more information.

HTTP keepalive

Specifies if connections to HTTP Server are kept open for reuse after data is transferred. The value must equal one of the special values described below. The default value is 0 (QZHT_DISABLED).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The connection is kept open after data transfer.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The connection is closed after data transfer.
-1	QZHT_DEFAULT: The default value is used.

Note: HTTP Server must support keepalive for this option to work properly.

HTTP TCP port

The TCP port number upon which HTTP Server listens for incoming requests. The value must be greater than 0 and less than 65536, or equal to one of the special values described below. The default value is 80.

Special values and their meanings are as follows:

-1	QZHT_DEFAULT: The default value is used.
----	--

Note: The TCP port number is used in combination with the IP host name or address in HTTP IP interface to establish communication with HTTP Server data source.

HTTP URI root

The path of HTTP Server URI that is the root of this data source (left justified and padded with blanks if necessary). The value must be a path acceptable to HTTP Server. The default path is / .

Note: All file requests from this data source have this path inserted into the URI, even if an absolute path for the file is specified. If the value is null, Offset to HTTP URI root must equal 0 (QZHT_NONE), or -1 (QZHT_DEFAULT).

Length of HTTP URI root

The length of the information for the HTTP URI root entry.

Note: If Offset to HTTP URI root equals 0 (QZHT_NONE), or -1 (QZHT_DEFAULT), this value must equal 0.

Offset to HTTP URI root

The offset from the beginning of the request variable to the HTTP URI root data, in bytes. The value must be greater than 0, or equal to one of the special values defined below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
-1	QZHT_DEFAULT: The default value is used.

Timeout

The number of seconds the triggered cache manager server waits before canceling a read operation on sockets and ending transactions, with the HTTP host, in error. The value must be greater than 0 and less than 2^{31} (or 2.147x10⁹), or equal to one of the special values described below. The default value is 0 (QZHT_DISABLED).

Special values and their meanings are as follows:

0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The time out operation is disabled. The triggered cache manager server should never cancel a read operation.
-1	QZHT_DEFAULT: The default value is used.

DSDP0310 format:

Offset		Type	Field
Dec	Hex		
0	0		Everything from DSDP0100 format
84	54	Char(32)	HTTP host
116	74	Binary(4)	HTTP TCP port
120	78	Binary(4)	Offset to HTTP URI root
124	7C	Binary(4)	Length of HTTP URI root
128	80	Binary(4)	HTTP keepalive
132	84	Binary(4)	Timeout

Offset		Type	Field
Dec	Hex		
		Char(*)	HTTP URI root

DSDP0310 format field descriptions:

HTTP host

The name of a host description associated with the triggered cache manager server that is referenced by the new data source description and used later, at server startup, to obtain information about the system hosting an HTTP server data source. The value must be a host description name, or one of the special values described below (left justified and padded with blanks if necessary). The default value is to reference the description currently designated as the default host description for the triggered cache manager server.

Special values and their meanings are as follows:

*DEFAULT

QZHT_DEFAULT_CHAR: The default value is referenced.

Note: See HTTP TCP port for more information. An escape message is sent if the referenced description does not currently exist, or if *DEFAULT is specified and there is currently no default host description for the server.

HTTP keepalive

Specifies if connections to HTTP Server are kept open for reuse after data is transferred. The value must equal one of the special values described below. The default value is 0 (QZHT_DISABLED).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The connection is kept open after data transfer.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The connection is closed after data transfer.
-1	QZHT_DEFAULT: The default value is used.

Note: HTTP Server must support keepalive for this option to work properly.

HTTP TCP port

The TCP port number upon which HTTP Server listens for incoming requests. The value must be greater than 0 and less than 65536, or equal to one of the special values described below. The default value is 80.

Special values and their meanings are as follows:

-1	QZHT_DEFAULT: The default value is used.
----	--

Note: The TCP port number is used in combination with information obtained at server startup from the host description specified in HTTP host to establish communication with HTTP Server data source.

HTTP URI root

The path of HTTP Server URI that is the root of this data source (left justified and padded with blanks if necessary). The value must be a path acceptable to HTTP Server. The default path is / .

Note: If the value is null, Offset to HTTP URI root must equal 0 (QZHT_NONE), or -1 (QZHT_DEFAULT).

Length of HTTP URI root

The length of the information for the HTTP URI root entry.

Note: If Offset to HTTP URI root equals 0 (QZHT_NONE), or -1 (QZHT_DEFAULT), this value must equal 0.

Offset to HTTP URI root

The offset from the beginning of the request variable to the HTTP URI root data, in bytes. The value must be greater than 0, or equal to one of the special values defined below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
-1	QZHT_DEFAULT: The default path is used for HTTP URI root.

Timeout

The number of seconds the triggered cache manager server waits before canceling a read operation on sockets and ending transactions, with the HTTP host, in error. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below. The default value is 0 (QZHT_DISABLED).

Special values and their meanings are as follows:

0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The time out operation is disabled. The triggered cache manager server should never cancel a read operation.
-1	QZHT_DEFAULT: The default value is used.

Error messages:

TCM7001 E

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7031 E

Request variable format is not valid.

TCM7033 E

Length of request variable is not valid for the specified format.

TCM703E E

Error code parameter is not valid

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 E

A configuration file for triggered cache manager server &1 was not found.

TCM7222 E

A default &1 is not designated for triggered cache manager server &2.

TCM7293 E

A &1 using the name &2 already exists for triggered cache manager server &3.

TCM72C0 E

Triggered cache manager &1 name is not valid.

TCM72C1 E

Triggered cache manager &1 type is not valid.

TCM72C2 E

Triggered cache manager description type &1 cannot be specified when using data format &2.

TCM72C3 E

Value passed to offset &1 is not valid.

TCM74C0 E

Triggered cache manager server name is not valid.

Add Triggered Cache Manager Host Description (QzhtAddTCMHostDesc) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 request variable	Input	Char(*)
2 length of request variable	Input	Binary(4)
3 request variable format	Input	Char(8)
4 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

Use the QzhtAddTCMHostDesc API to add new host descriptions to the configuration of triggered cache manager servers. New host descriptions are referenced subsequently, by name, from certain types of data source, cache target, and acknowledgment target descriptions associated with the same server. New host descriptions are utilized by all descriptions referencing them the next time the servers are started. The API is a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

*Required parameter group:***request variable**

INPUT: CHAR(*)

The variable used to pass the information used to add a new host description.

length of request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for request variable.

request variable format

INPUT: CHAR(8)

The format name of the Request variable data. The following values must be used:

- HSDP0100: Basic information format for a host description.

error code

I/O: CHAR(*)

The structure in which to return error information.

HSDP0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Char(32)	Host description
64	40	Binary(4)	Default property
68	44	Char(32)	IP interface

Field descriptions:

Default property

Specifies if the new description is the default host description for the triggered cache manager server specified in Server name. The value must equal one of the special values described below. If the value equals 1 (QZHT_YES), the default property for the current default host description is set to 0 (QZHT_NO), and this description takes over the default role. The default value is 0 (QZHT_NO).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The description is the default host description for this server.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The description is not the default host description.
-2	QZHT_DEFAULT: The default value is used.

Note: Data source, cache target, and acknowledgment target descriptions, added or changed using the *DEFAULT special value, reference the host description designated as default at the time they are added or changed.

Host description name

The name used by the new host description (left justified and padded with blanks if necessary).

Note: Host description names must be unique for each triggered cache manager server. They are referenced, by name, from certain types of data source, cache target, and acknowledgment target descriptions associated with the same server.

IP interface

The IP host name or address of the system hosting servers used by the triggered cache manager server (left justified and padded with blanks if necessary). The value must be a host name (for example, server.mycompany.com), dotted address (for example, 192.168.3.57), or one of the special values described below. If a host name is specified, it must use proper naming conventions as defined by RFC 1034, Domain Names - Concepts and Facilities. If a dotted address is specified, it must use proper IP version 4 address conventions as defined by RFC 791, Internet Protocol. The default value is 127.0.0.1, the local system loopback interface.

Special values and their meanings are as follows:

*DEFAULT

QZHT_DEFAULT_CHAR: The default value is used.

Server name

The name used to identify the triggered cache manager server for which the new description is associated (left justified and padded with blanks if necessary).

*Error messages:***TCM7001 E**

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7031 E

Request variable format is not valid.

TCM7033 E

Length of request variable is not valid for the specified format.

TCM703E E

Error code parameter is not valid

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 E

A configuration file for triggered cache manager server &1 was not found.

TCM7293 E

A &1 using the name &2 already exists for triggered cache manager server &3.

TCM72C0 E

Triggered cache manager &1 name is not valid.

TCM72C3 E

Value passed to offset &1 is not valid.

TCM74C0 E

Triggered cache manager server name is not valid.

Add Triggered Cache Manager Object Dependency Graph Description (QzhtAddTCMODGDesc) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 request variable	Input	Char(*)
2 length of request variable	Input	Binary(4)
3 request variable format	Input	Char(8)
4 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

The QzhtAddTCMODGDesc API adds new object dependency graph (ODG) descriptions to the configuration of triggered cache manager servers. New object dependency graph descriptions are

referenced subsequently, but name, from trigger handler descriptions associated with the same server. New object dependency graph descriptions are utilized by all descriptions referencing them the next time the servers are started.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

request variable

INPUT: CHAR(*)

The variable used to pass information used to add a new object dependency graph description.

length or request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for request variable.

request variable format

INPUT: CHAR(8)

The format name of Request variable data. The following values must be used:

- OGDPO100: Basic information format for an object dependency graph description.

error code

I/O: CHAR(*)

The structure in which to return error information.

OGDPO100 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Char(32)	Object dependency graph description name
64	40	Binary(4)	Default property
68	44	Binary(4)	Allow API updates

Field descriptions:

Allow API updates

Specifies whether APIs are allowed to update the object dependency graph described by the new object dependency graph description. The value must be one of the special values described below. The default value is 1 (QZHT_YES).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The object dependency graph is updated using APIs via the triggered cache manager server, as well as from trigger handler process parsing.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The object dependency graph may not be updated using APIs through the triggered cache manager server. Only updates as a result of trigger handler publish parsing are allowed.
-1	QZHT_DEFAULT: The default value is used.

Default property

Specifies whether the new description is to become the default object dependency graph description for the triggered cache manager server specified in Server name. The value must equal one of the special values described below. If the value equals 1 (QZHT_YES), the default property for the current default object dependency graph description is set to 0 (QZHT_NO), and this description takes over the default role. The default value is 0 (QZHT_NO).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The description is to become the default object dependency graph description for this server.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The description is not to become the default object dependency graph description.
-1	QZHT_DEFAULT: The default value is used.

Note: Trigger handler descriptions added or changed using the *DEFAULT special value references the object dependency graph description designated as default at the time the trigger handler descriptions are added or changed.

Object dependency graph description name

The name used by the new object dependency graph description (left justified and padded with blanks if necessary).

Note: Object dependency graph description names must be unique for each triggered cache manager server. They are referenced, by name, from trigger handler descriptions associated with the same server.

Server name

The name used to identify the triggered cache manager server with which the new description is associated (left justified and padded with blanks if necessary).

Error messages:

TCM7001 E

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7031 E

Request variable format is not valid.

TCM7033 E

Length of request variable is not valid for the format specified.

TCM703E E

Error code parameter is not valid.

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 E

A configuration file for triggered cache manager server &1 was not found.

TCM7293 E

A &1 using the name &2 already exists for triggered cache manager server &3.

TCM72C0 E

Triggered cache manager &1 name is not valid.

TCM72C3 E

Value passed at offset &1 is not valid.

TCM74C0 E

Triggered cache manager server name is not valid.

Add Triggered Cache Manager Publishing Rule (QzhtAddTCMPublishingRule) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 request variable	Input	Char(*)
2 length of request variable	Input	Binary(4)
3 request variable format	Input	Char(8)
4 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

The QzhtAddTCMPublishingRule API adds new publishing rules to the configurations of triggered cache manager servers. New publishing rules are referenced subsequently, by name, from rule sets associated with the same server. New publishing rules are utilized by all Rule Sets referencing them the next time the servers are started. The API is a callable service implemented as an ILE entry point within QZHTINCONF *SRVPGM in QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

*Required parameter group:***request variable**

INPUT: CHAR(*)

The variable used to pass information to add a new publishing rule.

length of request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for request variable.

request variable format

INPUT: CHAR(8)

The format name of the request variable data. The following values must be used:

- PRDP0100: Basic information format for a publishing rule.

error code

I/O: CHAR(*)

The structure in which to return error information.

PRDP0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Char(32)	Publishing rule name
64	40	Binary(4)	Offset to file extensions
68	44	Binary(4)	Length of file extensions
72	48	Binary(4)	Read from data source
76	4C	Binary(4)	Send data source version
80	50	Binary(4)	Parse and assemble
84	54	Binary(4)	Send assembled version
88	58	Binary(4)	Offset to new file extension
72	5C	Binary(4)	Length of new file extension
		Char(*)	File extensions
		Char(*)	New file extension

PRDP0100 format field descriptions:

File extension

A list of file extensions used to identify files that are processed according to this new publishing rule. File extensions must be listed as a string of characters, where each extension starts with a period character (.) and is separated by one or more spaces (left justified and padded with blanks if necessary). There is no default value for this entry.

Note: A list of file extensions is required. The file extensions are used by trigger handlers to determine when the publishing rule applies. File names are compared to file extensions starting at the last period in the file name. Therefore, each file extension must start with a period character (.), otherwise a match is not made. For example, .html, .gif, or .event.

Length of file extensions

The length of the information for the File extensions entry.

Length of new file extension

The length of the information for the New file extension entry.

Note: If Offset to new file extension equals 0 (QZHT_NONE), or -1 (QZHT_DEFAULT), this value must equal 0.

New file extension

A file extension used to rename files after they have been assembled and before they are sent to the cache targets (left justified and padded with blanks if necessary). The default value is null, indicating that files are not renamed.

Note: If the value is null, Offset to new file extension must equal 0 (QZHT_NONE) or -1 (QZHT_DEFAULT). If a new file extension is specified, file names matching any one of the file extensions listed in File extension are renamed with the file extension prior to being sent to the cache targets. An escape message is sent if a new file extension is provided and Send assembled version equals 0 (QZHT_NO).

Offset to new file extension

The offset from the beginning of the request variable to the New file extension data, in bytes. The value must be greater than 0 or equal one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
-1	QZHT_DEFAULT: The default value is used for New file extension.

Note: An escape message is sent if Offset to new file extension is greater than 0 (indicating that a new file extension has been provided) and Send assembled version equals 0 (QZHT_NO).

Offset to file extensions

The offset from the beginning of the request variable to the File extension data, in bytes.

Parse and assemble

Specifies if files matching this publishing rule are parsed for wrappers and includes, and possibly sent through the page assembler. If the name is changed during wrapper parsing, the extension of the new name is used to determine if the file should be parsed for includes and sent through the page assembler. The value must equal one of the special values described below. The default value is 0 (QZHT_NO).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: Files matching this publishing rule are parsed and possibly sent through the page assembler.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: Files matching this publishing rule are not parsed. Note: Files that are not parsed are not sent through the page assembler.
-1	QZHT_DEFAULT: The default value is used.

Note: An escape message is sent if Parse and assemble equals 1 (QZHT_YES) and Read from data source equals 0 (QZHT_NO).

Publishing rule name

The name used by the publishing rule, left justified and padded with blanks if necessary.

Note: Publishing rule names must be unique for each triggered cache manager server. They are referenced by name, from Rule Sets associated with the same server.

Read from data source

Specifies if files matching this publishing rule are read from the data source when triggered. The value must be one of the special values described below. The default value is 1 (QZHT_YES).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: Files matching this publishing rule are read from the data source when triggered.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: Files matching this publishing rule are not read from the data source. Note: Not reading files from the data source is useful when triggers specify a file which is purely symbolic and need not correspond to an actual file, yet causes dependent files processing.
-1	QZHT_DEFAULT: The default value is used.

Send assembled version

Specifies if the assembled version of files matching this publishing rule are sent to cache targets when triggered and processed. The value must equal one of the special values described below. The default value is 0 (QZHT_NO).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: Assembled versions of files matching this publishing rule are sent to cache targets when triggered and processed.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: Assembled versions of files matching this publishing rule are not sent to cache targets.
-1	QZHT_DEFAULT: The default value is used.

Note: An escape message is sent if Send assembled version equals 1 (QZHT_YES) and Parse and assemble equals 0 (QZHT_NO).

Send data source version

Specifies if the data source version of files matching this publishing rule (the version read from the data source prior to assembly) are sent to cache targets when triggered and processed. The value must equal one of the special values described below. The default value is 1 (QZHT_YES) when Read from data source equals 1 (QZHT_YES), and 0 (QZHT_NO) when Read from data source equals 0 (QZHT_NO).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: Data source versions of files matching this publishing rule are sent to cache targets when triggered and processed.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: Data source versions of files matching this publishing rule are not sent to cache targets.
-1	QZHT_DEFAULT: The default value is used.

Note: An escape message is sent if Send data source version equals 1 (QZHT_YES) and Read from data source equals 0 (QZHT_NO).

Server name

The name used to identify the triggered cache manager server to which the description is associated (left justified and padded with blanks if necessary).

Error messages:

TCM7001 E

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7031 E

Request variable format is not valid.

TCM7033 E

Length of request variable is not valid for the specified format.

TCM703E E

Error code parameter is not valid.

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 E

A configuration file for triggered cache manager server &1 was not found.

TCM7293 E

A &1 using the name&2 already exists for triggered cache manager server &3.

TCM72C0 E

Triggered cache manager &1 name is not valid.

TCM72C3 E

Value passed to offset &1 is not valid.

TCM74C0 E

Triggered cache manager server name is not valid.

Add Triggered Cache Manager Rule Set (QzhtAddTCMRuleSet) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 request variable	Input	Char(*)
2 length of request variable	Input	Binary(4)
3 request variable format	Input	Char(8)
4 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

The *QzhtAddTCMRuleSet* API adds new rule sets to the configurations of triggered cache manager servers. New rule sets are referenced subsequently, by name, from trigger handler descriptions associated with the same server. New rule sets are utilized by all descriptions referencing them the next time the servers are started.

Note: Triggers are sent to trigger handlers which process them according to publishing rules. Custom publishing rule descriptions may be provided for the trigger handler through a rule set. If extensions of files identified in triggers match one of the extensions listed in a custom publishing rule, they will be processed according to that publishing rule. If extensions do not match any of the custom publishing rules, the file will be processed according to the default publishing rule.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

request variable

INPUT: CHAR(*)

The variable used to pass information to add a new rule set.

length of request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for request variable.

request variable format

INPUT: CHAR(8)

The format name of the request variable data. The following values must be used:

- RSDP0100: Basic information format for a rule set.

error code

I/O: CHAR(*)

The structure in which to return error information.

RSDP0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Char(32)	Rule set name
64	40	Binary(4)	Default property
68	44	Binary(4)	Offset to publishing rules
72	48	Binary(4)	Length of publishing rules
		Char(*)	Publishing rules

RSDP0100 format field descriptions:

Default property

Specifies if the new description is to become the default rule set for the triggered cache manager server specified in Server name. The value must equal one of the special values described below. If the value equals 1 (QZHT_YES), the default property on the current default rule set, if any, is set to 0 (QZHT_NO). The default value is 0 (QZHT_NO).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The description is to become the default rule set for this server.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The description is not to become the default rule set.
-1	QZHT_DEFAULT: The default value is used.

Note: Trigger handler descriptions, added or changed using the *DEFAULT special value, reference the rule set designated as default at the time they are added or changed.

Length of publishing rules

The length of the information for the Publishing rules entry.

Note: If Offset to Publishing rules equals 0 (QZHT_NONE) or -1 (QZHT_DEFAULT), this value must equal 0.

Offset to publishing rules

The offset from the beginning of the request variable to the Publishing rules data, in bytes. The value must be greater than 0, or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
-1	QZHT_DEFAULT: The default value is used for Publishing rules.

Publishing rules

A list of publishing rules associated with the triggered cache manager server referenced by the new Rule Set and used by trigger handlers, at startup, to direct how files are processed.

Descriptions must be listed by name, where each name is separated by one or more spaces, and padded with blanks if necessary. The default value is null, indicating that an empty rule set is described.

Note: If the value is null, Offset to publishing rules must equal 0 (QZHT_NONE) or -1 (QZHT_DEFAULT). See Offset to publishing rules for more details. An empty rule set referenced by a trigger handler causes it to process all triggers according to the default publishing rule. An escape message is sent if any referenced description does not currently exist.

Rule set object name

The name used by the new rule set (left justified and padded with blanks if necessary).

Note: Rule set names must be unique for each triggered cache manager server. They are referenced, by name, from trigger handler descriptions associated with the same server.

Server name

The name used to identify the triggered cache manager server to which the new description is associated (left justified and padded with blanks if necessary).

Error messages:

TCM7001 E

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7031 E

Request variable format is not valid.

TCM7033 E

Length of request variable is not valid for the specified format.

TCM703E E

Error code parameter is not valid.

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 E

A configuration file for triggered cache manager server &1 was not found.

TCM7293 E

A &1 using the name&2 already exists for triggered cache manager server &3.

TCM72C0 E

Triggered cache manager &1 name is not valid.

TCM72C3 E

Value passed to offset &1 is not valid.

TCM74C0 E

Triggered cache manager server name is not valid.

Add Triggered Cache Manager Trigger Handler Description (QzhtAddTCMTriggerHandlerDesc) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 request variable	Input	Char(*)
2 length of request variable	Input	Binary(4)
3 request variable format	Input	Char(8)
4 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

Use the QzhtAddTCMTriggerHandlerDesc API to add new trigger handler descriptions to the configurations of triggered cache manager servers. New trigger handler descriptions are utilized by the triggered cache manager servers after they are restarted. The API is a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

request variable

INPUT: CHAR(*)

The variable used to pass the information used to add a new trigger handler description.

length of request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for Request variable.

request variable format

INPUT: CHAR(8)

The format name of the Request variable data. The following values must be used:

- THDP0100: Basic information format for a trigger handler description.
- THDP0200: Detailed information format for an *UPDATE type trigger handler description.
- THDP0300: Detailed information format for a *PUBLISH type trigger handler description.

error code

I/O: CHAR(*)

The structure in which to return error information.

THDP0100 format:

Note: When using this format, trigger handler description are added using default values for all unspecified values (according to the type specified by *Trigger handler type*). See other trigger handler description formats for details regarding these default values.

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Char(32)	Trigger handler description name
64	40	Char(10)	Trigger handler type

Offset		Type	Field
Dec	Hex		
74	4A	Char(2)	Reserved
76	4C	Char(32)	Data source
108	6C	Binary(4)	Offset to cache targets
112	70	Binary(4)	Length of cache targets
116	74	Binary(4)	Offset to ack targets
120	78	Binary(4)	Length of ack targets
124	7C	Binary(4)	Offset to nack targets
128	80	Binary(4)	Length of nack targets
132	84	Binary(4)	Number of threads
		Char(*)	Cache targets
		Char(*)	Ack targets
		Char(*)	Nack targets

THDP0100 format field descriptions:

Ack targets

A list of acknowledgment target descriptions associated with the triggered cache manager server referenced by the new trigger handler description and used later, at server startup, to obtain information as to where successful process completion messages are sent. Descriptions must be listed by name, where each name is separated by one or more spaces, and padded with blanks if necessary. The default setting is to reference the descriptions currently designated as default acknowledgment target descriptions for the triggered cache manager server. If there are no descriptions currently designated as default, a null list is used. A null list indicates successful process completion messages are not sent.

Note: If a null list is specified, Offset to ack targets must equal 0 (QZHT_NONE), or -1 (QZHT_DEFAULT). See Offset to ack targets for more details. Messages concerning successful process completion of triggers referencing this handler are sent to all listed acknowledgment targets. An escape message is sent if any referenced description does not currently exist.

Cache targets

A list of cache target descriptions associated with the triggered cache manager server referenced by the new trigger handler description and used later, at server startup, to obtain information about the cache targets to which the trigger handler sends data. Descriptions must be listed by name, where each name is separated by one or more spaces, and padded with blanks if necessary. The default setting is to reference the descriptions currently designated as default cache target descriptions for the triggered cache manager server. If there are no descriptions currently designated as default, a null list is used. A null list indicates data is not sent to cache targets.

Note: If a null list is specified, Offset to ack targets must equal 0 (QZHT_NONE), or -1 (QZHT_DEFAULT). See Offset to ack targets for more details. Data processed by this trigger handler is sent to all listed cache targets. An escape message is sent if any referenced description does not currently exist.

Data source

The name of a data source description associated with the triggered cache manager server referenced by the new trigger handler description and used later, at server startup, to obtain information about the data source from which the trigger handler receives data. The value must be a data source description name, or one of the special values described below (left justified and

padded with blanks if necessary). The default setting references the description currently designated as the default data source description for the triggered cache manager server.

Special values and their meanings are as follows:

***DEFAULT**

QZHT_DEFAULT_CHAR: The default value is referenced.

Note: An escape message is sent if the referenced description does not currently exist, or if *DEFAULT is specified and there is currently no default data source description for the server.

Length of ack targets

The length of the information for the Ack targets entry.

Note: If *Offset to ack targets* equals 0 (QZHT_NONE), or -1 (QZHT_DEFAULT), this value must equal 0.

Length of cache targets

The length of the information for the Cache targets entry.

Note: If *Offset to cache targets* equals 0 (QZHT_NONE), or -1 (QZHT_DEFAULT), this value must equal 0.

Length of nack targets

The length of the information for the Nack targets entry.

Note: If *Offset to nack targets* equals 0 (QZHT_NONE), or -1 (QZHT_DEFAULT), this value must equal 0.

Nack targets

A list of acknowledgment target descriptions associated with the triggered cache manager server referenced by the new trigger handler description and used later, at server startup, to obtain information as to where failed process completion messages are sent. Descriptions must be listed by name, where each name is separated by one or more spaces, and padded with blanks if necessary. The default setting is to reference the descriptions currently designated as default acknowledgment target descriptions for the triggered cache manager server. If there are no descriptions currently designated as default, a null list is used. A null list indicates failed process completion messages are not sent.

Note: If a null list is specified, *Offset to nack targets* must equal 0 (QZHT_NONE), or -1 (QZHT_DEFAULT). See *Offset to nack targets* for more details. Messages concerning failed trigger process completion referencing this handler are sent to all listed acknowledgment targets. An escape message is sent if any referenced description does not currently exist.

Number of threads

The number of concurrent threads the triggered cache manager server spawns when processing triggers sent to this trigger handler. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below. The default value is 10.

Special values and their meanings are as follows:

-1	QZHT_DEFAULT: The default value is used.
----	--

Offset to ack targets

The offset from the beginning of the request variable to the Ack targets data, in bytes. The value must be greater than 0, or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NONE: A null list is specified for Ack targets indicating successful process completion messages are not sent.
-1	QZHT_DEFAULT: The default descriptions, if any, are referenced for Ack targets.

Offset to cache targets

The offset from the beginning of the request variable to the Cache targets data, in bytes. The value must be greater than 0, or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NONE: A null list is specified for Cache targets indicating data is not sent to cache targets.
-1	QZHT_DEFAULT: The default descriptions, if any, are referenced for Cache targets.

Offset to nack targets

The offset from the beginning of the request variable to the Nack targets data, in bytes. The value must be greater than 0, or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NONE: A null list is specified for Nack targets indicating failed process completion messages are not sent.
-1	QZHT_DEFAULT: The default descriptions, if any, are referenced for Nack targets.

Server name

The name used to identify the triggered cache manager server for which the new description is associated (left justified and padded with blanks if necessary).

Trigger handler description name

The name used by the new trigger handler description (left justified and padded with blanks if necessary).

Note: Trigger handler description names must be unique for each triggered cache manager server.

Trigger handler type

The trigger handler type (left justified and padded with blanks if necessary). The value must be one of the special values described below.

Special values and their meanings are as follows:

*UPDATE

QZHT_UPDATE_TYPE: A *UPDATE type is described.

*PUBLISH

QZHT_PUBLISH_TYPE: A *PUBLISH type is described.

THDP0200 format:

Offset		Type	Field
Dec	Hex		
0	0		Everything from THDP0100 format
136	88	Binary(4)	Cache request queue priority

Offset		Type	Field
Dec	Hex		
140	8C	Binary(4)	Trigger queue collapse policy

THDP0200 format field description:

Cache request queue priority

Specifies the trigger handler priority value when submitting requests to the cache targets. Lower values indicate higher priority. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below. The default value is 2^{31} (the lowest priority).

Special values and their meanings are as follows:

-1	QZHT_DEFAULT: The default value is used.
----	--

Note: Triggered cache manager servers queue the trigger handler requests to the cache targets and process them according to queue priority. Requests from trigger handlers with higher priority are processed before requests from trigger handlers with lower priority. The trigger handler queue priority can be changed while servers are active by using the *-chspriority* command in a trigger message.

Trigger queue collapse policy

Specifies if identical triggers waiting on the request queue, for this trigger handler, are collapsed. The value must equal one of the special values described below. The default value is 1 (QZHT_YES).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: Identical triggers are collapsed.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: Identical triggers are not collapsed.
-1	QZHT_DEFAULT: The default value is used.

Note: Only triggers using the *-objects* keyword can be collapsed. Identical triggers are those having an identical set of listed objects. The order of the listed objects is not important. Once a trigger handler begins processing a trigger, it is not collapsed.

THDP0300 format:

Offset		Type	Field
Dec	Hex		
0	0		Everything from THDP0100 format
136	88	Char(32)	Object dependency graph
168	A8	Char(32)	Rule set
200	C8	Binary(4)	Offset to traversal edge name
204	CC	Binary(4)	Length of traversal edge name
208	D0	Binary(4)	Offset to default included file

Offset		Type	Field
Dec	Hex		
212	D4	Binary(4)	Length of default included file
216	D8	Binary(4)	Include dependency information
220	DC	Binary(4)	Include triggered file information
224	E0	Binary(4)	Include cached file information
		Char(*)	Traversal edge type
		Char(*)	Default include file

THDPO300 format field description:

Default included file

The name of a file the trigger handler includes, by global default, as a replacement for included files that have not been triggered (when a local default file is not specified or available). The file name must be left justified and padded with blanks if necessary. The default value is null, indicating that a global default file name is not specified for this trigger handler description.

Note: If the value is null, Offset to default included file must equal 0 (QZHT_NONE), or -1 (QZHT_DEFAULT). See Offset to default included file for more details. The file specified as a global default must be triggered (at runtime) before it can be used.

Include cached file information

Specifies if a list of names for all files sent to cache targets, as a result of handling original trigger requests, is included in successful process completion messages. The value must equal one of the special values described below. The default value is 0 (QZHT_NO).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: A list of names is included.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: A list of names is not included.
-1	QZHT_DEFAULT: The default value is used.

Include dependency information

Specifies if information concerning all dependent files, assembled into triggered files as a result of handling original trigger request, is included in successful process completion messages. The value must equal one of the special values described below. The default value is 0 (QZHT_NO).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: Information is included.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: Information is not included.
-1	QZHT_DEFAULT: The default value is used.

Include triggered file information

Specifies if a list of names for all files triggered, as a result of handling original trigger requests, is included in successful process completion messages. The value must equal one of the special values described below. The default value is 0 (QZHT_NO).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: A list of names is included.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: A list of names is not included.
-1	QZHT_DEFAULT: The default value is used.

Length of default included file

The length of information for the Default included file entry.

Note: If Offset to traversal edge type equals 0 (QZHT_NONE), or -1 (QZHT_DEFAULT), this value must equal 0.

Object dependency graph

The name of an object dependency graph description associated with the triggered cache manager server referenced by the new trigger handler description and used later, at server startup, to identify which object dependency graph is used by the handler to record and obtain object dependency information. The value must be an object dependency graph description name, or one of the special values described below (left justified and padded with blanks if necessary). The default setting is to reference the description currently designated as the default object dependency graph description for the triggered cache manager server.

Special values and their meanings are as follows:

*DEFAULT

QZHT_DEFAULT_CHAR: The default is referenced.

Note: An object dependency graph description is required. An escape message is sent if the referenced description does not currently exist, or if *DEFAULT is specified and there is currently no default object dependency graph description for the server.

Offset to default included file

The offset from the beginning of the request variable to the Default included file data, in bytes. The value must be greater than 0, or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
-1	QZHT_DEFAULT: The default value for <i>Default included file</i> is used.

Offset to traversal edge type

The offset from the beginning of the request variable to the Traversal edge type data, in bytes. The value must be greater than 0, or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
-1	QZHT_DEFAULT: The default value for <i>Traversal edge type</i> is used.

Rule set

The name of a rule set associated with the triggered cache manager server referenced by the new trigger handler description and used later, at server startup, to identify which publishing rules are used by the handler. The value must be a rule set name, or one of the special values described below (left justified and padded with blanks if necessary).

Special values and their meanings are as follows:

***DEFAULT**

QZHT_DEFAULT_CHAR: The default rule set currently associated with the triggered cache manager server is referenced.

***NONE**

QZHT_NONE_CHAR: No rule set is referenced by the trigger handler.

Note: Triggers that specify files that do not match any publishing rules, in the specified rule set, are processed according to the default publishing rule. That is, they are read from the data source and sent to all cache targets (they are not parsed). An empty rule set has the same affect as not specifying a rule set at all. An escape message is sent if the referenced description does not currently exist, or if *DEFAULT is specified and there is currently no default Rule Set for the server.

Traversal edge type

The name of an object dependency graph (ODG) edge type used by the trigger handler to determine object dependencies when assembling files. The edge type name must be left justified and padded with blanks if necessary. The default name is **composition**.

Note: If the value is null, Offset to traversal edge type must equal 0 (QZHT_NONE), or -1 (QZHT_DEFAULT). Only edges of the specified type are traversed by this trigger handler to determine object dependencies.

Error messages:

TCM7001 E

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7031 E

Request variable format is not valid.

TCM7033 E

Length of request variable is not valid for the specified format.

TCM703E E

Error code parameter is not valid

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 E

A configuration file for triggered cache manager server &1 was not found.

TCM7222 E

A default &1 is not designated for triggered cache manager server &2.

TCM7293 E

A &1 using the name &2 already exists for triggered cache manager server &3.

TCM72C0 E

Triggered cache manager &1 name is not valid.

TCM72C1 E

Triggered cache manager &1 type is not valid.

TCM72C2 E

Triggered cache manager description type &1 cannot be specified when using data format &2.

TCM72C3 E

Value passed to offset &1 is not valid.

TCM74C0 E

Triggered cache manager server name is not valid.

Change Triggered Cache Manager Acknowledgment Target Description (QzhtChgTCMAckTargetDesc) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 request variable	Input	Char(*)
2 length of request variable	Input	Binary(4)
3 request variable format	Input	Char(8)
4 server name	Input	Char(32)
5 description name	Input	Char(32)
6 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

Use the QzhtChgTCMAckTargetDesc API to change acknowledgment target descriptions associated with triggered cache manager servers. Changes made to acknowledgment target descriptions are utilized by all trigger handler descriptions referencing them (the next time the servers are started). The API is a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

request variable

INPUT: CHAR(*) The variable used to pass the information used to change an acknowledgment target description.

length of request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for request variable.

request variable format

INPUT: CHAR(8)

The format name of the Request variable data. The following values must be used:

- ATDP0100: Basic information format for an acknowledgment target description.
- ATDP0200: Detailed information format for an *HTTP1 type acknowledgment target description.
- ATDP0210: Detailed information format for an *HTTP2 type acknowledgment target description.

server name

INPUT: CHAR(32)

The name used to identify the triggered cache manager server for which the description is changed (left justified and padded with blanks if necessary).

description name

INPUT: CHAR(32)

The name used to identify the changed acknowledgment target description (left justified and padded with blanks if necessary).

error code

I/O: CHAR(*)

The structure in which to return error information.

ATDP0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Char(32)	Acknowledgement target description name
64	40	Char(10)	Acknowledgement target type
74	4A	Char(2)	Reserved
76	4C	Binary(4)	Default property
80	50	Binary(4)	Number of threads
84	54	Binary(4)	Initial state

ATDP0100 format field descriptions:

Note: Acknowledgment target descriptions are added using default values for all unspecified values according to the type specified by Acknowledgment target type). See other acknowledgment target description formats for details regarding these default values.

Acknowledgement target description name

The name used by the new acknowledgment target description (left justified and padded with blanks if necessary). The value must be a description name, or be one of the special values described below.

Special values and their meanings are as follows:

***SAME**

QZHT_NO_CHANGE_CHAR: The current name is not changed.

Note: Acknowledgment target description names must be unique for each triggered cache manager server.

Acknowledgement target type

The added acknowledgment target description type (left justified and padded with blanks if necessary). The value must be one of the special values described below.

Special values and their meanings are as follows:

***SAME**

QZHT_NO_CHANGE_CHAR: The description is not changed to a different type.

***HTTP1**

QZHT_HTTP_TYPE1: An *HTTP1 type is added.

*HTTP2

QZHT_HTTP_TYPE2: An *HTTP2 type is added.

Note: If a type is specified, that is different than the current type, certain information is discarded if it cannot be mapped to values for the new description type. See other acknowledgment target description formats for details.

Default property

Specifies if the new description is the default acknowledgment target description for the triggered cache manager server specified in Server name. The value must equal one of the special values described below. The default value is 0 (QZHT_NO).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The description is the default cache target description for this server.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The description is not the default cache target description.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The current value is be changed.

Note: Multiple default acknowledgment target descriptions are allowed. Trigger handler descriptions, added or changed using the *DEFAULT special value, reference all acknowledgment target descriptions designated as default at the time the trigger handler descriptions are added or changed.

Initial state

Specifies the state that the triggered cache manager server request processor, for this acknowledgment target, is in at server startup. The value must equal one of the special values described below. The default value is 1 (QZHT_ENABLED).

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The request processor is enabled at server startup.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The request processor is disabled at server startup.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The current value is not changed.

Note: The state of a acknowledgment target request processor can be changed while the triggered cache manager server is active by using the *-chack* command in a trigger message.

Number of threads

The number of concurrent threads the triggered cache manager server spawns when processing requests sent to this acknowledgment target. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below. The default value is 5.

Special values and their meanings are as follows:

-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The current value is not changed.

Server name

The name used to identify the triggered cache manager server for which the description is associated (left justified and padded with blanks if necessary). The value must be a server name,

or one of the special values described below. If a server name is specified, that is different than the one with which the description is currently associated, it is removed from its current association and added for the new server.

Special values and their meanings are as follows:

***SAME**

QZHT_NO_CHANGE_CHAR: The current server association is not changed.

Note: An escape message is sent if the description is removed from its current association (while being referenced by other descriptions).

ATDP0200 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(256)	Everything from ATDP0100 format
88	58	Binary(4)	HTTP IP interface
344	158	Binary(4)	HTTP TCP port
348	15C	Binary(4)	Offset to HTTP URI root
352	160	Binary(4)	Length of HTTP URI root
356	164	Binary(4)	HTTP keepalive
360	168	Binary(4)	Timeout
		Char(*)	HTTP URI root

ATDP0200 format field descriptions:

Note: If the current description type is not *HTTP1, all information currently stored for the description, that is not mapped to one the following entries, is discarded.

HTTP IP interface

The IP host name or address of the system hosting an HTTP server that accepts completion messages (left justified and padded with blanks if necessary). The value must be a host name (for example, server.mycompany.com), dotted address (for example, 192.168.3.57), or one of the special values described below. If a host name is specified, it must use proper naming conventions as defined by RFC 1034, Domain Names - Concepts and Facilities. If a dotted address is specified, it must use proper IP version 4 address conventions as defined by RFC 791, Internet Protocol. The default value is 127.0.0.1, the local system loopback interface.

Special values and their meanings are as follows:

***DEFAULT**

QZHT_DEFAULT_CHAR: The default value is used.

***SAME**

QZHT_NO_CHANGE_CHAR: The following rules are used to map current description values for reuse:

If the current type is:		Details
*HTTP1	use current HTTP IP interface value	No change is made.
*HTTP2	use IP interface value from the host description currently referenced by HTTP host	An exception occurs if the referenced description does not exist. Refers to ATDP0210 format.

HTTP keepalive

Specifies if connections to HTTP Server are kept open for reuse after data is transferred. The value must equal one of the special values described below. The default value is 0 (QZHT_DISABLED).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The connection is kept open after messages are sent.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The connection is closed after messages are sent.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The current value is not changed.

Note: HTTP Server must support keepalive for this option to work properly.

HTTP TCP port

The TCP port number upon which HTTP Server listens for incoming requests. The value must be greater than 0 and less than 65536, or equal to one of the special values described below. The default value is 80.

Special values and their meanings are as follows:

-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The current value is not changed.

Note: The TCP port number is used in combination with the IP host name or address in HTTP IP interface to establish communication with HTTP Server acknowledgment target.

HTTP URI root

The path of HTTP Server URI that is the root of this acknowledgment target (left justified and padded with blanks if necessary). The value must be a path acceptable to HTTP Server. The default path is / .

If Offset to HTTP URI root equals -2 (QZHT_NO_CHANGE), the current path is not changed.

Note: All completion message requests sent to the acknowledgment target are prepended with this path. If the value is null, Offset to HTTP URI root must equal 0 (QZHT_NONE), -1 (QZHT_DEFAULT), or -2 (QZHT_NO_CHANGE). See Offset to HTTP URI root for more details.

Length of HTTP URI root

The length of the information for the HTTP URI root entry.

Note: If Offset to HTTP URI root equals 0 (QZHT_NONE), -1 (QZHT_DEFAULT), or -2 (QZHT_NO_CHANGE), this value must equal 0. If Offset to HTTP URI root equals 0 (QZHT_NONE), -1 (QZHT_DEFAULT), or -2 (QZHT_NO_CHANGE), this value must equal 0.

Offset to HTTP URI root

The offset from the beginning of the request variable to the HTTP URI root data, in bytes. The value must be greater than 0, or equal to one of the special values defined below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
---	---------------------------------------

-1	QZHT_DEFAULT: The default value is used for HTTP URI root.
-2	QZHT_NO_CHANGE: The current path for HTTP URI root is not changed.

Timeout

The number of seconds the triggered cache manager server waits before canceling a read operation on sockets and ending transactions, with the HTTP host, in error. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below. The default value is 0 (QZHT_DISABLED).

Special values and their meanings are as follows:

0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The time out operation is disabled. The triggered cache manager server should never cancel a read operation.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The current value is not changed.

ATDP0210 format:

Offset		Type	Field
Dec	Hex		
0	0		Everything from ATDP0100 format
88	58	Char(32)	HTTP host
120	78	Binary(4)	HTTP TCP port
124	7C	Binary(4)	Offset to HTTP URI root
128	80	Binary(4)	Length of HTTP URI root
132	84	Binary(4)	HTTP keepalive
136	88	Binary(4)	Timeout
		Char(*)	HTTP URI root

ATDP0210 format field descriptions:

Note: If the current description type is not *HTTP2, all information currently stored for the description, that is not mapped to one the following entries, is discarded.

HTTP host

The name of a host description associated with the triggered cache manager server that is referenced by the new acknowledgment target description and used later, at server startup, to obtain information about the system hosting an HTTP server accepting completion messages. The value must be a host description name, or one of the special values described below (left justified and padded with blanks if necessary). The default value is to reference the description currently designated as the default host description for the triggered cache manager server.

Special values and their meanings are as follows:

*DEFAULT

QZHT_DEFAULT_CHAR: The default value is referenced.

*SAME

QZHT_NO_CHANGE_CHAR: The following rules are used to map current description values for reuse:

If the current type is:		Details
*HTTP1	the default host description, currently associated with the triggered cache manager server is, referenced.	
*HTTP2	the description, currently referenced for HTTP host, is not changed	No change is made.

Note: See HTTP TCP port for more information. An escape message is sent if the referenced description does not currently exist, or if *DEFAULT is specified and there is currently no default host description for the server.

HTTP keepalive

Specifies if connections to HTTP Server are kept open for reuse after completion messages are sent. The value must equal one of the special values described below. The default value is 0 (QZHT_DISABLED).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The connection is kept open after messages are sent.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The connection is closed after messages are sent.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The current value is not changed.

Note: HTTP Server must support keepalive for this option to work properly.

HTTP TCP port

The TCP port number upon which HTTP Server listens for incoming completion messages. The value must be greater than 0 and less than 65536, or equal to one of the special values described below. The default value is 80.

Special values and their meanings are as follows:

-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The current value is not changed.

Note: The TCP port number is used in combination with information obtained at server startup from the host description specified in HTTP host to establish communication with HTTP Server acknowledgment target.

HTTP URI root

The path of HTTP Server URI that is the root of this acknowledgment target (left justified and padded with blanks if necessary). The value must be a path acceptable to HTTP Server. The default path is / .

If Offset to HTTP URI root equals -2 (QZHT_NO_CHANGE), the current path is not changed.

Note: All requests to send completion messages to this acknowledgment target are prepended with this path. If the value is null, Offset to HTTP URI root must equal 0 (QZHT_NONE), -1 (QZHT_DEFAULT), or -2 (QZHT_NO_CHANGE). See Offset to HTTP URI root for more details.

Length of HTTP URI root

The length of the information for the HTTP URI root entry.

Note: If Offset to HTTP URI root equals 0 (QZHT_NONE), -1 (QZHT_DEFAULT), or -2 (QZHT_NO_CHANGE), this value must equal 0.

Offset to HTTP URI root

The offset from the beginning of the request variable to the HTTP URI root data, in bytes. The value must be greater than 0, or equal to one of the special values defined below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
-1	QZHT_DEFAULT: The default path is used for HTTP URI root.
-2	QZHT_NO_CHANGE: The current path for HTTP URI root is not changed.

Timeout

The number of seconds the triggered cache manager server waits before canceling a read operation on sockets and ending transactions, with the HTTP host, in error. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below. The default value is 0 (QZHT_DISABLED).

Special values and their meanings are as follows:

0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The time out operation is disabled. The triggered cache manager server should never cancel a read operation.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The current value is not changed.

Error messages:

TCM7001 E

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7031 E

Request variable format is not valid.

TCM7033 E

Length of request variable is not valid for the specified format.

TCM703E E

Error code parameter is not valid.

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 D

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 D

A configuration file for triggered cache manager server &1 was not found.

TCM7222 D

A default &1 is not designated for triggered cache manager server &2.

TCM7290 D

&1 &2 was not found for triggered cache manager server &3.

TCM7293 D

A &1 using the name &2 already exists for triggered cache manager server &3.

TCM72C0 D

Triggered cache manager &1 name is not valid.

TCM72C1 D

Triggered cache manager &1 type is not valid.

TCM72C2 D

Triggered cache manager description type &1 cannot be specified when using data format &2.

TCM72C3 D

Value passed to offset &1 is not valid.

TCM74C0 D

Triggered cache manager server name is not valid.

Change Triggered Cache Manager Basic Configuration (QzhtChgTCMBasicConfig) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 request variable	Input	Char(*)
2 length of request variable	Input	Binary(4)
3 request variable format	Input	Char(8)
4 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

Use the QzhtChgTCMBasicConfig API to change the basic configuration information for a triggered cache manager server. The API is a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

request variable

INPUT: CHAR(*)

The variable used to pass the information used to change the basic configuration information for a triggered cache manager server.

length of request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for request variable.

request variable format

INPUT: CHAR(8)

The format name of the request variable data. The following values must be used:

- INDP0200: Detailed information format for server data.

error code

I/O: CHAR(*)

The structure in which to return error information.

INDP0200 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Binary(4)	Autostart property
36	24	Binary(4)	Local TCP port
40	28	Binary(4)	Offset to root directory
44	2C	Binary(4)	Length of root directory
48	30	Binary(4)	Maximum number of retries
52	34	Binary(4)	Defer time between retries
56	38	Binary(4)	Memory buffer size
		Char(*)	Root directory

*Field descriptions:***Autostart property**

Specifies if the new triggered cache manager server is to start when startup of *AUTOSTART triggered cache manager servers, is requested. Usually *AUTOSTART servers are requested to start when TCP/IP is started, however they may also be requested via the STRTCPSVR command or QzhtStrTCMServer API. The default value is 0 (QZHT_NO).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The server is set to start when startup of *AUTOSTART servers is requested.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The server is not set to start when startup of *AUTOSTART servers is requested.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: No change is made to the current auto start setting.

Defer time between retries

The number of seconds the triggered cache manager server is to wait between retry attempts for a failing action. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below. The default value is 60.

Special values and their meanings are as follows:

0	QZHT_NONE: No defer time is to be used.
-1	QZHT_DEFAULT: The default value is to be used.
-2	QZHT_NO_CHANGE: The current defer time value is not to be changed.

Note: If maximum number of retries is equal to 0 (QZHT_NONE), this value must be equal to 0 (QZHT_NONE).

Length of root directory

The length of the information for the root directory entry.

Note: If offset to root directory is equal to 0 (QZHT_NONE), -1 (QZHT_DEFAULT), or -2 (QZHT_NO_CHANGE), this value must be equal to 0.

Maximum number of retries

The maximum number of time the server will attempt to retry a failing action. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below. The default is to retry failing actions until successful (QZHT_NOMAX).

Special values and their meanings are as follows:

0	QZHT_NONE: Do not retry failing actions.
-1	QZHT_DEFAULT: The default action is specified.
-2	QZHT_NO_CHANGE: The current maximum number of retries value is not to be changed.
-3	QZHT_NOMAX: Failing actions are to be retried until successful.

Memory buffer size

The maximum amount of working data the triggered cache manager server attempts to store in memory, in bytes. The value must be greater than -1 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below. A value equal to 0 indicates that the new triggered cache manager server should attempt to run in the least memory possible. The default value is 10,000,000 (or 1×10^7).

Special values and their meanings are as follows:

-1	QZHT_DEFAULT: The default value is to be used.
-2	QZHT_NO_CHANGE: The current memory buffer size value is not to be changed.

Offset to root directory

The offset from the beginning of the request variable to the Root directory data, in bytes. The value must be greater than 0, or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -2 (QZHT_NO_CHANGE).
-1	QZHT_DEFAULT: The default value is to be used for Root directory. The value for Root directory is null.
-2	QZHT_NO_CHANGE: The current root directory name is not to be changed for Root directory. The value for Root directory is null.

Root directory

The name of the local file system directory in which the triggered cache manager server is to maintain its persistent record of incoming transactions. The value must be null, or specify a directory name, left justified and padded with blanks if necessary. The default name is /QIBM/UserData/TCM/{servername}/ root, where {server-name} is the name of the server as defined by Server name.

Note: If the value is null, Offset to root directory must be set to 0 (QZHT_NONE), -1 (QZHT_DEFAULT), or -2 (QZHT_NO_CHANGE).

Server name

The name used to identify the server for which the configuration change is made (left justified and padded with blanks if necessary).

Error messages:

TCM7001 E

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7031 E

Request variable format is not valid.

TCM7033 E

Length of request variable is not valid for the specified format.

TCM703E E

Error code parameter is not valid

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 E

A configuration file for triggered cache manager server &1 was not found.

TCM72C3 E

Value passed to offset &1 is not valid.

TCM74C0 E

Triggered cache manager server name is not valid.

Change Triggered Cache Manager Cache Target Description (QzhtChgTCMCacheTargetDesc) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 request variable	Input	Char(*)
2 length of request variable	Input	Binary(4)
3 request variable format	Input	Char(8)
4 server name	Input	Char(32)
5 description name	Input	Char(32)
6 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

Use the QzhtChgTCMCacheTargetDesc API to change cache target description associated with triggered cache manager servers. Changes made to cache target descriptions are utilized by all trigger handler descriptions referencing them (the next time the servers are started). The API is a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

request variable

INPUT: CHAR(*)

The variable used to pass the information used to change a cache target description. See Cache target description formats for more information.

length of request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for request variable.

request variable format

INPUT: CHAR(8)

The format name of the Request variable data. See Cache target description formats for more information. The following values must be used:

- CTDP0100: Basic information format for a cache target description.
- CTDP0200: Detailed information format for an *IFS type cache target description.
- CTDP0300: Detailed information format for a *HTTP1 type cache target description.
- CTDP0310: Detailed information format for a *HTTP2 type cache target description.
- CTDP0500: Detailed information format for a *ECCP1 type cache target description.
- CTDP0510: Detailed information format for a *ECCP2 type cache target description.

server name

INPUT: CHAR(32)

The name used to identify the server for which the description is changed (left justified and padded with blanks if necessary).

description name

CHAR(32)

The name used to identify which cache target description information is changed (left justified and padded with blanks if necessary).

error code

I/O: CHAR(*)

The structure in which to return error information.

CTDP0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Char(32)	Cache target description name
64	40	Char(10)	Cache target type
74	4A	Char(2)	Reserved
76	4C	Binary(4)	Default property
80	50	Binary(4)	Number of threads
84	54	Binary(4)	Initial state

CTDP0100 format field descriptions:

Note: If a description type is specified in Cache target type that is different than the current description type, the description is changed using default values for all unspecified values according to the specified type). See other cache target description formats for details regarding these default values.

Cache target description name

The name used by the cache target description (left justified and padded with blanks if necessary). The value must be a description name, or one of the special values described below.

Special values and their meanings are as follows:

***SAME**

QZHT_NO_CHANGE_CHAR: The current name is not changed.

Note: Cache target description names must be unique for each triggered cache manager server. If the name is changed, while the description is referenced by trigger handler description, they too are changed to reference the new name.

Cache target type

The type of cache target description that is added (left justified and padded with blanks if necessary). The value must be one of the special values described below.

Special values and their meanings are as follows:

***SAME**

QZHT_NO_CHANGE_CHAR: The description is not changed.

***IFS** QZHT_IFS_TYPE: The description is changed to an *IFS type cache target description.

***HTTP1**

QZHT_HTTP_TYPE1: The description is changed to an *HTTP1 type cache target description.

***HTTP2**

QZHT_HTTP_TYPE2: The description is changed to an *HTTP2 type cache target description.

Note: If a type is specified, that is different than the current type, certain information is discarded if it cannot be mapped to values for reuse by the new description type. See other cache target description formats for details. Cache target description types of *ECCP1 and *ECCP2 must be added using the CTDP0500 and CTDP0510 formats, respectively.

Default property

Specifies if the new description will become the default cache target description for the triggered cache manager server specified in Server name. The value must equal one of the special values described below. The default value is 0 (QZHT_NO).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The description becomes the default cache target description for this server.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The description is not the default cache target description.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The current value is not changed.

Note: Multiple default cache target descriptions are allowed. Trigger handler descriptions added or changed using the *DEFAULT special value reference all cache target descriptions designated as default at the time the trigger handler descriptions are added or changed.

Initial state

Specifies the state that the triggered cache manager server request processor, for this cache target, is in at server startup. The value must equal one of the special values described below. The default value is 1 (QZHT_ENABLED).

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The request processor is enabled at server startup.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The request processor is disabled at server startup.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The current value is not changed.

Note: The state of a cache target request processor can be changed while the triggered cache manager server is active by using the `-chsink` command in a trigger message.

Number of threads

The number of concurrent threads the triggered cache manager server spawns when processing requests sent to this cache target. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below. The default value is 5.

Special values and their meanings are as follows:

-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The current value is not changed.

Server name

The name used to identify the triggered cache manager server for which the description is associated (left justified and padded with blanks if necessary). The value must be a server name, or one of the special values described below. If a server name is specified, that is different than the one with which the description is currently associated, it is removed from its current association and added for the new server.

Special values and their meanings are as follows:

*SAME

QZHT_NO_CHANGE_CHAR: The current server association is not changed.

Note: An escape message is sent if the description is removed from its current association while being referenced by other descriptions.

CTDP0200 format:

Offset		Type	Field
Dec	Hex		
0	0		Everything from CTDP0100 format
88	58	Binary(4)	Offset to local directory root
92	5C	Binary(4)	Length of local directory root
		Char(*)	Local directory root

CTDP0200 format field descriptions:

Note: If the current description type is not *IFS, all information currently stored for the description (that is not mapped to one the following entries) is discarded.

Length of local directory root

The length of the information for the Local directory root entry.

Note: If Offset to local directory root equals 0 (QZHT_NONE), -1 (QZHT_DEFAULT), or -2 (QZHT_NO_CHANGE) this value must equal 0.

Local directory root

The path to the local file system directory that is the root of this cache target (left justified and padded with blanks if necessary). The value must be a path acceptable to the root (/) file system of the local iSeries system. If a path is provided, but does not specify a absolute path (does not start with /), the path is prepended with the default path. The default path is /QIBM/UserData/TCM/{Server name} where {Server name} is the name of the triggered cache manager server as defined by Server name.

If Offset to local directory root equals -2 (QZHT_NO_CHANGE), the following rules are used to map current description values for reuse:

If the current type is:		Details
*IFS	use current Local directory root path	No change is made.
*HTTP1 or *HTTP2	use current HTTP URI root path	Refers to CTDP0300 or DSDP0310 format.
*ECCP1 or *ECCP2	use current HTTP cluster URI root path	Refers to CTDP0500 or CTDP0510 format.
any other	use default path	

Note: All file requests from the cache target must have this path prepended to the file name, even if an absolute file path is specified. If the value is null, Offset to local directory root must equal 0 (QZHT_NONE), -1 (QZHT_DEFAULT), or -2 (QZHT_NO_CHANGE). See Offset to local directory root for more details.

Offset to local directory root

The offset from the beginning of the request variable to the Local directory root data, in bytes. The value must be greater than 0, or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
-1	QZHT_DEFAULT: The default path is used for Local directory root.
-2	QZHT_NO_CHANGE: If the current description type is *IFS, the current path for Local directory root is not changed. If the current description type is not *IFS, see details for Local directory root on how current description values are mapped for reuse by the new description type.

CTDP0300 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(256)	Everything from CTDP0100 format
88	58	Binary(4)	HTTP IP interface

Offset		Type	Field
Dec	Hex		
344	158	Binary(4)	HTTP TCP port
348	15C	Binary(4)	Offset to HTTP URI root
352	160	Binary(4)	Length of HTTP URI root
356	164	Binary(4)	HTTP keepalive
360	168	Binary(4)	Timeout
		Char(*)	HTTP URI root

CTDP0300 format field descriptions:

Note: If the current description type is not *HTTP1, all information currently stored for the description (that is not mapped to one the following entries) is discarded.

HTTP IP interface

The IP host name or address of the system hosting an HTTP server cache target (left justified and padded with blanks if necessary). The value must be a host name (for example, server.mycompany.com), dotted address (for example, 192.168.3.57), or one of the special values described below. If a host name is specified, it must use proper naming conventions as defined by RFC 1034, Domain Names - Concepts and Facilities. If a dotted address is specified, it must use proper IP version 4 address conventions as defined by RFC 791, Internet Protocol. The default value is 127.0.0.1, the local system loopback interface.

Special values and their meanings are as follows:

*DEFAULT

QZHT_DEFAULT_CHAR: The default value is used.

*SAME

QZHT_NO_CHANGE_CHAR: The following rules are used to map current description values for reuse:

If the current type is:		Details
*HTTP1	use current HTTP IP interface value	No change is made.
*HTTP2	use IP interface value from the host description currently referenced by HTTP host	An exception occurs if the referenced description does not exist. Refers to the CTDP0310 format.
*ECCP1	use current ECCP IP interface value	Refers to CTDP0500 format.
*ECCP2	use IP interface value from the host description currently referenced by ECCP host	An exception occurs if the referenced description does not exist. Refers to CTDP0510 format.
any other	use default value	

HTTP keepalive

Specifies if connections to HTTP Server are kept open for reuse after data is transferred. The value must equal one of the special values described below. The default value is 0 (QZHT_DISABLED).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The connection is kept open after data transfer.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The connection is closed after data transfer.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: If the current description type is *HTTP1 or *HTTP2, the current value is not changed. If the current description type is not *HTTP1 or *HTTP2, the default value is used.

Note: HTTP Server must support keepalive for this option to work properly.

HTTP TCP port

The TCP port number upon which HTTP Server listens for incoming requests. The value must be greater than 0 and less than 65536, or equal to one of the special values described below. The default value is 80.

Special values and their meanings are as follows:

-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The following rules are used to map current description values for reuse:

If the current type is:		Details
*HTTP1 or *HTTP2	use current HTTP TCP port value	No change is made.
*ECCP1 or *ECCP2	use current ECCP TCP port value	Refers to CTDP0500 or CTDP0510 format.
any other	use default value	

Note: The TCP port number is used in combination with the IP host name or address in HTTP IP interface to establish communication with HTTP Server cache target.

HTTP URI root

The path of HTTP Server URI that is the root of this cache target (left justified and padded with blanks if necessary). The value must be a path acceptable to HTTP Server. The default path is / .

If Offset to HTTP URI root equals -2 (QZHT_NO_CHANGE), the following rules are used to map current description values for reuse:

If the current type is:		Details
*IFS	use current Local directory root path	Refers to CTDP0200 format.
*HTTP1 or *HTTP2	use current HTTP URI root path	No change is made.
*ECCP1 or *ECCP2	use current HTTP cluster URI root path	Refers to CTDP0500 or CTDP0510 format.
any other	use default path	

Note: All file requests from this cache target have this path inserted into the URI, even if an absolute file path is specified. If the value is null, Offset to HTTP URI root must equal 0 (QZHT_NONE), -1 (QZHT_DEFAULT), or -2 (QZHT_NO_CHANGE). See Offset to HTTP URI root for more details.

Length of HTTP URI root

The length of the information for the HTTP URI root entry.

Note: If Offset to HTTP URI root equals 0 (QZHT_NONE), -1 (QZHT_DEFAULT), or -2 (QZHT_NO_CHANGE), this value must equal 0.

Offset to HTTP URI root

The offset from the beginning of the request variable to the HTTP URI root data, in bytes. The value must be greater than 0, or equal to one of the special values defined below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
-1	QZHT_DEFAULT: The default value is used for HTTP URI root.
-2	QZHT_NO_CHANGE: If the current description type is *HTTP1 or *HTTP2, the current path for HTTP URI root is not changed. If the current description type is not *HTTP1 or *HTTP2, see details for HTTP URI root on how current description values are mapped for reuse by the new description type.

Timeout

The number of seconds the triggered cache manager server waits before canceling a read operation on sockets and ending transactions, with the HTTP host, in error. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below. The default value is 0 (QZHT_DISABLED).

Special values and their meanings are as follows:

0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The time out operation is disabled. The triggered cache manager server should never cancel a read operation.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: If the current description type is *HTTP1 or *HTTP2, the current value is not changed. If the current description type is not *HTTP1 or *HTTP2, the default value is used.

CTDP0310 format:

Offset		Type	Field
Dec	Hex		
0	0		Everything from CTD0100 format
88	58	Char(32)	HTTP host
120	78	Binary(4)	HTTP TCP port
124	7C	Binary(4)	Offset to HTTP URI root
128	80	Binary(4)	Length of HTTP URI root
132	84	Binary(4)	HTTP keepalive
136	88	Binary(4)	Timeout
		Char(*)	HTTP URI root

CTDP0310 format field descriptions:

Note: If the current description type is not *HTTP2, all information currently stored for the description, that is not mapped to one the following entries, is discarded.

HTTP host

The name of a host description associated with the triggered cache manager server that is referenced by the new cache target description and used later, at server startup, to obtain information about the system hosting an HTTP server cache target. The value must be a host description name, or one of the special values described below (left justified and padded with blanks if necessary). The default value is to reference the description currently designated as the default host description for the triggered cache manager server.

Special values and their meanings are as follows:

*DEFAULT

QZHT_DEFAULT_CHAR: The default value is referenced.

*SAME

QZHT_NO_CHANGE_CHAR: The following rules are used to map current description values for reuse:

If the current type is:		Details
*HTTP2	the description currently referenced for HTTP host is not changed	No change is made.
*ECCP2	the description currently referenced for ECCP host is referenced	An exception occurs if the referenced description does not exist. Refers to CTDP0510 format.
any other	the default host description currently associated with the triggered cache manager server is referenced	

Note: See HTTP TCP port for more information. An escape message is sent if the referenced description does not currently exist, or if *DEFAULT is specified and there is currently no default host description for the server.

HTTP keepalive

Specifies if connections to HTTP Server are kept open for reuse after data is transferred. The value must equal one of the special values described below. The default value is 0 (QZHT_DISABLED).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The connection is kept open after data transfer.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The connection is closed after data transfer.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: If the current description type is *HTTP1 or *HTTP2, the current value is not changed. If the current description type is not *HTTP1 or *HTTP2, the default value is used.

Note: HTTP Server must support keepalive for this option to work properly.

HTTP TCP port

The TCP port number upon which HTTP Server listens for incoming requests. The value must be greater than 0 and less than 65536, or equal to one of the special values described below. The default value is 80.

Special values and their meanings are as follows:

-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The following rules are used to map current description values for reuse:

If the current type is:		Details
*HTTP1 or *HTTP2	use current HTTP TCP port value	No change is made.
*ECCP1 or *ECCP2	use current ECCP TCP port value	Refers to CTDP0500 or CTDP0510 format.
any other	use default value	

Note: The TCP port number is used in combination with information obtained at server startup from the host description specified in HTTP host to establish communication with HTTP Server cache target.

HTTP URI root

The path of HTTP Server URI that is the root of this cache target (left justified and padded with blanks if necessary). The value must be a path acceptable to HTTP Server. The default path is / .

If Offset to HTTP URI root equals -2 (QZHT_NO_CHANGE), the following rules are used to map current description values for reuse:

If the current type is:		Details
*IFS	use current Local directory root path	Refers to CTDP0200 format.
*HTTP1 or *HTTP2	use current HTTP URI root path	No change is made.
*ECCP1 or *ECCP2	use current HTTP cluster URI root path	Refers to CTDP0500 or CTDP0510 format.
any other	use default path	

Note: All file requests sent to the cache target have this path inserted into the URI, even if an absolute file path is specified. If the value is null, Offset to HTTP URI root must equal 0 (QZHT_NONE), -1 (QZHT_DEFAULT), or -2 (QZHT_NO_CHANGE). See Offset to HTTP URI root for more details.

Length of HTTP URI root

The length of the information for the HTTP URI root entry.

Note: If Offset to HTTP URI root equals 0 (QZHT_NONE), -1 (QZHT_DEFAULT), or -2 (QZHT_NO_CHANGE), this value must equal 0.

Offset to HTTP URI root

The offset from the beginning of the request variable to the HTTP URI root data, in bytes. The value must be greater than 0, or equal to one of the special values defined below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
-1	QQZHT_DEFAULT: The default path is used for HTTP URI root.
-2	QZHT_NO_CHANGE: If the current description type is *HTTP1 or *HTTP2, the current path for HTTP URI root is not changed. If the current description type is not *HTTP1 or *HTTP2, see details for HTTP URI root on how current description values are mapped for reuse by the new description type.

Timeout

The number of seconds the triggered cache manager server waits before canceling a read operation on sockets and ending transactions, with the HTTP host, in error. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below. The default value is 0 (QZHT_DISABLED).

Special values and their meanings are as follows:

0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The time out operation is disabled. The triggered cache manager server should never cancel a read operation.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: If the current description type is *HTTP1 or *HTTP2, the current value is not changed. If the current description type is not *HTTP1 or *HTTP2, the default value is used.

CTDP0500 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Char(32)	Cache target description name
64	40	Char(10)	Cache target type
74	4A	Char(2)	Reserved
76	4C	Binary(4)	Default property
80	50	Binary(4)	Number of threads
84	54	Binary(4)	Initial state
88	58	Char(256)	ECCP IP interface
344	158	Binary(4)	ECCP TCP port
348	15C	Char(32)	HTTP cluster IP interface
604	260	Binary(4)	HTTP cluster TCP port
608	264	Binary(4)	Offset to HTTP cluster URI root
612	268	Binary(4)	Length of HTTP cluster URI root
616	26C	Binary(4)	ECCP keepalive
		Char(*)	HTTP cluster URI root

CTDP0500 format field descriptions:

Note: If the current description type is not *ECCP1, all information currently stored, for the description that is not mapped to one the following entries, is discarded.

Cache target description name

The name used by the cache target description (left justified and padded with blanks if necessary). The value must be a description name, or one of the special values described below.

Special values and their meanings are as follows:

***SAME**

QZHT_NO_CHANGE_CHAR: The current name is not changed.

Note: Cache target description names must be unique for each triggered cache manager server. If the name is changed, while the description is referenced by trigger handler description, they too are changed to reference the new name.

Cache target type

The changed cache target descriptions type (left justified and padded with blanks if necessary). The value must be one of the special values described below.

Special values and their meanings are as follows:

***SAME**

QZHT_NO_CHANGE_CHAR: The description is not changed to a different type.

***ECCP1**

QZHT_ECCP_TYPE1: The description is changed to an *ECCP1 type cache target description.

Note: If a type is specified, that is different than the current type, certain information is discarded if it cannot be mapped to values for reuse by the new description type.

Default property

Specifies if the description is a default cache target description for the triggered cache manager server specified in Server name. The value must equal one of the special values described below. The default value is 0 (QZHT_NO).

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The description is a default cache target description for this server.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The description is not a default cache target description.
-1	QZHT_DEFAULT: The default value is specified.
-2	QZHT_NO_CHANGE: The current value is not changed.

Note: Multiple default cache target descriptions are allowed. Trigger handler descriptions added or changed, using the *DEFAULT special value, reference all cache target descriptions designated as default at the time they are added or changed.

ECCP IP interface

The IP host name or address of the backend IP interface to the network router hosting a web server cluster cache target (left justified and padded with blanks if necessary). The value must be a host name (for example, server.mycompany.com) or dotted address (for example, 192.168.3.57). If a host name is specified, it must use proper naming conventions as defined by RFC 1034, Domain Names - Concepts and Facilities. If a dotted address is specified, it must use proper IP version 4 address conventions as defined by RFC 791, Internet Protocol. There is no default value for this entry.

Special values and their meanings are as follows:

***SAME**

QZHT_NO_CHANGE_CHAR: The following rules are used to map current description values for reuse:

If the current type is:		Details
*ECCP1	use current ECCP IP interface value	No change is made.

If the current type is:		Details
*ECCP2	use IP interface value from the host description currently referenced by ECCP host	An exception occurs if the referenced description does not exist. Refers to CTDP0510 format.
Note: An exception occurs if the current type is not one of the types listed above and *SAME is specified for ECCP IP interface.		

ECCP keepalive

Specifies if the connection to the network router is kept open for reuse after data is transferred. The value must equal one of the special values described below. The default value is 0 (QZHT_DISABLED).

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The connection is kept open after data transfer.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The connection is closed after data transfer.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The following rules are used to map current description values for reuse:

If the current type is:		Details
*HTTP1 or *HTTP2	use current HTTP keepalive value	Refers to CTDP0300 or CTDP0310 format.
*ECCP1 or *ECCP2	use current ECCP keepalive value	No change is made.
any other	use default value	

Note: The network router must support keepalive for this option to function properly.

ECCP TCP port

The TCP port number upon which the network router listens for incoming requests. The value must be greater than 0 and less than 65536, or equal to one of the special values described below. There is no default value for this entry.

Special values and their meanings are as follows:

-2	QZHT_NO_CHANGE: If the current description type is *ECCP1 or *ECCP2, the current value is not changed.
Note: If the current description type is not *ECCP1 or *ECCP2, an escape message is sent if -2 (QZHT_NO_CHANGE) is specified.	

Note: The TCP port number is used in combination with the IP host name or address, in ECCP IP interface, to establish communications with the network router cache target.

HTTP cluster IP interface

The IP host name or address of the web server cluster from which the network router is hosting a cache (left justified and padded with blanks if necessary). The value must be a host name (for example, server.mycompany.com), dotted address (for example, 192.168.3.57), or one of the special values described below. If a host name is specified, it must use proper naming conventions as defined by RFC 1034, Domain Names - Concepts and Facilities. If a dotted address is specified, it must use proper IP version 4 address conventions as defined by RFC 791, Internet Protocol. There is no default value for this entry.

Special values and their meanings are as follows:

***DEFAULT**

QZHT_DEFAULT_CHAR: The default value is used.

***SAME**

QZHT_NO_CHANGE_CHAR: The following rules are used to map current description values for reuse:

If the current type is:		Details
*HTTP1	use current HTTP IP interface value	Refers to CTDP0300 format.
*HTTP2	use IP interface value from the host description currently referenced by HTTP host .	Refers to the CTDP0310 format
*ECCP1	use current ECCP IP interface value	No change is made.
*ECCP2	use IP interface value from the host description currently referenced by ECCP host	An exception occurs if the referenced description does not exist. Refers to CTDP0510 format.
any other	use default value	
Note: An exception occurs if the current type is not one of the types listed above and *SAME is specified for HTTP cluster IP interface.		

HTTP cluster TCP port

The TCP port number associated with the IP host name or address described by HTTP cluster IP interface. The value must be greater than 0 and less than 65536, or one of the special values described below. There is no default value for this entry.

Special values and their meanings are as follows:

-2	QZHT_NO_CHANGE: The following rules are used to map current description values for reuse:
----	---

If the current type is:		Details
*HTTP1 or *HTTP2	use current HTTP TCP port value	Refers to CTDP0300 or CTDP0310 format.
*ECCP1 or *ECCP2	use current ECCP TCP port value	No change is made.
any other	use default value	
Note: An exception occurs if the current type is not one of the types listed above and -2 (QZHT_NO_CHANGE) is specified for HTTP cluster IP interface.		

Note: The network router using the address, described by ECCP IP interface, may host multiple web server caches. The specified TCP port number is used in combination with the IP interface in HTTP cluster IP interface to identify a particular web server cache when communicating with the network router.

HTTP cluster URI root

The web server cluster URI path that is used to define the root of this cache target (left justified and padded with blanks if necessary). The value must be a path acceptable to the web server cluster from which the network router is hosting a cache. The default path is /.

If Offset to HTTP cluster URI root equals -2 (QZHT_NO_CHANGE), the following rules are used to map current description values for reuse:

If the current type is:		Details
*IFS	use current Local directory root path	Refers to CTDPO200 format.
*HTTP1 or *HTTP2	use current HTTP URI root path	Refers to CTDPO300 or CTDPO310 format.
*ECCP1 or *ECCP2	use current HTTP cluster URI root path	No change is made.
any other	use default path	

Note: All file requests sent to the cache target have this path inserted into the URI, even if an absolute file path is specified. If the value is null, Offset to HTTP cluster URI root must equal 0 (QZHT_NONE), -1 (QZHT_DEFAULT), or -2 (QZHT_NO_CHANGE). See Offset to HTTP URI root for more details.

Initial state

Specifies the state that the triggered cache manager server request processor, for this cache target, is in at server startup. The value must equal one of the special values described below. The default value is 1 (QZHT_ENABLED).

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The request processor is enabled at server startup.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The request processor is disabled at server startup.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The current value is not changed.

Note: The state of a cache target request processor can be changed while the triggered cache manager server is active by using the *-chsink* command in a trigger message.

Length of HTTP cluster URI root

The length of the information for the HTTP cluster URI root entry.

Note: If Offset to HTTP cluster URI root equals 0 (QZHT_NONE), or -1 (QZHT_DEFAULT), or -2 (QZHT_NO_CHANGE), this value must equal 0.

Number of threads

The number of concurrent threads the triggered cache manager server spawns when processing requests sent to this cache target. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below. The default value is 5.

Special values and their meanings are as follows:

-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The current value is not changed.

Offset to HTTP URI root

The offset from the beginning of the request variable to the HTTP cluster URI root data, in bytes. The value must be greater than 0, or equal to one of the special values defined below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
-1	QZHT_DEFAULT: The default path is used for HTTP cluster URI root.

-2	QZHT_NO_CHANGE: If the current description type is *ECCP1 or *ECCP2, the current path for HTTP cluster URI root is not changed. If the current description type is not *ECCP1 or *ECCP2, see details for HTTP cluster URI root on how current description values are mapped for reuse by the new description type.
----	--

Server name

The name used to identify the triggered cache manager server for which the new description is associated (left justified and padded with blanks if necessary). The value must specify a server name, or one of the special values described below. If a server name is specified, that is different than the one with which the description is currently associated, it is removed from its current association and added for the new server.

Special values and their meanings are as follows:

*SAME

QZHT_NO_CHANGE_CHAR: The current server association is not changed.

CTDP0510 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Char(32)	Cache target description name
64	40	Char(10)	Cache target type
74	4A	Char(2)	Reserved
76	4C	Binary(4)	Default property
80	50	Binary(4)	Number of threads
84	54	Binary(4)	Initial state
88	58	Char(32)	ECCP host
120	78	Binary(4)	ECCP TCP port
124	7C	Char(256)	HTTP cluster IP interface
380	17C	Binary(4)	HTTP cluster TCP port
384	180	Binary(4)	Offset to HTTP cluster URI root
388	184	Binary(4)	Length of HTTP cluster URI root
392	188	Binary(4)	ECCP keepalive
		Char(*)	HTTP cluster URI root

CTDP0510 format field descriptions:

Note: If the current description type is not *ECCP2, all information currently stored for the description, that is not mapped to one the following entries, is discarded.

Cache target description name

The name used by the cache target description (left justified and padded with blanks if necessary). The value must be a description name, or one of the special values described below.

Special values and their meanings are as follows:

***SAME**

QZHT_NO_CHANGE_CHAR: The current name is not changed.

Note: Cache target description names must be unique for each triggered cache manager server. If the name is changed, while the description is referenced by trigger handler description, they too are changed to reference the new name.

Cache target type

The changed cache target descriptions type (left justified and padded with blanks if necessary). The value must be one of the special values described below.

Special values and their meanings are as follows:

***SAME**

QZHT_NO_CHANGE_CHAR: The description is not changed to a different type.

***ECCP2**

QZHT_ECCP_TYPE2: The description is changed to an *ECCP2 type cache target description.

Note: If a type is specified, that is different than the current type, certain information is discarded if it cannot be mapped to values for reuse by the new description type.

Default property

Specifies if the description is a default cache target description for the triggered cache manager server specified in Server name. The value must equal one of the special values described below. The default value is 0 (QZHT_NO).

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The description is a default cache target description for this server.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The description is not a default cache target description.
-1	QZHT_DEFAULT: The default value is specified.
-2	QZHT_NO_CHANGE: The current value is not changed.

Note: Multiple default cache target descriptions are allowed. Trigger handler descriptions added or changed, using the *DEFAULT special value, reference all cache target descriptions designated as default at the time they are added or changed.

ECCP host

The name of a host description associated with the triggered cache manager server referenced by the cache target description and used later, at server startup, to obtain information about the network router that hosts a web server cluster cache target. The value must be a host description name, or one of the special values described below (left justified and padded with blanks if necessary). The default setting references the description currently designated as the default host description for the triggered cache manager server.

Special values and their meanings are as follows:

***DEFAULT**

QZHT_DEFAULT_CHAR: The default value is referenced.

***SAME**

QZHT_NO_CHANGE_CHAR: The following rules are used to map current description values for reuse:

If the current type is:		Details
*HTTP2	the description currently referenced for HTTP host is referenced	An exception occurs if the referenced description does not exist. Refers to CTDP0510 format.
*ECCP2	the description currently referenced for ECCP host is not changed	No change is made.
any other	the default host description currently associated with the triggered cache manager server is referenced	

Note: See ECCP TCP port for more information. An escape message is sent if the referenced description does not currently exist, or if *DEFAULT is specified and there is currently no default host description for the server.

ECCP keepalive

Specifies if the connection to the network router is kept open for reuse after data is transferred. The value must equal one of the special values described below. The default value is 0 (QZHT_DISABLED).

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The connection is kept open after data transfer.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The connection is closed after data transfer.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The following rules are used to map current description values for reuse:

If the current type is:		Details
*HTTP1 or *HTTP2	use current HTTP keepalive value	Refers to CTDP0300 or CTDP0310 format.
*ECCP1 or *ECCP2	use current ECCP keepalive value	No change is made.
any other	use default value	

Note: The network router must support keepalive for this option to function properly.

ECCP TCP port

The TCP port number upon which the network router listens for incoming requests. The value must be greater than 0 and less than 65536, or equal to one of the special values described below. There is no default value for this entry.

Special values and their meanings are as follows:

-2	QZHT_NO_CHANGE: If the current description type is *ECCP1 or *ECCP2, the current value is not changed.
Note: If the current description type is not *ECCP1 or *ECCP2, an escape message is sent if -2 (QZHT_NO_CHANGE) is specified.	

Note: The TCP port number is used in combination with the IP host name or address, in ECCP host, to establish communications with the network router cache target.

HTTP cluster IP interface

The IP host name or address of the web server cluster from which the network router is hosting a cache (left justified and padded with blanks if necessary). The value must be a host name (for example, server.mycompany.com), dotted address (for example, 192.168.3.57), or one of the special values described below. If a host name is specified, it must use proper naming conventions as defined by RFC 1034, Domain Names - Concepts and Facilities. If a dotted address is specified, it must use proper IP version 4 address conventions as defined by RFC 791, Internet Protocol. There is no default value for this entry.

Special values and their meanings are as follows:

*SAME

QZHT_NO_CHANGE_CHAR: The following rules are used to map current description values for reuse:

If the current type is:		Details
*HTTP1	use current HTTP IP interface value	Refers to CTDP0300 format.
*HTTP2	use IP interface value from the host description currently referenced by HTTP host	Refers to the CTDP0310 format.
*ECCP1	use current ECCP IP interface value	Refers to CTDP0500 format.
*ECCP2	use IP interface value from the host description currently referenced by ECCP host	No change is made.
Note: An exception occurs if the current type is not one of the types listed above and *SAME is specified for HTTP cluster IP interface.		

Note: See HTTP cluster TCP port for more information.

HTTP cluster TCP port

The TCP port number associated with the IP host name or address described by HTTP cluster IP interface. The value must be greater than 0 and less than 65536, or one of the special values described below. There is no default value for this entry.

Special values and their meanings are as follows:

-2	QZHT_NO_CHANGE: The following rules are used to map current description values for reuse:
----	---

If the current type is:		Details
*HTTP1 or *HTTP2	use current HTTP TCP port value	Refers to CTDP0300 or CTDP0310 format.
*ECCP1 or *ECCP2	use current ECCP TCP port value	No change is made.
Note: An exception occurs if the current type is not one of the types listed above and -2 (QZHT_NO_CHANGE) is specified for HTTP cluster IP interface.		

Note: The network router using the address, described by ECCP IP interface, may host multiple web server caches. The specified TCP port number is used in combination with the IP interface in HTTP cluster IP interface to identify a particular web server cache when communicating with the network router.

HTTP cluster URI root

The web server cluster URI path that is used to define the root of this cache target (left justified

and padded with blanks if necessary). The value must be a path acceptable to the web server cluster from which the network router is hosting a cache. The default path is /.

If Offset to HTTP cluster URI root equals -2 (QZHT_NO_CHANGE), the following rules are used to map current description values for reuse:

If the current type is:		Details
*IFS	use current Local directory root path	Refers to CTDP0200 format.
*HTTP1 or *HTTP2	use current HTTP URI root path	Refers to CTDP0300 or CTDP0310 format.
*ECCP1 or *ECCP2	use current HTTP cluster URI root path	No change is made.
any other	use default path	

Note: All file requests sent to the cache target have this path inserted into the URI, even if an absolute file path is specified. If the value is null, Offset to HTTP cluster URI root must equal 0 (QZHT_NONE), -1 (QZHT_DEFAULT), or -2 (QZHT_NO_CHANGE). See Offset to HTTP URI root for more details.

Initial state

Specifies the state that the triggered cache manager server request processor, for this cache target, is in at server startup. The value must equal one of the special values described below. The default value is 1 (QZHT_ENABLED).

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The request processor is enabled at server startup.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The request processor is disabled at server startup.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The current value is not changed.

Note: The state of a cache target request processor can be changed while the triggered cache manager server is active by using the *-chsink* command in a trigger message.

Length of HTTP cluster URI root

The length of the information for the HTTP cluster URI root entry.

Note: If Offset to HTTP cluster URI root equals 0 (QZHT_NONE), or -1 (QZHT_DEFAULT), or -2 (QZHT_NO_CHANGE), this value must equal 0.

Number of threads

The number of concurrent threads the triggered cache manager server spawns when processing requests sent to this cache target. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below. The default value is 5.

Special values and their meanings are as follows:

-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The current value is not changed.

Offset to HTTP URI root

The offset from the beginning of the request variable to the HTTP cluster URI root data, in bytes. The value must be greater than 0, or equal to one of the special values defined below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
-1	QZHT_DEFAULT: The default path is used for HTTP cluster URI root.
-2	QZHT_NO_CHANGE: If the current description type is *ECCP1 or *ECCP2, the current path for HTTP cluster URI root is not changed. If the current description type is not *ECCP1 or *ECCP2, see details for HTTP cluster URI root on how current description values are mapped for reuse by the new description type.

Server name

The name used to identify the triggered cache manager server for which the new description is associated (left justified and padded with blanks if necessary). The value must specify a server name, or one of the special values described below. If a server name is specified, that is different than the one with which the description is currently associated, it is removed from its current association and added for the new server.

Special values and their meanings are as follows:

*SAME

QZHT_NO_CHANGE_CHAR: The current server association is not changed.

Note: An escape message is sent if the description is removed from its current association while being referenced by other descriptions.

Error messages:

TCM7001 E

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7031 E

Request variable format is not valid.

TCM7033 E

Length of request variable is not valid for the specified format.

TCM703E E

Error code parameter is not valid.

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 E

A configuration file for triggered cache manager server &1 was not found.

TCM7222 E

A default &1 is not designated for triggered cache manager server &2.

TCM7290 E

&1 &2 was not found for triggered cache manager server &3.

TCM7293 E

A &1 using the name &2 already exists for triggered cache manager server &3.

TCM72C0 E

Triggered cache manager &1 name is not valid.

TCM72C1 E

Triggered cache manager &1 type is not valid.

TCM72C2 E

Triggered cache manager description type &1 cannot be specified when using data format &2.

TCM72C3 E

Value passed to offset &1 is not valid.

TCM74C0 E

Triggered cache manager server name is not valid.

Change Triggered Cache Manager Data Source Description (QzhtChgTCMDataSourceDesc) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 request variable	Input	Char(*)
2 length of request variable	Input	Binary(4)
3 request variable format	Input	Char(8)
4 server name	Input	Char(32)
5 description name	Input	Char(32)
6 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

Use the QzhtChgTCMDataSourceDesc API to change data source descriptions associated with triggered cache manager servers. Changes made to data source descriptions are utilized by all trigger handler descriptions that reference them the next time the servers are started. The API is a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

request variable

INPUT: CHAR(*)

The variable used to pass the information used to change a data source description. See Data source description formats for more information.

length of request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for request variable.

request variable format

INPUT: CHAR(8)

The format name of the Request variable data. The following values must be used:

- DSDP0100: Basic information format for a data source description.
- DSDP0200: Detailed information format for an *IFS type data source description.
- DSDP0300: Detailed information format for a *HTTP1 type data source description.

- DSDP0310: Detailed information format for a *HTTP2 type data source description.

See Data source description formats for more information.

server name

The name used to identify the server for which the description is changed (left justified and padded with blanks if necessary).

description name

The name used to identify which data source description is changed (left justified and padded with blanks if necessary).

error code

I/O: CHAR(*)

The structure in which to return error information.

DSDP0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Char(32)	Data source description name
64	40	Char(10)	Data source type
74	4A	Char(2)	Reserved
76	4C	Binary(4)	Default property
80	50	Binary(4)	Number of threads

DSDP0100 format field descriptions:

Note: Data source descriptions are added using default values for all unspecified values according to the type specified by Data source type). See other data source description formats for details regarding these default values.

Data source description name

The name used by the new data source description (left justified and padded with blanks if necessary).

Special values and their meanings are as follows:

***SAME**

QZHT_NO_CHANGE_CHAR: The current type is not changed.

Note: Data source description names must be unique for each triggered cache manager server. They are referenced, by name, from trigger handler descriptions associated with the same server.

Data source type

The type of data source description that is added (left justified and padded with blanks if necessary). The value must be one of the special values described below.

Special values and their meanings are as follows:

***SAME**

QZHT_NO_CHANGE_CHAR: The description is not changed to a different type.

***IFS** QZHT_IFS_TYPE: The description is changed to an *IFS type data source description.

***HTTP1**

QZHT_HTTP_TYPE1: The description is changed to an *HTTP1 type data source description.

***HTTP2**

QZHT_HTTP_TYPE2: The description is changed to an *HTTP2 type data source description.

Note: If a specified type is different than the current type, certain information is discarded if it cannot be mapped to values for reuse by the new description type. See other data source description formats for details.

Default property

Specifies if the new description will become the default data source description for the triggered cache manager server specified in Server name. The value must equal one of the special values described below. If the value equals 1 (QZHT_YES), the default property on the current default data source description is set to 0 (QZHT_NO). This description takes over the default role. The default value is 0 (QZHT_NO).

Special values and their meanings are as follows:

1	If a specified type is different than the current type, certain information is discarded if it cannot be mapped to values for reuse by the new description type. See other data source description formats for details.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The current value is not changed.

Note: Trigger handler descriptions added or changed using the *DEFAULT special value are referenced from this data source description designated as default at the time the trigger handler descriptions are added or changed.

Number of threads

The number of concurrent threads the triggered cache manager server spawns when processing triggers sent to this trigger handler. The value must be greater than 0 and less than 2³¹ (or 2.147x10⁹), or equal to one of the special values described below. The default value is 5.

Special values and their meanings are as follows:

-1	QQZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The current value is not changed.

Server name

The name used to identify the triggered cache manager server to which the description is associated (left justified and padded with blanks if necessary). The value must be a server name, or one of the special values described below. If a server name is specified that is different than the one with which the description is currently associated, it is removed from its current association and added for the new server.

Special values and their meanings are as follows:

***SAME**

QZHT_NO_CHANGE_CHAR: The current server association is not changed.

Note: An escape message is sent if the description is removed from its current association while being referenced by other descriptions.

DSDP0200 format:

Offset		Type	Field
Dec	Hex		
0	0		Everything from DSDP0100 format
84	54	Binary(4)	Offset to local directory root
88	58	Binary(4)	Length of local directory root
		Char(*)	Local root directory

DSDP0200 format field descriptions:

Note: If the current description type is not *IFS, all information currently stored for the description, that is not mapped for one the following entries, is discarded.

Length of local directory root

The length of the information for the Local directory root entry.

Note: If Offset to local directory root equals 0 (QZHT_NONE) or -1 (QZHT_DEFAULT), this value must equal 0.

Local directory root

The path to the local file system directory that is the root of this data source (left justified and padded with blanks if necessary). The value must be a path acceptable to the root (/) file system of the local iSeries system. If a path is provided, but does not specify a absolute path (does not start with /), the path is prepended with the default path. The default path is /QIBM/UserData/TCM/{Server name} where {Server name} is the name of the triggered cache manager server as defined by Server name.

If the current type is:		Details
*IFS	use current Local directory root path	No change is made
*HTTP1 or *HTTP2	use current HTTP URI root path	Refers to DSDP0300 or DSDP0310 format
any other	use default path	

Note: All requests for files from the data source must have this path prepended to the file name, even if an absolute path is specified for the file. If the value is null, Offset to local directory root must equal 0 (QZHT_NONE) or -1 (QZHT_DEFAULT).

Offset to local directory root

The offset from the beginning of the request variable to the Local directory root data, in bytes. The value must be greater than 0, or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
-1	QZHT_DEFAULT: The default path is used for Local directory root.
-2	QZHT_NO_CHANGE: If the current description type is *IFS, the current path for Local directory root is not changed. If the current description type is not *IFS, see details for Local directory root on how current description values are mapped for reuse by the new description type.

DSDP0300 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(256)	Everything from DSDP0100 format
88	58	Binary(4)	HTTP IP interface
340	154	Binary(4)	HTTP TCP port
344	158	Binary(4)	Offset to HTTP URI root
348	15C	Binary(4)	Length of HTTP URI root
352	160	Binary(4)	HTTP keepalive
356	164	Binary(4)	Timeout
		Char(*)	HTTP URI root

DSDP0300 format field descriptions:

Note: If the current description type is not *HTTP1, all information currently stored for the description, that is not mapped for one the following entries, is discarded.

HTTP IP interface

The IP host name or address of the system hosting an HTTP server data source (left justified and padded with blanks if necessary). The value must be a host name (for example, server.mycompany.com), dotted address (for example, 192.168.3.57), or one of the special values described below. If a host name is specified, it must use proper naming conventions as defined by RFC 1034, Domain Names - Concepts and Facilities. If a dotted address is specified, it must use proper IP version 4 address conventions as defined by RFC 791, Internet Protocol. The default value is 127.0.0.1, the local system loopback interface.

Special values and their meanings are as follows:

*DEFAULT

QZHT_DEFAULT_CHAR: The default value is used.

*SAME

QZHT_NO_CHANGE_CHAR: The following rules are used to map current description values for reuse.

If the current type is:		Details
*HTTP1	use current HTTP IP interface value	No change is made.
*HTTP2	use IP interface value from the host description currently referenced by HTTP host An exception occurs if the referenced description does not exist. Refers to the	An exception occurs if the referenced description does not exist. Refers to the DSDP0310 format.
any other	use default value	

Note: See HTTP TCP port for more information.

HTTP keepalive

Specifies if connections to HTTP Server are kept open for reuse after data is transferred. The value must equal one of the special values described below. The default value is 0 (QZHT_DISABLED).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The connection is kept open after data transfer.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The connection is closed after data transfer.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: If the current description type is *HTTP1 or *HTTP2, the current value is not changed. If the current description type is not *HTTP1 or *HTTP2, the default value is used.

Note: HTTP Server must support keepalive for this option to work properly.

HTTP TCP port

The TCP port number upon which HTTP Server listens for incoming requests. The value must be greater than 0 and less than 65536, or equal to one of the special values described below. The default value is 80.

Special values and their meanings are as follows:

-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The following rules are used to map current description values for reuse.

If the current type is:		Details
*HTTP1 or *HTTP2	use current HTTP TCP port value	No change is made
any other	use default value	

Note: The TCP port number is used in combination with the IP host name or address in HTTP IP interface to establish communication with HTTP Server data source.

HTTP URI root

The path of HTTP Server URI that is the root of this data source (left justified and padded with blanks if necessary). The value must be a path acceptable to HTTP Server. The default path is / .

If Offset to HTTP URI root equals -2 (QZHT_NO_CHANGE), the following rules are used to map current description values for reuse:

If the current type is:		Details
*IFS	use current Local directory root path	Refers to DSDP0200 format.
*HTTP1 or *HTTP2	use current HTTP URI root path	No change is made
any other	use default path	

Note: All file requests from this data source have this path inserted into the URI, even if an absolute path for the file is specified. If the value is null, Offset to HTTP URI root must equal 0 (QZHT_NONE), or -1 (QZHT_DEFAULT).

Length of HTTP URI root

The length of the information for the HTTP URI root entry.

Note: If Offset to HTTP URI root equals 0 (QZHT_NONE), or -1 (QZHT_DEFAULT), this value must equal 0.

Offset to HTTP URI root

The offset from the beginning of the request variable to the HTTP URI root data, in bytes. The value must be greater than 0, or equal to one of the special values defined below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
-1	QZHT_DEFAULT: The default value is used for HTTP URI root.
-2	QZHT_NO_CHANGE: If the current description type is *HTTP1 or *HTTP2, the current path for HTTP URI root is not changed. If the current description type is not *HTTP1 or *HTTP2, see details for HTTP URI root on how current description values are mapped for reuse by the new description type.

Timeout

The number of seconds the triggered cache manager server waits before canceling a read operation on sockets and ending transactions, with the HTTP host, in error. The value must be greater than 0 and less than 2^{31} (or 2.147x109), or equal to one of the special values described below. The default value is 0 (QZHT_DISABLED).

Special values and their meanings are as follows:

0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The time out operation is disabled. The triggered cache manager server should never cancel a read operation.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: If the current description type is *HTTP1 or *HTTP2, the current value is not changed. If the current description type is not *HTTP1 or *HTTP2, the default value is used.

DSDP0310 format:

Offset		Type	Field
Dec	Hex		
0	0		Everything from DSDP0100 format
84	54	Char(32)	HTTP host
116	74	Binary(4)	HTTP TCP port
120	78	Binary(4)	Offset to HTTP URI root
124	7C	Binary(4)	Length of HTTP URI root
128	80	Binary(4)	HTTP keepalive
132	84	Binary(4)	Timeout
		Char(*)	HTTP URI root

DSDP0310 format field descriptions:

Note: If the current description type is not *HTTP2, all information currently stored for the description, that is not mapped to one the following entries, is discarded.

HTTP host

The name of a host description associated with the triggered cache manager server that is referenced by the new data source description and used later, at server startup, to obtain information about the system hosting an HTTP server data source. The value must be a host description name, or one of the special values described below (left justified and padded with blanks if necessary). The default value is to reference the description currently designated as the default host description for the triggered cache manager server.

Special values and their meanings are as follows:

*DEFAULT

QZHT_DEFAULT_CHAR: The default value is referenced.

*SAME

QZHT_NO_CHANGE_CHAR: The following rules are used to map current description values for reuse.

If the current type is:		Details
*HTTP2	the description currently referenced for HTTP host is not changed	No change is made.
any other	the default host description currently associated with the triggered cache manager server is referenced	

Note: See HTTP TCP port for more information. An escape message is sent if the referenced description does not currently exist, or if *DEFAULT is specified and there is currently no default host description for the server.

HTTP keepalive

Specifies if connections to HTTP Server are kept open for reuse after data is transferred. The value must equal one of the special values described below. The default value is 0 (QZHT_DISABLED).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The connection is kept open after data transfer.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The connection is closed after data transfer.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: If the current description type is *HTTP1 or *HTTP2, the current value is not changed. If the current description type is not *HTTP1 or *HTTP2, the default value is used.

Note: HTTP Server must support keepalive for this option to work properly.

HTTP TCP port

The TCP port number upon which HTTP Server listens for incoming requests. The value must be greater than 0 and less than 65536, or equal to one of the special values described below. The default value is 80.

Special values and their meanings are as follows:

-1	QZHT_DEFAULT: The default value is used.
----	--

-2	QZHT_NO_CHANGE: The following rules are used to map current description values for reuse.
----	---

If the current type is:		Details
*HTTP1 or *HTTP2	use current HTTP TCP port value	No change is made.
any other	use default value	

Note: The TCP port number is used in combination with information obtained at server startup from the host description specified in HTTP host to establish communication with HTTP Server data source.

HTTP URI root

The path of HTTP Server URI that is the root of this data source (left justified and padded with blanks if necessary). The value must be a path acceptable to HTTP Server. The default path is / .

If Offset to HTTP URI root equals -2 (QZHT_NO_CHANGE), the following rules are used to map current description values for reuse:

If the current type is:		Details
*IFS	use current Local directory root path	Refers to DSDP0200 format
*HTTP1 or *HTTP2	use current HTTP URI root path	No change is made
any other	use default path	

Note: All requests for files from this data source have this path inserted into the URI, even if an absolute file path is specified. If the value is null, Offset to HTTP URI root must equal 0 (QZHT_NONE), -1 (QZHT_DEFAULT), or -2 (QZHT_NO_CHANGE). See Offset to HTTP URI root for more details.

Length of HTTP URI root

The length of the information for the HTTP URI root entry.

Note: If Offset to HTTP URI root equals 0 (QZHT_NONE), or -1 (QZHT_DEFAULT), this value must equal 0.

Offset to HTTP URI root

The offset from the beginning of the request variable to the HTTP URI root data, in bytes. The value must be greater than 0, or equal to one of the special values defined below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
-1	QZHT_DEFAULT: The default path is used for HTTP URI root.
-2	QZHT_NO_CHANGE: If the current description type is *HTTP1 or *HTTP2, the current path for HTTP URI root is not changed. If the current description type is not *HTTP1 or *HTTP2, see details for HTTP URI root on how current description values are mapped for reuse by the new description type.

Timeout

The number of seconds the triggered cache manager server waits before canceling a read operation on sockets and ending transactions, with the HTTP host, in error. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below. The default value is 0 (QZHT_DISABLED).

Special values and their meanings are as follows:

0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The time out operation is disabled. The triggered cache manager server should never cancel a read operation.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: If the current description type is *HTTP1 or *HTTP2, the current value is not changed. If the current description type is not *HTTP1 or *HTTP2, the default value is used.

Error messages:

TCM7001 E

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7031 E

Request variable format is not valid.

TCM7033 E

Length of request variable is not valid for the specified format.

TCM703E E

Error code parameter is not valid.

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 E

A configuration file for triggered cache manager server &1 was not found.

TCM7222 E

A default &1 is not designated for triggered cache manager server &2.

TCM7290 E

&1 &2 was not found for triggered cache manager server &3.

TCM7293 E

A &1 using the name &2 already exists for triggered cache manager server &3.

TCM72C0 E

Triggered cache manager &1 name is not valid.

TCM72C1 E

Triggered cache manager &1 type is not valid.

TCM72C2 E

Triggered cache manager description type &1 cannot be specified when using data format &2.

TCM72C3 E

Value passed to offset &1 is not valid.

TCM74C0 E

Triggered cache manager server name is not valid.

Change Triggered Cache Manager Host Description (QzhtChgTCMHostDesc) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP

Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 request variable	Input	Char(*)
2 length of request variable	Input	Binary(4)
3 request variable format	Input	Char(8)
4 server name	Input	Char(32)
5 description name	Input	Char(32)
6 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: yes

The QzhtChgTCMHostDesc API changes host descriptions associated with triggered cache manager servers. Changes made to host descriptions are utilized by all data source, cache target, and acknowledgment target descriptions referencing them the next time the servers are started.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

request variable

INPUT: CHAR(*)

The variable used to pass information used to change a host description.

length of request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for request variable.

request variable format

INPUT: CHAR(8)

The format name of request variable data. The following values must be used:

- HSDP0100: Basic information format for a host description.

server name

INPUT: CHAR(32)

The name used to identify the server for which the description is the be changed (left justified and padded with blanks if necessary).

description name

INPUT: CHAR(32)

The name used to identify which host description is the be changed (left justified and padded with blanks if necessary).

error code

I/O: CHAR(*)

The structure in which to return error information.

HSDP0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Char(32)	Host description
64	20	Binary(4)	Default property
68	44	Binary(4)	IP interface

Field descriptions:

Default property

Specifies whether the description is to become the default host description for the triggered cache manager server specified in Server name. The value must equal one of the special values described below. If the value equals 1 (QZHT_YES) and the description is not currently the default description, the default property on the current default host description, if any, are set to 0 (QZHT_NO) and this description takes over the default role. The default value is -2 (QZHT_NO_CHANGE).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The description is to become the default host description for this server.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The current value is not changed.

Host description name

The name used by the host description (left justified and padded with blanks if necessary). The value must be a description name, or one of the special values described below.

Special values and their meanings are as follows:

*SAME

QZHT_NO_CHANGE: The current name is not changed.

Note: Host description name must be unique for each triggered cache manager server. If the name is changed while the description is referenced by data source, cache target, or acknowledgement target descriptions, they too are changed to reference the new name.

IP interface

The IP host name or address of the computer system hosting servers used by the triggered cache manager server (left justified and padded with blanks if necessary). The value must specify a host name (i.e.: server.mycompany.com), dotted address (i.e.: 192.168.3.57), or one of the special values described below. If a host name is specified, it must use proper naming conventions as defined by RFC 1034, Domain Names – Concepts and Facilities. If a dotted address is specified, it must use proper IP version 4 address conventions as defined by RFC 791, Internet Protocol. The default value is 127.0.0.1, the local systems loopback interface.

Special values and their meanings are as follows:

*DEFAULT

QZHT_DEFAULT_CHAR: The default value is used.

*SAME

QZHT_NO_CHANGE_CHAR: The current value is not changed.

Server name

The name used to identify the triggered cache manager server with which the description is associated (left justified and padded with blanks if necessary). The value must be a server name, or one of the special values described below. If a server name is specified which is different than the one with which the description is currently associated, it is removed from its current association and added for the new server.

Special values and their meanings are as follows:

*SAME

QZHT_NO_CHANGE_CHAR: The current server association is not changed.

Note: An escape message is sent if the description is removed from its current association while it is referenced by other descriptions.

Error messages:

TCM7001 E

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7031 E

Request variable format is not valid.

TCM7033 E

Length of request variable is not valid for the format specified.

TCM703E E

Error code parameter is not valid.

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager servers &1.

TCM7190 E

A configuration file for triggered cache manager server &1 was not found.

TCM7290 E

&1 &2 was not found for triggered cache manager server &3.

TCM7293 E

A &1 using the name &2 already exists for triggered cache manager server &3.

TCM72C0 E

Triggered cache manager&1 name is not valid.

TCM72C3 E

Value passed at offset &1 is not valid.

TCM74C0 E

Triggered cache manager server name is not valid.

Change Triggered Cache Manager Object Dependency Graph Description (QzhtChgTCMODGDesc) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 request variable	Output	Char(*)
2 length of request variable	Input	Binary(4)
3 request variable format	Input	Char(8)
4 server name	Input	Char(32)
5 description name	Input	Char(32)
6 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

The *QzthgTCMODGDesc* API changes object dependency graph (ODG) descriptions associated with triggered cache manager servers. Changes made to object dependency graph descriptions are utilized by all trigger handler descriptions referencing them the next time the servers are started.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

request variable

INPUT: CHAR(*)

The variable used to pass information used to change an object dependency graph description.

length of request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for request variable.

request variable format

CHAR(8)

The format name of request variable data. The following values must be used:

- OGDPO100: Basic information format for an object dependency graph description.

server name

INPUT: CHAR(32)

The name used to identify the server for which the description is changed (left justified and padded with blanks if necessary).

description name

INPUT: CHAR(32)

The name used to identify which object dependency graph (ODG) description is changed (left justified and padded with blanks if necessary).

error code

I/O: CHAR(*)

The structure in which to return error information.

OGDPO100 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name

Offset		Type	Field
Dec	Hex		
32	20	Char(32)	Object dependency graph description name
64	40	Binary(4)	Default property
68	44	Binary(4)	Allow API updates

OGDP0100 format field descriptions:

Allow API updates

Specifies whether APIs are allowed to update the object dependency graph described by the object dependency description. The value must be one of the special values described below. The default value is 1 (QZHT_YES).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The object dependency graph is updating using APIs via the triggered cache manager server, as well as from trigger handler publish parsing.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The object dependency graph may not be updated using APIs via the triggered cache manager server. Only updates as a result of trigger handler publish parsing are allowed.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The current value is not changed.

Default property

Specifies whether the description is to become the default object dependency graph description for the triggered cache manager server specified in Server name. The value must equal one of the special values described below. If the value equals 1 (QZHT_YES) and the description is not currently the default description, the default property for the current default object dependency graph description, is set to 0 (QZHT_NO) and this description takes over the default role. The default value is -2 (QZHT_NO_CHANGE).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The description is to become the default object dependency graph description for this server.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The current value is not changed.

Note: Trigger handler descriptions, added or changed using the *DEFAULT special value, reference the object dependency graph description designated as default at the time they are added or changed.

Object dependency graph description name

The name used by the object dependency graph description (left justified and padded with blanks if necessary). The value must be a description name, or one of the special values described below.

Special values and their meanings are as follows:

*SAME

QZHT_NO_CHANGE_CHAR: The current name is not changed.

Note: Object dependency graph description names must be unique for each triggered cache manager server. If the name is changed while the description is referenced by trigger handler descriptions, they too are changed to reference the new name.

Server name

The name used to identify the triggered cache manager server with which the description is associated (left justified and padded with blanks if necessary). The value must be a server name, or one of the special values described below. If a server name is specified which is different than the one with which the description is currently associated, it is removed from its current association and added for the new server.

Special values and their meanings are as follows:

***SAME**

QZHT_NO_CHANGE_CHAR: The current server association is not changed.

Note: An escape message is sent if the description is removed from its current association while it is being referenced by other descriptions.

Error messages:

TCM7001 E

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7031 E

Request variable format is not valid.

TCM7033 E

Length of request variable is not valid for the format specified.

TCM703E E

Error code parameter is not valid.

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 E

A configuration file for triggered cache manager server &1 was not found.

TCM7290 E

&1 &2 was not found for triggered cache manager server &3.

TCM7293 E

A &1 using the name &2 already exists for triggered cache manager server &3.

TCM72C0 E

Triggered cache manager &1 name is not valid.

TCM72C3 E

Value passed at offset &1 is not valid.

TCM7401 E

User QTCM is not authorized to server data for triggered cache manager server &1.

TCM74C0 E

Triggered cache manager server name is not valid.

TCM7701 E

User QTCM is not authorized to object dependency graph data for triggered cache manager server &1.

Change Triggered Cache Manager Publishing Rule (QzhtChgTCMPublishingRule) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 request variable	Input	Char(*)
2 length of request variable	Input	Binary(4)
3 request variable format	Input	Char(8)
4 server name	Input	Char(32)
5 description name	Input	Char(32)
6 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

The QzhtChgTCMPublishingRule API changes publishing rules associated with triggered cache manager servers. Changes made to publishing rules are utilized by all Rule Sets that are referencing them the next time that the servers are started. The API is a callable service implemented as an ILE entry point within QZHTINCONF *SRVPGM in QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

request variable

INPUT: CHAR(*)

The variable used to pass information used to change a publishing rule.

length of request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for request variable.

request variable format

INPUT: CHAR(8)

The format name of the request variable data. The following values must be used:

- PRDP0100: Basic information format for a publishing rule.

server name

INPUT: CHAR(32)

The name used to identify the server for which the description is changed (left justified and padded with blanks if necessary).

description name

INPUT: CHAR(32)

The name used to identify the which publishing rule is changed (left justified and padded with blanks if necessary).

error code

I/O: CHAR(*)

The structure in which to return error information.

PRDP0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Char(32)	Publishing rule name
64	40	Binary(4)	Offset to file extensions
68	44	Binary(4)	Length of file extensions
72	48	Binary(4)	Read from data source
76	4C	Binary(4)	Send data source version
80	50	Binary(4)	Parse and assemble
84	54	Binary(4)	Send assembled version
88	58	Binary(4)	Offset to new file extension
72	5C	Binary(4)	Length of new file extension
		Char(*)	File extensions
		Char(*)	New file extension

*PRDP0100 format field descriptions:***File extension**

A list of file extensions used to identify files that are processed according to this new publishing rule. File extensions must be listed as a string of characters, where each extension starts with a period character (.) and is separated by one or more spaces (left justified and padded with blanks if necessary). There is no default value for this entry.

If Offset to file extensions equals -2 (QZHT_NO_CHANGE), the current list of file extensions is not changed.

Note: A list of file extensions is required. The file extensions are used by trigger handlers to determine when the publishing rule applies. File names are compared to file extensions starting at the last period in the file name. Therefore, each file extension must start with a period character (.), otherwise a match is not made. For example, .html, .gif, or .event.

Length of file extensions

The length of the information for the File extensions entry.

Note: If Offset to file extensions equals 0 (QZHT_NONE) or -2 (QZHT_NO_CHANGE), this value must equal 0.

Length of new file extension

The length of the information for the New file extension entry.

Note: If Offset to new file extension equals 0 (QZHT_NONE), or -1 (QZHT_DEFAULT), this value must equal 0.

New file extension

A file extension used to rename files after they have been assembled and before they are sent to the cache targets (left justified and padded with blanks if necessary). The default value is null, indicating that files are not renamed.

Note: If the value is null, Offset to new file extension must equal 0 (QZHT_NONE), -1 (QZHT_DEFAULT), or -2 (QZHT_NO_CHANGE). If a file extension is specified, files matching any one of the file extensions listed in File extension are renamed with the file extension prior to being sent to the cache targets. An escape message is sent if a new file extension is provided and Send assembled version equals 0 (QZHT_NO).

Offset to new file extension

The offset from the beginning of the request variable to the New file extension data, in bytes. The value must be greater than 0 or equal one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
-1	QZHT_DEFAULT: The default value is used for New file extension.
-2	QZHT_NO_CHANGE: The current file extension, if any, is not changed for New file extension.

Note: An escape message is sent if Offset to new file extension is greater than 0 (indicating that a new file extension has been provided) and Send assembled version equals 0 (QZHT_NO).

Offset to file extensions

The offset from the beginning of the request variable to the New file extension data, in bytes. The value must be greater than 0, or equal one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
-1	QZHT_DEFAULT: The default value is used for New file extension.
-2	QZHT_NO_CHANGE: The current file extension, if any, is not changed for New file extension.

Note: An escape message is sent if Offset to new file extension is greater than 0 (indicating a new file extension has been provided) and Send assembled version equals 0 (QZHT_NO).

Parse and assemble

Specifies if files matching this publishing rule are parsed for wrappers and includes, and possibly sent through the page assembler. If the name is changed during wrapper parsing, the extension of the new name is used to determine if the file should be parsed for includes and sent through the page assembler. The value must equal one of the special values described below. The default value is 0 (QZHT_NO).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: Files matching this publishing rule are parsed and possibly sent through the page assembler.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: Files matching this publishing rule are not parsed. Note: Files that are not parsed are not sent through the page assembler.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The current value is not changed.

Note: An escape message is sent if Parse and assemble equals 1 (QZHT_YES) and Read from data source equals 0 (QZHT_NO).

Publishing rule name

An escape message is sent if Parse and assemble equals 1 (QZHT_YES) and Read from data source equals 0 (QZHT_NO).

Special values and their meanings are as follows:

***SAME**

QZHT_NO_CHANGE_CHAR: The current name is not changed.

Note: Publishing rule names must be unique for each triggered cache manager server. They are referenced by name, from Rule Sets associated with the same server.

Read from data source

Specifies if files matching this publishing rule are read from the data source when triggered. The value must be one of the special values described below. The default value is 1 (QZHT_YES).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: Files matching this publishing rule are read from the data source when triggered.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: Files matching this publishing rule are not read from the data source. Note: Not reading files from the data source is useful when triggers specify a file which is purely symbolic and need not correspond to an actual file, yet causes dependent files processing.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The current value is not changed.

Send assembled version

Specifies if the assembled version of files matching this publishing rule are sent to cache targets when triggered and processed. The value must equal one of the special values described below. The default value is 0 (QZHT_NO).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: Assembled versions of files matching this publishing rule are sent to cache targets when triggered and processed.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: Assembled versions of files matching this publishing rule are not sent to cache targets.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The current value is not changed.

Note: An escape message is sent if Send assembled version equals 1 (QZHT_YES) and Parse and assemble equals 0 (QZHT_NO).

Send data source version

Specifies if the data source version of files matching this publishing rule (the version read from the data source prior to assembly) are sent to cache targets when triggered and processed. The value must equal one of the special values described below. The default value is 1 (QZHT_YES) when Read from data source equals 1 (QZHT_YES), and 0 (QZHT_NO) when Read from data source equals 0 (QZHT_NO).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: Data source versions of files matching this publishing rule are sent to cache targets when triggered and processed.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: Data source versions of files matching this publishing rule are not sent to cache targets.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The current value is not changed.

Note: An escape message is sent if Send data source version equals 1 (QZHT_YES) and Read from data source equals 0 (QZHT_NO).

Server name

The name used to identify the triggered cache manager server to which the description is associated (left justified and padded with blanks if necessary). The value must be a server name, or be one of the special values described below. If a server name is specified which is different than the one that is currently associated, it is removed from its current association and added to the new server.

Special values and their meanings are as follows:

*SAME

QZHT_NO_CHANGE_CHAR: The current server association is not changed.

Note: An escape message is sent if the description is removed from its current association while it is being referenced by other descriptions.

Error messages:

TCM7001 E

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7031 E

Request variable format is not valid.

TCM7033 E

Length of request variable is not valid for the specified format.

TCM703E E

Error code parameter is not valid.

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 E

A configuration file for triggered cache manager server &1 was not found.

TCM7290 E

&1&2 was not found for triggered cache manager server &3.

TCM7293 E

A &1 using the name&2 already exists for triggered cache manager server &3.

TCM72C0 E

Triggered cache manager &1 name is not valid.

TCM72C3 E

Value passed to offset &1 is not valid.

TCM74C0 E

Triggered cache manager server name is not valid.

Change Triggered Cache Manager Rule Set (QzhtChgTCMRuleSet) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 request variable	Input	Char(*)
2 length of request variable	Input	Binary(4)
3 request variable format	Input	Char(8)
4 server name	Input	Char(32)
5 description name	Input	Char(32)
6 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

The QzhtChgTCMRuleSet API changes rule sets associated with triggered cache manager servers. Changes made to rule sets are utilized by all trigger handler descriptions that are referencing them the next time that the servers are started.

Note: Triggers are sent to trigger handlers which process them according to publishing rules. Custom publishing rules are provided for the trigger handler through a rule set. If extensions of files identified in triggers match one of the extensions listed in a custom publishing rule, they are processed according to that publishing rule. If extensions do not match any of the custom publishing rules, the file is processed according to the default publishing rule.

The API is a callable service implemented as an ILE entry point within QZHTINCONF *SRVPGM in QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

request variable

INPUT: CHAR(*)

The variable used to pass information used to change a rule set.

length of request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for request variable.

request variable format

INPUT: CHAR(8)

The format name of the request variable data. The following values must be used:

- RSDP0100: Basic information format for a rule set.

server name

The name used to identify the server for which the description is changed (left justified and padded with blanks if necessary).

description name

INPUT: CHAR(32)

The name used to identify the which rule set is changed (left justified and padded with blanks if necessary).

error code

I/O: CHAR(*)

The structure in which to return error information.

RSDP0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Char(32)	Rule set name
64	40	Binary(4)	Default property
68	44	Binary(4)	Offset to publishing rules
72	48	Binary(4)	Length of publishing rules
		Char(*)	Publishing rules

RSDP0100 format field descriptions:

Default property

Specifies if the new description is to become the default rule set for the triggered cache manager server specified in Server name. The value must equal one of the special values described below. If the value equals 1 (QZHT_YES), the default property on the current default rule set, if any, is set to 0 (QZHT_NO). The default value is 0 (QZHT_NO).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The description is to become the default rule set for this server.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The description is not to become the default rule set.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The current value is not changed.

Note: Trigger handler descriptions, added or changed using the *DEFAULT special value, reference the rule set designated as default at the time they are added or changed.

Length of publishing rules

The length of the information for the Publishing rules entry.

Note: If Offset to Publishing rules equals 0 (QZHT_NONE) or -1 (QZHT_DEFAULT), this value must equal 0.

Offset to publishing rules

The offset from the beginning of the request variable to the Publishing rules data, in bytes. The value must be greater than 0, or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
-1	QZHT_DEFAULT: The default value is used for Publishing rules.
-2	QZHT_NO_CHANGE: The current list of publishing rules, if any, is not changed for Publishing rules.

Note: An escape message is sent if Offset to new file extension is greater than 0 (indicating that a new file extension has been provided) and Send assembled version equals 0 (QZHT_NO).

Publishing rules

A list of publishing rules associated with the triggered cache manager server referenced by the new Rule Set and used by trigger handlers, at startup, to direct how files are processed. Descriptions must be listed by name, where each name is separated by one or more spaces, and padded with blanks if necessary. The default value is null, indicating that an empty rule set is described.

If Offset to publishing rules equals 0 (QZHT_NONE) or -1 (QZHT_DEFAULT), an empty rule set is described. If Offset to publishing rules equals -2 (QZHT_NO_CHANGE), the current list of publishing rules are not changed.

Note: If the value is null, Offset to publishing rules must equal 0 (QZHT_NONE) or -1 (QZHT_DEFAULT). See Offset to publishing rules for more details. An empty rule set referenced by a trigger handler causes it to process all triggers according to the default publishing rule. An escape message is sent if any referenced description does not currently exist.

Rule set object name

The name used by the rule set (left justified and padded with blanks if necessary). The value must be a description name, or one of the special values described below.

*SAME

QZHT *SAME

Note: Rule set names must be unique for each triggered cache manager server. They are referenced, by name, from trigger handler descriptions associated with the same server.

Server name

The name used to identify the triggered cache manager server with which the description is associated (left justified and padded with blanks if necessary). The value must be a server name, or one of the special values described below. If a server name is specified which is different than the one with which the description is currently associated, it is removed from its current association and added for the new server.

Special values and their meanings are as follows:

*SAME

QZHT_NO_CHANGE_CHAR: The current server association is not changed.

Note: An escape message is sent if the description is removed from its current association while it is being referenced by other descriptions.

Error messages:

TCM7001 E

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

- TCM7031 E**
Request variable format is not valid.
- TCM7033 E**
Length of request variable is not valid for the specified format.
- TCM703E E**
Error code parameter is not valid.
- TCM70F0 E**
Unknown error occurred while processing request.
- TCM7101 E**
User QTCM is not authorized to the configuration file for triggered cache manager server &1.
- TCM7190 E**
A configuration file for triggered cache manager server &1 was not found.
- TCM7290 E**
&1&2 was not found for triggered cache manager server &3.
- TCM7293 E**
A &1 using the name&2 already exists for triggered cache manager server &3.
- TCM72C0 E**
Triggered cache manager &1 name is not valid.
- TCM72C3 E**
Value passed to offset &1 is not valid.
- TCM74C0 E**
Triggered cache manager server name is not valid.

Change Triggered Cache Manager Trigger Handler Description (QzhtChgTCMTriggerHandlerDesc) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 request variable	Input	Char(*)
2 length of request variable	Input	Binary(4)
3 request variable name	Input	Char(8)
4 server name	Input	Char(32)
5 description name	Input	Char(32)
6 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

Use the QzhtChgTCMTriggerHandlerDesc API to change trigger handler descriptions associated with triggered cache manager servers. Changes made to trigger handler descriptions are utilized by triggered cache manager servers after they are restarted. The API is a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

request variable

INPUT: CHAR(*)

The variable used to pass the information used to add a new trigger handler description.

length of request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for request variable.

request variable format

INPUT: CHAR(8)

The format name of the request variable data. The following values must be used:

- THDP0100: Basic information format for a trigger handler description.
- THDP0200: Detailed information format for an *UPDATE type trigger handler description.
- THDP0300: Detailed information format for a *PUBLISH type trigger handler description.

server name

INPUT: CHAR(32)

The name used to identify the server for which the description is changed (left justified and padded with blanks if necessary).

description name

INPUT: CHAR(32)

The name used to identify which trigger handler description is changed (left justified and padded with blanks if necessary).

error code

I/O: CHAR(*)

The structure in which to return error information.

THDP0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Char(32)	Trigger handler description name
64	40	Char(10)	Trigger handler type
74	4A	Char(2)	Reserved
76	4C	Char(32)	Data source
108	6C	Binary(4)	Offset to cache targets
112	70	Binary(4)	Length of cache targets
116	74	Binary(4)	Offset to ack targets
120	78	Binary(4)	Length of ack targets
124	7C	Binary(4)	Offset to nack targets
128	80	Binary(4)	Length of nack targets
132	84	Binary(4)	Number of threads
		Char(*)	Cache targets
		Char(*)	Ack targets

Offset		Type	Field
Dec	Hex		
		Char(*)	Nack targets

THDP0100 format field descriptions:

Note: If a description type, that is different than the current type, is specified in Trigger handler type), the description is changed using default values for all unspecified values (according to the type specified. See other trigger handler description formats for details regarding these default values.

Ack targets

A list of acknowledgment target descriptions associated with the triggered cache manager server referenced by the trigger handler description and used later, at server startup, to obtain information as to where successful process completion messages are sent. Descriptions must be listed by name, where each name is separated by one or more spaces, and padded with blanks if necessary. The default setting is to reference the descriptions currently designated as default acknowledgment target descriptions for the triggered cache manager server. If there are no descriptions currently designated as default, a null list is used. A null list indicates successful process completion messages are not sent.

Note: If a null list is specified, Offset to ack targets must equal 0 (QZHT_NONE), -1 (QZHT_DEFAULT), or -2 (QZHT_NO_CHANGE). See Offset to ack targets for more details. Messages concerning successful process completion of triggers referencing this handler are sent to all listed acknowledgment targets. An escape message is sent if any referenced description does not currently exist.

Cache targets

A list of cache target descriptions associated with the triggered cache manager server referenced by the trigger handler description and used later, at server startup, to obtain information about the cache targets to which the trigger handler sends data. Descriptions must be listed by name, where each name is separated by one or more spaces, and padded with blanks if necessary. The default setting is to reference the descriptions currently designated as default cache target descriptions for the triggered cache manager server. If there are no descriptions currently designated as default, a null list is used. A null list indicates data is not sent to cache targets.

Note: If a null list is specified, Offset to ack targets must equal 0 (QZHT_NONE), -1 (QZHT_DEFAULT), or -2 (QZHT_NO_CHANGE). See *Offset to ack targets* for more details. Data processed by this trigger handler is sent to all listed cache targets. An escape message is sent if any referenced description does not currently exist.

Data source

The name of a data source description associated with the triggered cache manager server referenced by the trigger handler description and used later, at server startup, to obtain information about the data source from which the trigger handler receives data. The value must be a data source description name, or one of the special values described below (left justified and padded with blanks if necessary). The default setting references the description currently designated as the default data source description for the triggered cache manager server.

Special values and their meanings are as follows:

***DEFAULT**

QZHT_DEFAULT_CHAR: The default value is referenced.

***SAME**

QZHT_NO_CHANGE_CHAR: The current reference is not changed.

Note: A data source name is required. An escape message is sent if the referenced description does not currently exist, or if *DEFAULT is specified and there is currently no default data source description for the server.

Length of ack targets

The length of the information for the Ack targets entry.

Note: If Offset to ack targets equals 0 (QZHT_NONE), -1 (QZHT_DEFAULT), or -2 (QZHT_NO_CHANGE), this value must equal 0.

Length of cache targets

The length of the information for the Cache targets entry.

Note: If Offset to cache targets equals 0 (QZHT_NONE), -1 (QZHT_DEFAULT), or -2 (QZHT_NO_CHANGE), this value must equal 0.

Length of nack targets

The length of the information for the Nack targets entry.

Note: If Offset to nack targets equals 0 (QZHT_NONE), -1 (QZHT_DEFAULT), or -2 (QZHT_NO_CHANGE), this value must equal 0.

Nack targets

A list of acknowledgment target descriptions associated with the triggered cache manager server referenced by the trigger handler description and used later, at server startup, to obtain information as to where failed process completion messages are sent. Descriptions must be listed by name, where each name is separated by one or more spaces, and padded with blanks if necessary. The default setting is to reference the descriptions currently designated as default acknowledgment target descriptions for the triggered cache manager server. If there are no descriptions currently designated as default, a null list is used. A null list indicates failed process completion messages are not sent.

Note: If a null list is specified, Offset to nack targets must equal 0 (QZHT_NONE), -1 (QZHT_DEFAULT), or -2 (QZHT_NO_CHANGE). See *Offset to nack targets* for more details. Messages concerning failed trigger process completion referencing this handler are sent to all listed acknowledgment targets. An escape message is sent if any referenced description does not currently exist.

Number of threads

The number of concurrent threads the triggered cache manager server spawns when processing triggers sent to this trigger handler. The value must be greater than 0 and less than 2_{31} (or 2.147×10^9), or equal to one of the special values described below. The default value is 10.

Special values and their meanings are as follows:

-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: The current value is not changed.

Offset to ack targets

The offset from the beginning of the request variable to the Ack targets data, in bytes. The value must be greater than 0, or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NONE: A null list is specified for Ack targets indicating successful process completion messages are not sent.
-1	QZHT_DEFAULT: The default descriptions, if any, are referenced for Ack targets.
-2	QZHT_NO_CHANGE: The current references for Ack targets are not changed.

Offset to cache targets

The offset from the beginning of the request variable to the Cache targets data, in bytes. The value must be greater than 0, or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NONE: A null list is specified for Cache targets indicating data is not sent to cache targets.
-1	QZHT_DEFAULT: The default descriptions, if any, are referenced for Nack targets.
-2	QZHT_NO_CHANGE: The current references for Nack targets are not changed.

Offset to nack targets

The offset from the beginning of the request variable to the Nack targets data, in bytes. The value must be greater than 0, or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NONE: A null list is specified for Nack targets indicating failed process completion messages are not sent.
-1	QZHT_DEFAULT: The default descriptions, if any, are referenced for Nack targets.
-2	QZHT_NO_CHANGE: The current references for Nack targets are not changed.

Server name

The name used to identify the triggered cache manager server for which the new description is associated (left justified and padded with blanks if necessary). The value must specify a server name, or be one of the special values described below. If a server name that is different than the one with which the description is currently associated is specified, it is removed from its current association and added for the new server.

Special values and their meanings are as follows:

***SAME**

QZHT_NO_CHANGE_CHAR: The current server association is not changed.

Trigger handler description name

The name used by the new trigger handler description (left justified and padded with blanks if necessary). The value must specify a description name, or be one of the special values described below.

Special values and their meanings are as follows:

***SAME**

QZHT_NO_CHANGE_CHAR: The current name is not changed.

Note: Trigger handler description names must be unique for each triggered cache manager server.

Trigger handler type

The trigger handler type (left justified and padded with blanks if necessary). The value must be one of the special values described below.

Special values and their meanings are as follows:

***UPDATE**

QZHT_UPDATE_TYPE: A *UPDATE type is described.

***PUBLISH**

QZHT_PUBLISH_TYPE: A *PUBLISH type is described.

***SAME**

QZHT_NO_CHANGE_CHAR: The current type is not changed.

Note: If a type (that is different than the current type) is specified, certain information (for the current description type) is discarded if it cannot be mapped to values for the new description type.

THDP0200 format:

Offset		Type	Field
Dec	Hex		
0	0		Everything from THDP0100 format
136	88	Binary(4)	Cache request queue priority
140	8C	Binary(4)	Trigger queue collapse policy

THDP0200 format field descriptions:

Note: If the current description type is not *UPDATE, all information currently stored for the trigger handler description (that is not mapped to one of the following entries) is discarded.

Cache request queue priority

Specifies the trigger handler priority value when submitting requests to the cache targets. Lower values indicate higher priority. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below. The default value is 2^{31} (the lowest priority).

Special values and their meanings are as follows:

-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: If the current description type is *UPDATE, the current value is not changed. If the current description type is not *UPDATE, the default value is used.

Note: Triggered cache manager servers queue the trigger handler requests to the cache targets and process them according to queue priority. Requests from trigger handlers with higher priority are processed before requests from trigger handlers with lower priority. The trigger handler queue priority can be changed while servers are active by using the -chspriority command in a trigger message.

Trigger queue collapse policy

Specifies if identical triggers waiting on the request queue, for this trigger handler, are collapsed. The value must equal one of the special values described below. The default value is 1 (QZHT_YES).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: Identical triggers are collapsed.
0	QZHT_DEFAULT: The default value is used.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: If the current description type is *UPDATE, the current value is not changed. If the current description type is not *UPDATE, the default value is used.

Note: Only triggers using the *-objects* keyword can be collapsed. Identical triggers are those having an identical set of listed objects. The order of the listed objects is not important. Once a trigger handler begins processing a trigger, it is not collapsed.

THDP0300 format:

Offset		Type	Field
Dec	Hex		
0	0		Everything from THDP0100 format
136	88	Char(32)	Object dependency graph
168	A8	Char(32)	Rule set
200	C8	Binary(4)	Offset to traversal edge name
204	CC	Binary(4)	Length of traversal edge name
208	D0	Binary(4)	Offset to default included file
212	D4	Binary(4)	Length of default included file
216	D8	Binary(4)	Include dependency information
220	DC	Binary(4)	Include triggered file information
224	E0	Binary(4)	Include cached file information
		Char(*)	Traversal edge type
		Char(*)	Default include file

THDP0300 format field description:

Note: If the current description type is not *PUBLISH, all information currently stored for the trigger handler description (that is not mapped to one of the following entries) is discarded.

Default included file

The name of a file the trigger handler includes, by global default, as a replacement for included files that have not been triggered (when a local default file is not specified or available). The file name must be left justified and padded with blanks if necessary. The default value is null, indicating that a global default file name is not specified for this trigger handler description.

Note: If the value is null, Offset to default included file must equal 0 (QZHT_NONE), -1 (QZHT_DEFAULT) or -2 (QZHT_NO_CHANGE). See Offset to default included file for more details. The file specified as a global default must be triggered (at runtime) before it can be used.

Include cached file information

Specifies if a list of names for all files sent to cache targets, as a result of handling original trigger requests, is included in successful process completion messages. The value must equal one of the special values described below. The default value is 0 (QZHT_NO).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: A list of names is included.
----------	---

0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: A list of names is not included.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: If the current description type is *PUBLISH, the current value is not changed. If the current description type is not *PUBLISH, the default value is used.

Include dependency information

Specifies if information concerning all dependent files, assembled into triggered files as a result of handling original trigger request, is included in successful process completion messages. The value must equal one of the special values described below. The default value is 0 (QZHT_NO).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: Information is included.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: Information is not included.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: If the current description type is *PUBLISH, the current value is not changed. If the current description type is not *PUBLISH, the default value is used.

Include triggered file information

Specifies if a list of names for all files triggered, as a result of handling original trigger requests, is included in successful process completion messages. The value must equal one of the special values described below. The default value is 0 (QZHT_NO).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: A list of names is included.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: A list of names is not included.
-1	QZHT_DEFAULT: The default value is used.
-2	QZHT_NO_CHANGE: If the current description type is *PUBLISH, the current value is not changed. If the current description type is not *PUBLISH, the default value is used.

Length of default included file

The length of information for the *Default included file* entry.

Note: If *Offset to traversal edge type* equals 0 (QZHT_NONE), -1 (QZHT_DEFAULT), or -2 (QZHT_NO_CHANGE), this value must equal 0.

Length of traversal edge type

The length of the information for the *Traversal edge type* entry.

Note: If *Offset to traversal edge type* equals 0 (QZHT_NONE), -1 (QZHT_DEFAULT), or -2 (QZHT_NO_CHANGE), this value must equal 0.

Object dependency graph

The name of an object dependency graph description associated with the triggered cache manager server referenced by the new trigger handler description and used later, at server startup, to identify which object dependency graph is used by the handler to record and obtain object dependency information. The value must be an object dependency graph description name, or one of the special values described below (left justified and padded with blanks if necessary). The default setting is to reference the description currently designated as the default object dependency graph description for the triggered cache manager server.

Special values and their meanings are as follows:

*DEFAULT

QZHT_DEFAULT_CHAR: The default is referenced.

***SAME**

QZHT_NO_CHANGE: If the current description type is *PUBLISH, the current reference is not changed. If the current description type is not *PUBLISH, the default description is referenced.

Note: An object dependency graph description is required. An escape message is sent if the referenced description does not currently exist, or if *DEFAULT is specified and there is currently no default object dependency graph description for the server.

Offset to default included file

The offset from the beginning of the request variable to the Default included file data, in bytes. The value must be greater than 0, or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
-1	QZHT_DEFAULT: The default value for Default included file is used.
-2	QZHT_NO_CHANGE: If the current description type is *PUBLISH, the current file name, if any, is not changed for Default included file. If the current description type is not *PUBLISH, the default value is used for Default included file.

Offset to traversal edge type

The offset from the beginning of the request variable to the Traversal edge type data, in bytes. The value must be greater than 0, or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NONE: Same as -1 (QZHT_DEFAULT).
-1	QZHT_DEFAULT: The default value for Traversal edge type is used.
-2	QZHT_NO_CHANGE: If the current description type is *PUBLISH, the current name is not changed for Traversal type edge. If the current description type is not *PUBLISH, the default name is used for Traversal edge type.

Rule set

The name of a rule set associated with the triggered cache manager server referenced by the new trigger handler description and used later, at server startup, to identify which publishing rules are used by the handler. The value must be a rule set name, or one of the special values described below (left justified and padded with blanks if necessary).

Special values and their meanings are as follows:

***DEFAULT**

QZHT_DEFAULT_CHAR: The default rule set currently associated with the triggered cache manager server is referenced.

***NONE**

QZHT_NONE_CHAR: No rule set is referenced by the trigger handler.

***SAME**

QZHT_NO_CHANGE: If the current description type is *PUBLISH, the current reference if any, is not changed. If the current description type is not *PUBLISH, the default Rule Set currently associated with the triggered cache manager server is referenced.

Note: Triggers that specify files that do not match any publishing rules, in the specified rule set, are processed according to the default publishing rule. That is, they are read from the data source and sent to all cache targets (they are not parsed). An empty rule set has the same

affect as not specifying a rule set at all. An escape message is sent if the referenced description does not currently exist, or if *DEFAULT is specified and there is currently no default rule set for the server.

Traversal edge type

The name of an object dependency graph (ODG) edge type used by the trigger handler to determine object dependencies when assembling files. The edge type name must be left justified and padded with blanks if necessary. The default name is **composition**.

Note: The name of an object dependency graph (ODG) edge type used by the trigger handler to determine object dependencies when assembling files. The edge type name must be left justified and padded with blanks if necessary. The default name is composition.

Error messages:

TCM7001 E

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7031 E

Request variable format is not valid.

TCM7033 E

Length of request variable is not valid for the specified format.

TCM703E E

Error code parameter is not valid.

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 E

A configuration file for triggered cache manager server &1 was not found.

TCM7222 E

A default &1 is not designated for triggered cache manager server &2.

TCM7290 E

&1 &2 was not found for triggered cache manager server &3.

TCM7293 E

A &1 using the name &2 already exists for triggered cache manager server &3.

TCM72C0 E

Triggered cache manager &1 name is not valid.

TCM72C1 E

Triggered cache manager &1 type is not valid.

TCM72C2 E

Triggered cache manager description type &1 cannot be specified when using data format &2.

TCM72C3 E

Value passed to offset &1 is not valid.

TCM74C0 E

Triggered cache manager server name is not valid.

Create Triggered Cache Manager Server (QzhtCrtTCMServer) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 request variable	Input	Char(*)
2 length of request variable	Input	Binary(4)
3 request variable format	Input	Char(8)
4 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

Use the QzhtCrtTCMServer API to create triggered cache manager servers. Configuration information associated with servers is stored separately for each server. The API is a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

request variable

INPUT: CHAR(*)

The variable used to pass information used to create a new triggered cache manager servers.

length of request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for *Request variable*.

request variable format

INPUT: CHAR(8)

The format name of Request variable data. The following values must be used:

- INDP0100: Basing server information for server data.

error code

I/O: CHAR(*)

The structure in which to return error information.

INDP0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Binary(4)	Autostart property
36	24	Binary(4)	Local TCP port
40	28	Char(32)	Basing server

Field descriptions:

Server name

The name used by the new triggered cache manager server (left justified and padded with blanks if necessary).

Note: Server names must be unique.

Autostart property

Specifies if the new triggered cache manager server is to start when startup of *AUTOSTART triggered cache manager servers is requested. Usually *AUTOSTART servers are requested to start when TCP/IP is started, however they may also be requested via the STRTCPSVR command or QzhtStrTCMServer API. The default value is 0 (QZHT_NO).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The new server is to start when startup of *AUTOSTART servers is requested.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The new server is not to start when startup of *AUTOSTART servers is requested.
-1	QZHT_DEFAULT: The default value is used.

Local TCP port

The TCP port number used by the new triggered cache manager server. The value must be greater than 0 and less than 65536, or equal the special value described below. The default value is 7049.

The special value and its meaning is as follows:

-1	QZHT_DEFAULT: The default value is used.
----	--

Note: Triggered cache manager servers use the same port number for all IP interfaces.

Basing server

The name of an existing triggered cache manager server from which all configuration data is obtained and used to create the new triggered cache manager server (left justified and padded with blanks if necessary). The value must be the name of an existing triggered cache manager server, or one of the special values described below. If the *DEFAULT special value is used, the default configuration is used to create the new triggered cache manager server.

The special value and its meaning is as follows:

***Default**

QZHT_DEFAULT_CHAR: The default configuration is used for the new server.

*Error messages:***TCM7001 E**

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7031 E

Request variable format is not valid.

TCM7033 E

Length of request variable is not valid for the format specified.

TCM703E E

Error code parameter is not valid.

TCM70F0 E

Unknown error occurred while processing request.

TCM7103 E

User QTCM is not authorized to the configuration file for the basing triggered cache manager server &1.

TCM7193 E

A configuration file for the basing triggered cache manager server &1 was not found.

TCM72C3 E

Value passed at offset &1 is not valid.

TCM4937 E

Triggered cache manager server using name &1 already exists.

TCM74C0 E

Triggered cache manager server name is not valid.

TCM734C3 E

Basing triggered cache manager server name is not valid.

Delete Triggered Cache Manager Server Status (QzhtDltTCMServer) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 request variable	Input	Char(*)
2 length of request variable	Input	Binary(4)
3 request variable format	Input	Char(*)
4 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

Use the QzhtDltTCMServer API to delete triggered cache manager servers including all associated configuration information and all object dependency graph (ODG) data files. The API is a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:**request variable**

INPUT: CHAR(*)

The variable used to the pass information used to delete an existing triggered cache manager server.

length of request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for request variable.

request variable format

INPUT: CHAR(8)

The format name of request variable data. The following values must be used:

- INNP0100: Basic Server name format.

error code

I/O: CHAR(*)

The structure in which to return error information.

INNP0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name

*Field description:***Server name**

The name used to identify which triggered cache manager server is deleted (left justified and padded with blanks if necessary).

*Error messages:***TCM7001 E**

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7031 E

Request variable format is not valid.

TCM7033 E

Length of request variable is not valid for the format specified.

TCM703E E

Error code parameter is not valid.

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7301 E

User QTCM is not authorized to triggered cache manager server &1.

TCM7390 E

Triggered cache manager server &1 not found.

TCM73C3 E

Triggered cache manager server &1 is active.

TCM7401 E

User QTCM is not authorized to server data for triggered cache manager server &1.

TCM74C0 E

Triggered cache manager server name is not valid.

TCM7701 E

User QTCM is not authorized to object dependency graph data for triggered cache manager server &1.

End Triggered Cache Manager Server (QzhtEndTCMServer) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 request variable	Input	Char(*)
2 length of request variable	Input	Binary(4)
3 Request variable format	Input	Char(8)
4 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINOPS

Threadsafe: Yes

Use the *QzhtEndTCMServer* API to end triggered cache manager servers. The API is a callable service implemented as an ILE entry point within the QZHTINOPS *SRVPGM in the QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks: None.

Required parameter group:

request variable

INPUT: CHAR(*)

The variable user ID to pass information used to end active triggered cache manager servers.

length of request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for Request variable.

request variable format

INPUT: CHAR(8)

The format name of Request variable data. The following values must be used:

- INNP0100: Basic server name format.

error code

I/O:CHAR(*)

The structure in which to return error information.

INNP0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name

Field description:

Server name

The name used to identify which triggered cache manager server is stopped (left justified and padded with blanks if necessary). The value must specify a server name, or be one of the special values described below.

Special values and their meanings are as follows:

*ALL QZHT_ALL_CHAR: All triggered cache manager servers are to stopped.

Error messages:

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7031 E

Request variable format is not valid.

TCM7033 E

Length of request variable is not valid for the format specified.

TCM703E E

Error code parameter is not valid.

TCM70F0 E

Unknown error occurred while processing request.

TCM7390 E

Triggered cache manager server &1 not found.

TCM73C0 E

Triggered cache manager server &1 is not active.

TCM7401 E

User QTCM is not authorized to server data for triggered cache manager server &1.

TCM74C0 E

Triggered cache manager server name is not valid.

Get Triggered Cache Manager Server Status (QzhtGetTCMServerStatus) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1	receiver variable	Output	Char(*)
2	length of receiver variable	Input	Binary(4)
3	receiver variable format	Input	Char(8)
4	server name	I/O	Char(32)
5	error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

The *QzhtGetTCMServerStatus* API retrieves the current status (or state) of triggered cache manager servers along with their respective autostart property. The API is a callable service implemented as an ILE entry point within QZHTINCONF *SRVPGM in QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks: None.

Required parameter group:

receiver variable

OUTPUT: CHAR(*)

The variable used to return the information indicating the status and autostart property of triggered cache manager servers.

length of receiver variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for receiver variable. This value must be greater than or equal to 8.

receiver variable format

INPUT: CHAR(8)

The format name of Receiver variable data. The following values must be used:

- INSG0100: Basic status information format for triggered cache manager servers.

server name

INPUT: CHAR(32)

The name used to identify the server for which the status information is returned (left justified and padded with blanks if necessary). The value must specify a server name, or be one of the special values described below.

The special value and its meaning is as follows:

- ***ALL** QZHT_ALL_CHAR: Status information for all triggered cache manager servers returned.

error code

I/O:CHAR(*)

The structure in which to return error information.

INSG0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Binary(4)	Bytes returned
4	4	Binary(4)	Bytes available
8	8	Binary(4)	Offset to server entries
12	C	Binary(4)	Number of server entries
16	10	Binary(4)	Length of server entry
Server entries			
Note: The following entries are repeated for each set of server data returned.			
		Char(32)	Server name
		Binary(4)	Autostart property
		Binary(4)	Current state
		Binary(4)	Local TCP port

Field descriptions:

Bytes returned

The number of information bytes returned to the caller of the API.

Bytes available

The number of information bytes available for return to the caller of the API.

Note: If this value is greater than the value of Bytes Returned, the receiver variable was not large enough to return all information.

Offset to server entries

The offset from the beginning of the receiver variable to the Server entries data, in bytes.

Number of server entries

The number of entries returned in server entries.

Note: If no servers exist, the returned value equals 0 (QZHT_NONE).

Length of server entry

The length of the information for each entry in server entries.

Server name

The name of the triggered cache manager server (left justified and padded with blanks if necessary).

Autostart property

Indicates whether the new triggered cache manager server is to start when startup of *AUTOSTART triggered cache manager servers is requested. Usually *AUTOSTART servers are requested to start when TCP/IP is started, however they may also be requested via the STRTCPSVR command or QzhtStrTCMServer API.

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The server is set to start when startup of *AUTOSTART servers is requested.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The server is not set to start when startup of *AUTOSTART servers is requested.

Current state

The current state of the triggered cache manager server identified by Server name. The possible values are:

1	QZHT_INACTIVE: The server is not active.
2	QZHT_STARTING: The server is in the process of starting up.
3	QZHT_ACTIVE: The server is active.
4	QZHT_ENDING: The server is in the process of shutting down.

Local TCP port

The number of one of the TCP ports used by the triggered cache manager server. The value will be greater than 0 and less than 65536.

Note: Triggered cache manager servers use the same port number for all IP interfaces.

Error messages:

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7032 E

Receiver variable format is not valid.

TCM7034 E

Length of receiver variable is not valid for the format specified.

TCM7035 E

Server parameter is not valid.

TCM703E E

Error code parameter is not valid.

TCM70F0 E

Unknown error occurred while processing request.

TCM7390 E

TCM server instance server &1 not found.

TCM74C0 E

TCM server instance server name is not valid.

Remove Triggered Cache Manager Acknowledgment Target Description (QzhtRmvTCMAckTargetDesc) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 request variable	Input	Char(*)
2 length of request variable	Input	Binary(4)
3 request variable format	Input	Char(8)
4 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

Use the QzhtRmvTCMAckTargetDesc API to remove acknowledgment target descriptions associated with triggered cache manager servers. Trigger handler descriptions, currently referencing the removed acknowledgment target description, are not changed however they are no longer be valid the next time the servers are started. The API is a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

*Required parameter group:***request variable**

INPUT: CHAR(*)

The variable used to pass the information used to remove an existing acknowledgment target description.

length of request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for request variable.

request variable format

INPUT: CHAR(8)

The format name of the Request variable data. The following values must be used:

- DNIP0100: Identity information format for configuration descriptions.

error code

I/O: CHAR(*)

The structure in which to return error information.

DNIP0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Char(32)	Description name

Field descriptions:

Description name

The name used to identify the removed acknowledgment target description (left justified and padded with blanks if necessary).

Note: An escape message is sent if the description is removed while referenced by other descriptions. An escape message is sent if the description is removed while being referenced by other descriptions.

Server name

The name used to identify the triggered cache manager server for which the new description is associated (left justified and padded with blanks if necessary).

Error messages:

TCM7001 E

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7031 E

Request variable format is not valid.

TCM7033 E

Length of request variable is not valid for the specified format.

TCM703E E

Error code parameter is not valid.

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 E

A configuration file for triggered cache manager server &1 was not found.

TCM72C0 E

Triggered cache manager &1 name is not valid.

TCM72C3 E

Value passed to offset &1 is not valid.

TCM74C0 E

Triggered cache manager server name is not valid.

Remove Triggered Cache Manager Cache Target Description (QzhtRmvTCMCacheTargetDesc) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 request variable	Input	Char(*)
2 length of request variable	Input	Binary(4)
3 request variable format	Input	Char(8)
4 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

Use the QzhtRmvTCMCacheTargetDesc API to remove cache target descriptions associated with triggered cache manager servers. Trigger handler descriptions, currently referencing the cache target description, are removed (not changed), however they are no longer be valid the next time the servers are started. The API is a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

request variable

INPUT: CHAR(*)

The variable used to pass the information used to add an existing cache target description. See Description identity formats for more information.

length of request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for request variable.

request variable format

INPUT: CHAR(8)

The format name of the Request variable data. The following values must be used:

- DNIP0100: Identity information format for configuration descriptions.

error code

I/O: CHAR(*)

The structure in which to return error information.

DNIP0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Char(32)	Description name

Field descriptions:

Description name

The removed cache target description name (left justified and padded with blanks if necessary).

Note: An escape message is sent if the description is removed while being referenced by other descriptions.

Server name

INPUT: CHAR(32)

The name used to identify the triggered cache manager server for which the new description is associated (left justified and padded with blanks if necessary).

*Error messages:***TCM7001 E**

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7031 E

Request variable format is not valid.

TCM7033 E

Length of request variable is not valid for the specified format.

TCM703E E

Error code parameter is not valid

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 E

A configuration file for triggered cache manager server &1 was not found.

TCM72C0 E

Triggered cache manager &1 name is not valid.

TCM72C3 E

Value passed to offset &1 is not valid.

TCM74C0 E

Triggered cache manager server name is not valid.

Remove Triggered Cache Manager Data Source Description (QzhtRmvTCMDataSourceDesc) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 request variable	Input	Char(*)
2 length of request variable	Input	Binary(4)
3 request variable format	Input	Char(8)
4 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

Use the QzhtRmvTCMDataSourceDesc API to remove data source descriptions associated with triggered cache manager servers. Trigger handler descriptions currently referencing the removed data source descriptions are not changed, however they are no longer valid after the servers are started. The API is a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

request variable

INPUT: CHAR(*)

The variable used to pass the information used to remove a data source description. See Description identity formats for more information.

length of request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for Request variable.

request variable format

INPUT: CHAR(8) The format name of the Request variable data. The following values must be used:

- DSDP0100: Basic information format for configuration descriptions.

Note: See Description identity formats for more information.

error code

I/O: CHAR(*)

The structure in which to return error information.

DSDP0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Char(32)	Data source description name

Field descriptions:

Note: Data source descriptions are added using default values for all unspecified values according to the type specified by Data source type). See other data source description formats for details regarding these default values.

Data source description name

The name used by the new data source description (left justified and padded with blanks if necessary).

Note: An escape message is sent if the description is removed while being referenced by trigger handler descriptions.

Server name

The name used to identify the triggered cache manager server for which the description is removed (left justified and padded with blanks if necessary).

*Error messages:***TCM7001 E**

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7031 E

Request variable format is not valid.

TCM7033 E

Length of request variable is not valid for the specified format.

TCM703E E

Error code parameter is not valid.

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 E

A configuration file for triggered cache manager server &1 was not found.

TCM72C0 E

Triggered cache manager &1 name is not valid.

TCM72C3 E

Value passed to offset &1 is not valid.

TCM74C0 E

Triggered cache manager server name is not valid.

Remove Triggered Cache Manager Host Description (QzhtRmvTCMHostDesc) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 request variable	Input	Char(*)
2 length of request variable	Input	Binary(4)
3 request variable format	Input	Char(8)
4 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

The QzhtRmvTCMHostDesc API removes host descriptions associated with triggered cache manager servers. Data source, cache target, and acknowledgement target descriptions currently referencing the removed host descriptions are not changed, however they are no longer be valid the next time the servers are started.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

request variable

INPUT: CHAR(*)

The variable used to pass information used to remove an existing host description.

length of request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for the request variable.

request variable format

INPUT: CHAR(8)

The format name of the request variable data. The following values must be used:

- DNIP0100: Identity information format for configuration descriptions.

error code

I/O: CHAR(*)

The structure in which to return error information.

DNIP0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Char(32)	Description name

Field descriptions:

Description name

The name of the removed host description (left justified and padded with blanks if necessary).

Note: An escape message is sent if the description is removed.

Server name

The name used to identify the triggered cache manager server for which the description is removed (left justified and padded with blanks if necessary).

Error messages:

TCM7001 E

User &1 is not authorized.

TCM7030 E

Server error occurred while addressing parameter list.

TCM7031 E

Request variable format is not valid.

TCM7033 E

Length of request variable is not valid for the format specified.

TCM703 E

Error code parameter is not valid.

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 E

A configuration file for triggered cache manager server &1 was not found.

TCM72C0 E

Triggered cache manager &1 name is not valid.

TCM72C3 E

Value passed at offset &1 is not valid.

TCM74C0 E

Triggered cache manager server name is not valid.

Remove Triggered Cache Manager Object Dependency Graph Description (QzhtRmvTCMODGDesc) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 request variable	Input	Char(*)
2 length of request variable	Input	Binary(4)
3 request variable format	Input	Char(8)
4 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

The *QzhtRmvTCMODGDesc* API removes object dependency graph descriptions associated with triggered cache manager servers. Trigger handler descriptions, currently referencing the removed object dependency graph description, are not changed, however they are no longer be valid the next time the servers are started.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:**request variable**

INPUT: CHAR(*)

The variable used to pass information used to remove an existing object dependency graph description.

length of request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for the request variable.

request variable format

INPUT: CHAR(8)

The format name of the request variable data. The following values must be used:

- DNIP0100: Identity information format for configuration descriptions.

error code

I/O: CHAR(*)

The structure in which to return error information.

OGDP0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Char(32)	Description name

OGDP0100 format field descriptions:

Description name

The name of the removed object dependency graph description (left justified and padded with blanks if necessary).

Note: An escape message is sent if the description is removed while it is being referenced by other descriptions.

Server name

The name used to identify the triggered cache manager server for which the description is removed (left justified and padded with blanks if necessary).

Error messages:

TCM7001 E

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7031 E

Request variable format is not valid.

TCM7033 E

Length of request variable is not valid for the format specified.

TCM703E E

Error code parameter is not valid.

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 E

A configuration file for triggered cache manager server &1 was not found.

TCM72C0 E

Triggered cache manager &1 name is not valid.

TCM72C3 E

Value passed at offset &1 is not valid.

TCM74C0 E

Triggered cache manager server name is not valid.

Remove Triggered Cache Manager Publishing Rule (QzhtRmvTCMPublishingRule) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 request variable	Input	Char(*)
2 length of request variable	Input	Binary(4)
3 request variable format	Input	Char(8)
4 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

The QzhtRmvTCMPublishingRule API removes publishing rules associated with triggered cache manager servers. Rule Sets currently referencing the removed publishing rule are not changed, however they are no longer valid the next time that the servers are started. The API is a callable service implemented as an ILE entry point within the QZHTINOPS *SRVPGM in the QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

request variable

INPUT: CHAR(*)

The variable used to pass information used to remove an existing publishing rule.

length of request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for the request variable.

request variable format

INPUT: CHAR(8)

The format name of the request variable data. The following values must be used:

- DNIP0100: Identity information format for configuration descriptions.

error code

I/O: CHAR(*)

The structure in which to return error information.

DNIP0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Char(32)	Description name

DNIP0100 format field descriptions:

Description name

The name of the publishing rule (left justified and padded with blanks if necessary).

Note: An escape message is sent if the description is removed while it is being referenced by other descriptions.

Server name

The name used to identify the triggered cache manager server for which the description is removed (left justified and padded with blanks if necessary).

*Error messages:***TCM7001 E**

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7031 E

Request variable format is not valid.

TCM7033 E

Length of request variable is not valid for the specified format.

TCM703E E

Error code parameter is not valid.

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 E

A configuration file for triggered cache manager server &1 was not found.

TCM72C0 E

Triggered cache manager &1 name is not valid.

TCM72C3 E

Value passed to offset &1 is not valid.

TCM74C0 E

Triggered cache manager server name is not valid.

Remove Triggered Cache Manager Rule Set (QzhtRmvTCMRuleSet) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 request variable	Input	Char(*)
2 length of request variable	Input	Binary(4)
3 request variable format	Input	Char(8)
4 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

The *QzhtRmvTCMRuleSet* API removes rule sets associated with triggered cache manager servers. Trigger handler descriptions currently referencing the rule set removed description are not changed, however they are no longer be valid the next time that the servers are started. The API is a callable service implemented as an ILE entry point within the QZHTINOPS *SRVPGM in the QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

request variable

INPUT: CHAR(*)

The variable used to pass information used to remove an existing rule set.

length of request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for the request variable.

request variable format

INPUT: CHAR(8)

The format name of the request variable data. The following values must be used:

- DNIP0100: Identity information format for configuration descriptions.

error code

I/O: CHAR(*)

The structure in which to return error information.

DNIP0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Char(32)	Description name

DNIP0100 format field descriptions:

Description name

The name of the rule set (left justified and padded with blanks if necessary).

Note: An escape message is sent if the description is removed while it is being referenced by other descriptions.

Server name

The name used to identify the triggered cache manager server for which the description is removed (left justified and padded with blanks if necessary).

Error messages:

TCM7001 E

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7031 E

Request variable format is not valid.

TCM7033 E

Length of request variable is not valid for the specified format.

TCM703E E

Error code parameter is not valid.

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 E

A configuration file for triggered cache manager server &1 was not found.

TCM72C0 E

Triggered cache manager &1 name is not valid.

TCM72C3 E

Value passed to offset &1 is not valid.

TCM74C0 E

Triggered cache manager server name is not valid.

Remove Triggered Cache Manager Trigger Handler Description (QzhtRmvTCMTriggerHandlerDesc) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1	request variable	Input	Char(*)
2	length of request	Input	Binary(4)
3	request variable format	Input	Char(8)
4	error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

Use the QzhtRmvTCMTriggerHandlerDesc API to remove trigger handler descriptions associated with triggered cache manager servers. Trigger handler descriptions that are removed are no longer utilized by the triggered cache manager servers the next time they are started. The API is a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:**request variable**

INPUT: CHAR(*)

The variable used to pass the information used to remove an existing trigger handler description.

length of request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for the request variable.

request variable format

INPUT: CHAR(8)

The format name of the Request variable data. The following values must be used:

- INDP0100: Basic information format for configuration descriptions.

error code

I/O: CHAR(*)

The structure in which to return error information.

INDP0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name
32	20	Char(32)	Description name

*Field descriptions:***Description name**

The name of the removed trigger handler description (left justified and padded with blanks if necessary).

Server name

The name used to identify the triggered cache manager server from which the description is removed (left justified and padded with blanks if necessary).

*Error messages:***TCM7001 E**

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7031 E

Request variable format is not valid.

TCM7033 E

Length of request variable is not valid for the specified format.

TCM703E E

Error code parameter is not valid

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 D

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 D

A configuration file for triggered cache manager server &1 was not found.

TCM72C0 D

Triggered cache manager &1 name is not valid.

TCM72C3 D

Value passed to offset &1 is not valid.

TCM74C0 D

Triggered cache manager server name is not valid.

Retrieve Triggered Cache Manager Acknowledgment Target Description (QzhtRtvTCMAckTargetDesc) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 receiver variable	Output	Char(*)
2 length of receiver variable	Input	Binary(4)
3 receiver variable format	Input	Char(8)
4 server name	Input	Char(32)
5 description name	Input	Char(32)
6 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

Use the QzhtRtvTCMAckTargetDesc API to retrieve information from acknowledgment target descriptions associated with triggered cache manager servers. The API is a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

receiver variable

OUTPUT: CHAR(*)

The variable used to return acknowledgment target description information. See Acknowledgment target description formats for more information.

length of receiver variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for Receiver variable. This value must be greater than or equal to 8.

receiver variable format

INPUT: CHAR(8)

The format name of Receiver variable data. The following values must be used:

- ATDG0100: Basic information format for an acknowledgment target description.
- ATDG0200: Detailed information format for an *HTTP1 type acknowledgment target description.
- ATDG0210: Detailed information format for an *HTTP2 type acknowledgment target description.

server name

INPUT: CHAR(32)

The name used to identify the triggered cache manager server from which information is retrieved (left justified and padded with blanks if necessary).

description name

INPUT: CHAR(32)

The name used to identify the retrieved acknowledgment target description information (left justified and padded with blanks if necessary).

error code

I/O: CHAR(*)

The structure in which to return error information.

ATDG0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Binary(4)	Bytes returned
4	4	Binary(4)	Bytes available
8	8	Char(32)	Server name
40	28	Char(32)	Acknowledgement target description name
72	40	Char(10)	Acknowledgement target type
82	52	Char(2)	Reserved
84	56	Binary(4)	Default property
88	58	Binary(4)	Number of threads
92	5C	Binary(4)	Initial state

ATDG0100 format field descriptions:

Bytes available

The number of information bytes available for return to the caller of the API.

Note: If this value is greater than the value of Bytes returned, the receiver variable was not large enough to return all information.

Bytes returned

The number of information bytes returned to the caller of the API.

Acknowledgement target description name

The acknowledgment target description name (left justified and padded with blanks if necessary).

Acknowledgement target type

The acknowledgment target description type (left justified and padded with blanks if necessary). The value returned is one of the special values below.

Special values and their meanings are as follows:

***HTTP1**

QZHT_HTTP_TYPE1

***HTTP2**

QZHT_HTTP_TYPE2

Default property

Indicates if the description is a current default acknowledgment target description for the triggered cache manager server specified in Server name. The returned value equals one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The description is a current default acknowledgment target description for this server.
-1	QZHT_DEFAULT: The description is not a current default acknowledgment target description.

Note: Multiple default acknowledgment target descriptions are possible.

Initial state

Indicates if the triggered cache manager server message processor, for this acknowledgment target, is enabled or disabled at server startup. The returned value equals one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The message processor is enabled at server startup.
-1	QZHT_DEFAULT: The message processor is disabled at server startup.

Number of threads

The number of concurrent threads the triggered cache manager server spawns when sending completion messages to this acknowledgment target. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9).

Server name

The name of the triggered cache manager server for which the description is associated (left justified and padded with blanks if necessary).

ATDG0200 format:

Offset		Type	Field
Dec	Hex		
0	0		Everything from ATDG0100 format
96	60	Char(256)	HTTP IP interface
352	160	Binary(4)	HTTP TCP port
356	164	Binary(4)	Offset to HTTP URI root
360	168	Binary(4)	Length of HTTP URI root
364	16C	Binary(4)	HTTP keepalive
368	170	Binary(4)	Timeout
		Char(*)	HTTP URI root

ATDG0200 format field descriptions:

HTTP IP interface

The IP host name or address of the system hosting an HTTP server that accepts completion messages (left justified and padded with blanks if necessary). The value must be a host name (for example, server.mycompany.com) or dotted address (for example, 192.168.3.57).

HTTP keepalive

Specifies if connections to HTTP Server are kept open for reuse after completion messages are sent. The value must equal one of the special values described below.

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The connection is kept open after data transfer.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The connection is closed after data transfer.

HTTP TCP port

The TCP port of HTTP Server acknowledgment target. The value must be greater than 0 and less than 65536.

Note: The TCP port number is used in combination with the IP host name or address in HTTP IP interface to establish communication with HTTP Server acknowledgment target.

HTTP URI root

The path of HTTP Server URI that defines the root of this acknowledgment target (left justified and padded with blanks if necessary).

Note: All completion message requests sent to the acknowledgment target are prepended with this path.

Length of HTTP URI root

The length of the information for the HTTP URI root entry.

Offset to HTTP URI root

The offset from the beginning of the request variable to the HTTP URI root data, in bytes.

Timeout

The number of seconds the triggered cache manager server waits before canceling a read operation on sockets and ending transactions, with the HTTP host, in error. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below. The default value is 0 (QZHT_DISABLED).

Special values and their meanings are as follows:

0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The time out operation is disabled. The triggered cache manager server should never cancel a read operation.
---	--

ATDP0210 format:

Offset		Type	Field
Dec	Hex		
0	0		Everything from ATDG0100 format
96	60	Char(32)	HTTP host
128	80	Binary(4)	HTTP TCP port
132	84	Binary(4)	Offset to HTTP URI root
136	88	Binary(4)	Length of HTTP URI root
140	8C	Binary(4)	HTTP keepalive
144	90	Binary(4)	Timeout
		Char(*)	HTTP URI root

ATDP0210 format field descriptions:

HTTP host

The name of a host description referenced by the acknowledgment target description and used, at server startup, to obtain information about the system hosting an HTTP server accepting completion messages. The value must be a host description name (left justified and padded with blanks if necessary).

Note: See HTTP TCP port for more information. The referenced host description may or may not currently exist. It is possible that it was removed or associated with a different triggered cache manager server after this acknowledgment target description was last modified.

HTTP keepalive

Specifies if connections to HTTP Server are kept open for reuse after data transfer. The value must equal one of the special values described below.

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The connection is kept open after data transfer.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The connection is closed after data transfer.

HTTP TCP port

The TCP port number upon which HTTP Server listens for incoming requests. The value must be greater than 0 and less than 65536.

Note: The TCP port number is used in combination with information obtained at server startup from the host description specified in HTTP host to establish communication with HTTP Server acknowledgment target.

HTTP URI root

The path of HTTP Server URI that defines the root of this acknowledgment target (left justified and padded with blanks if necessary).

Note: All requests to send completion messages to this acknowledgment target are prepended with this path.

Length of HTTP URI root

The length of the information for the HTTP URI root entry.

Offset to HTTP URI root

The offset from the beginning of the request variable to the HTTP URI root data, in bytes.

Timeout

The number of seconds the triggered cache manager server waits before canceling a read operation on sockets and ending transactions, with the HTTP host, in error. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The time out operation is disabled. The triggered cache manager server should never cancel a read operation.
---	--

Error messages:

TCM7001 E

User &1 is not authorized.

- TCM7030 E**
Severe error occurred while addressing parameter list.
- TCM7032 E**
Receiver variable format is not valid.
- TCM7034 E**
Length of receiver variable is not valid for the specified format.
- TCM7035 E**
Server parameter is not valid.
- TCM7036 E**
Description parameter is not valid.
- TCM703E E**
Error code parameter is not valid.
- TCM70F0 E**
Unknown error occurred while processing request.
- TCM7101 E**
User QTCM is not authorized to the configuration file for triggered cache manager server &1.
- TCM7190 E**
A configuration file for triggered cache manager server &1 was not found.
- TCM7290 E**
&1 &2 was not found for triggered cache manager server &3.

Retrieve Triggered Cache Manager Basic Configuration (QzhtRtvTCMBasicConfig) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 receiver variable	Output	Char(*)
2 length of receiver variable	Input	Binary(4)
3 receiver variable format	Input	Char(8)
4 server name	Input	Char(32)
5 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

Use the QzhtRtvTCMBasicConfig API to retrieve the basic configuration information for a triggered cache manager server. The API is a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

receiver variable

OUTPUT: CHAR(*)

The variable used to return the basic configuration information for the triggered cache manager server.

length of receiver variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for receiver variable. This value must be greater than or equal to 8.

receiver variable format

INPUT: CHAR(8)

The format name of receiver variable data. The following values must be used:

- INDG0200: Detailed information format for server data.

server name

INPUT: CHAR(32)

The name used to identify the triggered cache manager server from which the configuration information is retrieved (left justified and padded with blanks if necessary).

error code

I/O: CHAR(*)

The structure in which to return error information.

INDG0200 format:

Offset		Type	Field
Dec	Hex		
0	0	Binary(4)	Bytes returned
4	4	Binary(4)	Bytes available
8	8	Char(32)	Server name
40	28	Binary(4)	Autostart property
44	2C	Binary(4)	Local TCP port
48	30	Binary(4)	Offset to root directory
52	34	Binary(4)	Length of root directory
56	38	Binary(4)	Maximum number of entries
60	3C	Binary(4)	Defer time between retries
64	40	Binary(4)	Memory buffer size
		Binary(4)	Root directory

Field descriptions:

Autostart property

Specifies if the new triggered cache manager server is to start when startup of *AUTOSTART triggered cache manager servers, is requested. Usually *AUTOSTART servers are requested to start when TCP/IP is started, however they may also be requested via the STRTCPSVR command or QzhtStrTCMServer API. The default value is 0 (QZHT_NO).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The server is set to start when startup of *AUTOSTART servers is requested.
----------	--

2	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The server is not set to start when startup of *AUTOSTART servers is requested.
---	---

Bytes available

The number of information bytes available for return to the caller of the API.

Note: If this value is greater than the value of Bytes returned, the receiver variable was not large enough to return all information.

Bytes returned

The number of information bytes returned to the caller of the API.

Defer time between retries

The number of seconds the triggered cache manager server waits between retry attempts for a failing action. The value returned is greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to the following special value: **0** QZHT_NONE: No defer time is used.

Length of root directory

The length of the information for the root directory entry.

Local TCP port

The number of one of the TCP ports used by the triggered cache manager server. The return value is greater than 0 and less than 65536.

Maximum number of entries

The maximum number of time the triggered cache manager server attempts to retry a failing action. The return value is greater than -1 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NONE: Failing actions are not retried.
-3	QZHT_NOMAX: Retry failing actions until successful.

Memory buffer size

The maximum amount of working data the triggered cache manager server attempts to store in memory, in bytes. The return value is greater than -1 and less than 2^{31} (or 2.147×10^9) bytes.

Offset to root directory

The offset from the beginning of the request variable to the root directory data, in bytes.

Root directory

The name of the local file system directory in which the triggered cache manager server maintains its persistent record of incoming transactions, left justified and padded with blanks if necessary.

Server name

The name of the triggered cache manager server for which the configuration information is retrieved (left justified and padded with blanks if necessary).

Error messages:

TCM7001 E

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7032 E

Receiver variable format is not valid.

TCM7034 E
Length of receiver variable is not valid for the specified format.

TCM7035 E
Server parameter is not valid.

TCM703E E
Error code parameter is not valid.

TCM70F0 E
Unknown error occurred while processing request.

TCM7101 E
User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 E
A configuration file for triggered cache manager server &1 was not found.

Retrieve Triggered Cache Manager Cache Target Description (QzhtRtvTCMCacheTargetDesc) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 receiver variable	Output	Char(*)
2 length of receiver variable	Input	Binary(4)
3 receiver variable format	Input	Char(8)
4 server name	Input	Char(32)
5 description name	Input	Char(32)
6 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

Use the QzhtRtvTCMCacheTargetDesc API to retrieve information from cache target descriptions associated with triggered cache manager servers. The API is a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

receiver variable

OUTPUT: CHAR(*)

The variable used to return cache target description information. See Cache target description formats for more information.

length of receiver variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for Receiver variable. This value must be greater than or equal to 8.

receiver variable format

INPUT: CHAR(8)

The format name of Receiver variable data. The following values must be used:

- CTDG0100: Basic information format for a cache target description.
- CTDG0200: Detailed information format for an *IFS type cache target description.
- CTDG0300: Detailed information format for an *HTTP1 type cache target description.
- CTDG0310: Detailed information format for an *HTTP2 type cache target description.
- CTDG0500: Detailed information format for an *ECCP1 type cache target description.
- CTDG0510: Detailed information format for an *ECCP2 type cache target description.

Note: See Cache target description formats for more information.

server name

INPUT: CHAR(32)

The name used to identify the triggered cache manager server from which the configuration information is retrieved (left justified and padded with blanks if necessary).

description name

CHAR(32)

The name used to identify which cache target description information is retrieved (left justified and padded with blanks if necessary).

error code

I/O: CHAR(*)

The structure in which to return error information.

CTDG0100 format:

Note: When the cache target description type is unknown, use the CTDG0100 format and the basic information (including cache target type) is returned. You may then use the returned cache target type to call the API and return detailed information about the cache target description (using one of the detailed information formats).

Offset		Type	Field
Dec	Hex		
0	0	Binary(4)	Bytes returned
4	4	Binary(4)	Bytes available
8	8	Char(32)	Server name
40	28	Char(32)	Cache target description name
72	48	Char(10)	Cache target type
82	52	Char(2)	Reserved
84	54	Binary(4)	Default property
88	58	Binary(4)	Number of threads
92	5C	Binary(4)	Initial state

CTDG0100 format field descriptions:

Bytes available

The number of information bytes available for return to the caller of the API.

Note: If this value is greater than the value of Bytes returned, the receiver variable was not large enough to return all information.

Bytes returned

The number of information bytes returned to the caller of the API.

Cache target description name

The cache target description name (left justified and padded with blanks if necessary).

Cache target type

The cache target description type (left justified and padded with blanks if necessary). The returned value equals one of the special values below.

Special values and their meanings are as follows:

*IFS QZHT_IFS_TYPE

*HTTP1
QZHT_HTTP_TYPE1

*HTTP2
QZHT_HTTP_TYPE2

*ECCP1
QZHT_ECCP_TYPE1

*ECCP2
QZHT_ECCP_TYPE2

Default property

Specifies if the new description is a default cache target description for the triggered cache manager server specified in Server name. The value must equal one of the special values described below. The default value is 0 (QZHT_NO).

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The description is a default cache target description for this server.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The description is not a default cache target description.

Note: Multiple default cache target descriptions are possible.

Initial state

Specifies the state that the triggered cache manager server request processor, for this cache target, is in at server startup. The value must equal one of the special values described below.

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The request processor is enabled at server startup.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The request processor is disabled at server startup.

Number of threads

The number of concurrent threads the triggered cache manager server spawns when processing requests sent to this cache target. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9).

Server name

The name of the triggered cache manager server from which the configuration information is retrieved (left justified and padded with blanks if necessary).

CTDG0200 format:

Offset		Type	Field
Dec	Hex		
0	0		Everything from CTDG0100 format
96	60	Binary(4)	Offset to local directory root
100	64	Binary(4)	Length of local directory root
		Char(*)	Local directory root

CTDG0200 format field descriptions:

Length of local directory root

The length of the information for the Local directory root entry.

Local directory root

The path to the local file system directory that defines the root for this cache target (left justified and padded with blanks if necessary).

Note: All file requests send to the cache target have this path prepended to the file name, even if an absolute file path is specified.

Offset to local directory root

The offset from the beginning of the receiver variable to the Local directory root data, in bytes.

CTDG0300 format:

Offset		Type	Field
Dec	Hex		
0	0		Everything from CTDG0100 format
96	60	Char(256)	HTTP IP interface
352	160	Binary(4)	HTTP TCP port
356	164	Binary(4)	Offset to HTTP URI root
360	168	Binary(4)	Length of HTTP URI root
364	16C	Binary(4)	HTTP keepalive
368	170	Binary(4)	Timeout
		Char(*)	HTTP URI root

CTDG0300 format field descriptions:

HTTP IP interface

The IP host name or address of the system hosting an HTTP server cache target. The value must be a host name (for example, server.mycompany.com), dotted address (for example, 192.168.3.57) (left justified and padded with blanks if necessary).

Note: See HTTP TCP port for more information.

HTTP keepalive

Specifies if connections to HTTP Server are kept open for reuse after data is transferred. The value must equal one of the special values described below. The default value is 0 (QZHT_DISABLED).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The connection is kept open after data transfer.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The connection is closed after data transfer.

HTTP TCP port

The TCP port number upon which HTTP Server listens for incoming requests. The value must be greater than 0 and less than 65536.

Note: The TCP port number is used in combination with the IP host name or address in HTTP IP interface to establish communication with HTTP Server cache target.

HTTP URI root

The path of HTTP Server URI that is the root of this cache target (left justified and padded with blanks if necessary).

Note: All file requests from this cache target have this path inserted into the URI, even if an absolute file path is specified.

Length of HTTP URI root

The length of the information for the HTTP URI root entry.

Offset to HTTP URI root

The offset from the beginning of the request variable to the HTTP URI root data, in bytes.

Timeout

The number of seconds the triggered cache manager server waits before canceling a read operation on sockets and ending transactions, with the HTTP host, in error. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The time out operation is disabled. The triggered cache manager server should never cancel a read operation.
---	--

CTDG0310 format:

Offset		Type	Field
Dec	Hex		
0	0		Everything from CTDG0100 format
96	60	Char(32)	HTTP host
128	80	Binary(4)	HTTP TCP port
132	84	Binary(4)	Offset to HTTP URI root
136	88	Binary(4)	Length of HTTP URI root
140	8C	Binary(4)	HTTP keepalive
144	90	Binary(4)	Timeout

Offset		Type	Field
Dec	Hex		
		Char(*)	HTTP URI root

CTDG0310 format field descriptions:

HTTP host

The name of a host description associated with the triggered cache manager server that is referenced by the new cache target description and used later, at server startup, to obtain information about the system hosting an HTTP server cache target. The value must be a host description name (left justified and padded with blanks if necessary).

Note: See HTTP TCP port for more information. The referenced host description may or may not currently exist. It is possible that it was removed or associated with a different triggered cache manager server after this cache target description was last modified.

HTTP keepalive

Specifies if connections to HTTP Server are kept open for reuse after data is transferred. The value must equal one of the special values described below. The default value is 0 (QZHT_DISABLED).

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The connection is kept open after data transfer.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The connection is closed after data transfer.

HTTP TCP port

The TCP port number upon which HTTP Server listens for incoming requests. The value must be greater than 0 and less than 65536.

Note: The TCP port number is used in combination with the IP host name or address in HTTP host to establish communication with HTTP Server cache target.

HTTP URI root

The path of HTTP Server URI that defines the root of this cache target (left justified and padded with blanks if necessary).

Note: All file requests from this cache target have this path inserted into the URI, even if an absolute file path is specified.

Length of HTTP URI root

The length of the information for the HTTP URI root entry.

Offset to HTTP URI root

The offset from the beginning of the request variable to the HTTP URI root data, in bytes.

Timeout

The number of seconds the triggered cache manager server waits before canceling a read operation on sockets and ending transactions, with the HTTP host, in error. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The time out operation is disabled. The triggered cache manager server should never cancel a read operation.
---	--

CTFG0500 format:

Offset		Type	Field
Dec	Hex		
0	0		Everything from CTDG0100 format
96	60	Char(32)	ECCP IP interface
352	160	Char(10)	ECCP TCP port
356	164	Char(2)	HTTP cluster IP interface
612	264	Binary(4)	HTTP cluster TCP port
616	268	Binary(4)	Offset to HTTP cluster URI root
620	26C	Binary(4)	Length of HTTP cluster URI root
624	270	Char(256)	ECCP keepalive
		Char(*)	HTTP cluster URI root

CTDG0500 format field descriptions:

ECCP IP interface

The IP host name or address of the backend IP interface to the network router hosting a web server cluster cache target. The value must be a host name (for example, server.mycompany.com) or dotted address (for example, 192.168.3.57) (left justified and padded with blanks if necessary).

Note: See ECCP TCP port for more information.

ECCP keepalive

Specifies if the connection to the network router is kept open for reuse after data is transferred. The value must equal one of the special values described below. The default value is 0 (QZHT_DISABLED).

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The connection is kept open after data transfer.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The connection is closed after data transfer.

ECCP TCP port

The TCP port number upon which the network router listens for incoming requests. The value must be greater than 0 and less than 65536.

Note: The TCP port number is used in combination with the IP host name or address, in ECCP IP interface, to establish communications with the network router cache target.

HTTP cluster IP interface

The IP host name or address of the web server cluster from which the network router is hosting a cache (left justified and padded with blanks if necessary). The value must be a host name (for example, server.mycompany.com), dotted address (for example, 192.168.3.57).

Note: See HTTP cluster TCP port for more information.

HTTP cluster TCP port

The number of the TCP port used by the triggered cache manager server to identify a particular web server cluster cache. The value returned must be greater than 0 and less than 65536.

Note: The network router described by ECCP IP interface may host multiple web server caches. The TCP port number returned here is used in combination with the IP address returned in HTTP cluster IP interface to identify a particular web server cache when communicating with the network router.

HTTP cluster URI root

The web server cluster URI path that defines the root of this cache target (left justified and padded with blanks if necessary).

Note: All file requests sent to this cache target have this path inserted into the URI, even if an absolute file path is specified.

Length of HTTP cluster URI root

The length of the information for the HTTP cluster URI root entry.

Offset to HTTP cluster URI root

The offset from the beginning of the receiver variable to the HTTP cluster URI root data, in bytes.

CTDG0510 format:

Offset		Type	Field
Dec	Hex		
0	0		Everything from CTDG0100 format
96	60	Char(32)	ECCP host
128	80	Binary(4)	ECCP TCP port
132	84	Char(256)	HTTP cluster IP interface
388	184	Binary(4)	HTTP cluster TCP port
392	188	Binary(4)	Offset to HTTP cluster URI root
396	18C	Binary(4)	Length of HTTP cluster URI root
400	190	Binary(4)	ECCP keepalive
		Char(*)	HTTP cluster URI root

CTDG0510 format field descriptions:

ECCP host

The name of a host description associated with the triggered cache manager server referenced by the new cache target description and used later, at server startup, to obtain information about the network router hosting a web server cluster cache target. The value must be a host description name (left justified and padded with blanks if necessary).

Note: See ECCP TCP port for more information. The referenced host description may or may not currently exist. It is possible that it was removed or associated with a different triggered cache manager server after this cache target description was last modified.

ECCP keepalive

Specifies if the connection to the network router is kept open for reuse after data is transferred. The value must equal one of the special values described below. The default value is 0 (QZHT_DISABLED).

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The connection is kept open after data transfer.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The connection is closed after data transfer.

ECCP TCP port

The TCP port number upon which the network router listens for incoming requests. The value must be greater than 0 and less than 65536.

Note: The TCP port number is used in combination with the IP host name or address, obtained at server startup, from the host description (specified in ECCP host) to establish communication with the network router cache target.

HTTP cluster IP interface

The IP host name or address of the web server cluster from which the network router is hosting a cache (left justified and padded with blanks if necessary). The value must be a host name (for example, server.mycompany.com), dotted address (for example, 192.168.3.57).

Note: See HTTP cluster TCP port for more information.

HTTP cluster TCP port

The TCP port number used by the triggered cache manager server to identify a particular web server cluster cache. The value must be greater than 0 and less than 65536.

Note: The network router using the address, described by ECCP host, may host multiple web server caches. The TCP port number specified here is used in combination with the IP host name or address in HTTP cluster IP interface to identify a particular web server cache when communicating with the network router.

HTTP cluster URI root

The web server cluster URI path that is used to define the root of this cache target (left justified and padded with blanks if necessary).

Note: All file requests sent to this cache target have this path inserted into the URI, even if an absolute file path is specified.

Length of HTTP cluster URI root

The length of the information for the HTTP cluster URI root entry.

Offset to HTTP cluster URI root

The offset from the beginning of the receiver variable to the HTTP cluster URI root data, in bytes.

Error messages:

TCM7001 E

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7032 E

Receiver variable format is not valid.

TCM7034 E

Length of receiver variable is not valid for the specified format.

TCM7035 E

Server parameter is not valid.

TCM7036 E

Description parameter is not valid.

TCM703E E

Error code parameter is not valid.

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 E

A configuration file for triggered cache manager server &1 was not found.

TCM7290 E

&1 &2 was not found for triggered cache manager server &3.

Retrieve Triggered Cache Manager Data Source Description (QzhtRtvTCMDataSourceDesc) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 receiver variable	Output	Char(*)
2 length of receiver variable	Input	Binary(4)
3 receiver variable format	Input	Char(8)
4 server name	Input	Char(32)
5 description name	Input	Char(32)
6 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

Use the QzhtRtvTCMDataSourceDesc API to retrieve information from data source descriptions associated with triggered cache manager servers. The API is a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

*Required parameter group:***receiver variable**

OUTPUT: CHAR(*)

The variable used to return data source description information. See Data source description formats for more information.

length of receiver variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for Receiver variable. This value must be greater than or equal to 8.

receiver variable format

INPUT: CHAR(8)

The format name of Receiver variable data. The following values must be used:

- DSDG0100: Basic information format for data source description.

- DSDG0200: Detailed information format for an *IFS type data source description.
- DSDG0300: Detailed information format for an *HTTP1 type data source description.
- DSDG0310: Detailed information format for an *HTTP2 type data source description.

See Data source description formats for more information.

server name

INPUT: CHAR(32)

The name used to identify the server from which information is retrieved (left justified and padded with blanks if necessary).

error code

I/O: CHAR(*)

The structure in which to return error information.

DSDG0100 format: When the data source description type is unknown, use the DSDG0100 format and the basic information (including data source type) is returned. You may then use the returned data source type to call the API again and return detailed information about the data source description using one of the detailed information formats.

Offset		Type	Field
Dec	Hex		
0	0	Binary(4)	Bytes returned
4	4	Binary(4)	Bytes available
8	8	Char(32)	Server name
40	28	Char(32)	Data source description name
72	48	Char(10)	Data source type
82	52	Char(2)	Reserved
84	54	Binary(4)	Default property
88	58	Binary(4)	Number of threads

DSDG0100 format field descriptions:

Bytes available

The number of information bytes available for return to the caller of the API.

Note: If this value is greater than the value of Bytes returned, the receiver variable was not large enough to return all information.

Bytes returned

The number of information bytes returned to the caller of the API.

Data source description name

The data source description name (left justified and padded with blanks if necessary).

Data source type

The data source description type (left justified and padded with blanks if necessary). The returned value is one of the special values below.

Special values and their meanings are as follows:

*IFS QZHT_IFS_TYPE

*HTTP1
QZHT_HTTP_TYPE1

*HTTP2

QZHT_HTTP_TYPE2

Default property

Indicates whether the description is the current default data source description for the triggered cache manager server specified in Server name. The returned value equals to one of the special values described below.

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The description is the current default for this server.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The description is not the current default.

Number of threads

The number of concurrent threads the triggered cache manager server spawns when processing requests that receive data from this data source. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9).

Server name

The name of the triggered cache manager server with which the description is associated (left justified and padded with blanks if necessary).

DSDG0200 format:

Offset		Type	Field
Dec	Hex		
0	0		Everything from DSDG0100 format
92	5C	Binary(4)	Offset to local directory root
96	60	Binary(4)	Length of local directory root
		Char(*)	Local directory root

DSDG0200 format field descriptions:

Length of local directory root

The length of the information for the Local directory root entry.

Local directory root

The path to the local file system directory that defines the root for this data source (left justified and padded with blanks if necessary).

Note: All file requests from this data source have this path prepended to the file name, even if an absolute path for the file is specified.

Offset to local directory root

The offset from the beginning of the receiver variable to the Local directory root data, in bytes.

DSDG0300 format:

Offset		Type	Field
Dec	Hex		
0	0		Everything from DSDG0100 format

Offset		Type	Field
Dec	Hex		
92	5C	Char(256)	HTTP IP interface
348	15C	Binary(4)	HTTP TCP port
352	160	Binary(4)	Offset to HTTP URI root
356	164	Binary(4)	Length of HTTP URI root
360	168	Binary(4)	HTTP keepalive
364	16C	Binary(4)	Timeout
		Char(*)	HTTP URI root

DSDG0300 format field descriptions:

HTTP IP interface

The IP host name or address of the system hosting an HTTP server data source (left justified and padded with blanks if necessary). The returned value is either an IP host name (for example, server.mycompany.com) or dotted IP address (for example, 192.168.3.57) (left justified and padded with blanks if necessary).

Note: See HTTP TCP port for more information.

HTTP keepalive

Indicates whether the triggered cache manager server attempts to keep connections to HTTP Server open for reuse after data is transferred. The returned value equals one of the special values described below.

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The server attempts to keep connections open after data transfer.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The server closes each connection after data transfer.

HTTP TCP port

The TCP listening port number used by HTTP Server data source. The value returned is greater than 0 and less than 65536.

Note: The returned TCP port number is used in combination with the IP host name or address returned in HTTP IP interface to establish communication with HTTP Server data source.

HTTP URI root

HTTP Server URI path that defines the root for this data source (left justified and padded with blanks if necessary).

Note: All requests for files from this data source have this path inserted into the URI, even if an absolute file path is specified.

Length of HTTP URI root

The length of the information returned for the HTTP URI root entry.

Offset to HTTP URI root

The offset from the beginning of the receiver variable to the HTTP URI root data, in bytes.

Timeout

The number of seconds the triggered cache manager server waits before canceling a read

operation on sockets and ending transactions, with the HTTP host, in error. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The time out operation is disabled. The triggered cache manager server should never cancel a read operation.
---	--

DSDG0310 format:

Offset		Type	Field
Dec	Hex		
0	0		Everything from DSDP0100 format
92	5C	Char(32)	HTTP host
124	7C	Binary(4)	HTTP TCP port
128	80	Binary(4)	Offset to HTTP URI root
132	84	Binary(4)	Length of HTTP URI root
136	88	Binary(4)	HTTP keepalive
140	8C	Binary(4)	Timeout
		Char(*)	HTTP URI root

DSDG0310 format field descriptions:

HTTP host

The name of the host description referenced by the data source description and used, at server startup, to obtain information about the system hosting an HTTP Server data source. The returned value is a host description name (left justified and padded with blanks if necessary).

Note: See HTTP TCP port for more information. The referenced host description may or may not currently exist. It is possible that it was removed or associated with a different triggered cache manager server after the data source description was last modified.

HTTP keepalive

Indicates whether the triggered cache manager server attempts to keep connections to HTTP Server open for reuse after data is transferred. The returned value equals one of the special values described below.

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The server attempts to keep connections open after data transfer.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The server closes each connection after data transfer.

HTTP TCP port

The TCP listening port number used by HTTP Server data source. The returned value is greater than 0 and less than 65536.

Note: The returned TCP port number is used in combination with the IP host name or address returned in HTTP host to establish communication with HTTP Server data source.

HTTP URI root

HTTP Server URI path that defines the root for this data source (left justified and padded with blanks if necessary).

Note: All requests for files from this data source have this path inserted into the URI, even if an absolute file path is specified.

Length of HTTP URI root

The length of the information returned for the HTTP URI root entry.

Offset to HTTP URI root

The offset from the beginning of the receiver variable to the HTTP URI root data, in bytes.

Timeout

The number of seconds the triggered cache manager server waits before canceling a read operation on sockets and ending transactions, with the HTTP host, in error. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9), or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The time out operation is disabled. The triggered cache manager server should never cancel a read operation.
---	--

*Error messages:***TCM7001 E**

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7032 E

Receiver variable format is not valid.

TCM7034 E

Length of receiver variable is not valid for the specified format.

TCM7035 E

Server parameter is not valid.

TCM7036 E

Description parameter is not valid.

TCM703E E

Error code parameter is not valid.

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 E

A configuration file for triggered cache manager server &1 was not found.

TCM7290 E

&1 &2 was not found for triggered cache manager server &3.

Retrieve Triggered Cache Manager Host Description (QzhtRtvTCMHostDesc) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP

Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 receiver variable	Output	Char(*)
2 length of receiver variable	Input	Binary(4)
3 receiver variable format	Input	Char(8)
4 server name	Input	Char(32)
5 description name	Input	Char(32)
6 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

Use the QzhtRtvTCMHostDesc API to retrieve information from host descriptions associated with triggered cache manager servers. The API is a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

receiver variable

OUTPUT: CHAR(*)

The variable used to return host description information.

length of receiver variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for Receiver variable. This value must be greater than or equal to 8.

receiver variable format

INPUT: CHAR(8)

The format name of Receiver variable data. The following values must be used:

- HSDG0100: Basic information format for a host description.

server name

INPUT: CHAR(32)

The name used to identify the triggered cache manager server from which information is retrieved (left justified and padded with blanks if necessary).

description name

INPUT: CHAR(32)

The name used to identify the retrieved host description information (left justified and padded with blanks if necessary).

error code

I/O: CHAR(*)

The structure in which to return error information.

HSDG0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Binary(4)	Bytes returned
4	4	Binary(4)	Bytes available
8	8	Char(32)	Server name
40	28	Char(32)	Host description name
72	48	Binary(4)	Default property
76	4C	Char(256)	IP interface

Field descriptions:

Bytes available

The number of information bytes available for return to the caller of the API.

Note: If this value is greater than the value of Bytes returned, the receiver variable was not large enough to return all information.

Bytes returned

The number of information bytes returned to the caller of the API.

Default property

Specifies if the new description is the default host description for the triggered cache manager server specified in Server name. The value must equal to one of the special values described below.

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The description is the default host description for this server.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The description is not the default host description.

Note: Data source, cache target, and acknowledgment target descriptions, added or changed using the *DEFAULT special value, reference the host description designated as default at the time they are added or changed.

Host description name

The name used by the host description (left justified and padded with blanks if necessary).

IP interface

The IP host name or address of the system hosting servers used by the triggered cache manager server (left justified and padded with blanks if necessary). The value must be a host name (for example, server.mycompany.com), dotted address (for example, 192.168.3.57).

Server name

The name of the triggered cache manager server for which the description is associated (left justified and padded with blanks if necessary).

Error messages:

TCM7001 E

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

- TCM7032 E**
Receiver variable format is not valid.
- TCM7034 E**
Length of receiver variable is not valid for the specified format.
- TCM7035 E**
Server parameter is not valid.
- TCM7036 E**
Description parameter is not valid.
- TCM703E E**
Error code parameter is not valid
- TCM70F0 E**
Unknown error occurred while processing request.
- TCM7101 D**
User QTCM is not authorized to the configuration file for triggered cache manager server &1.
- TCM7190 D**
A configuration file for triggered cache manager server &1 was not found.
- TCM7290 D**
&1 &2 was not found for triggered cache manager server &3.

Retrieve Triggered Cache Manager Object Dependency Graph Description (QzhtRtvTCMODGDesc) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 receiver variable	Output	Char(*)
2 length of receiver variable	Input	Binary(4)
3 receiver variable format	Input	Char(8)
4 server name	Input	Char(32)
5 description name	Input	Char(32)
6 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

The QzhtRtvTCMODGDesc API retrieves information from object dependency graph (ODG) descriptions associated with triggered cache manager servers.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

receiver variable

OUTPUT: CHAR(*)

The variable used to return object dependency graph description information.

length of receiver variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for Receiver variable. This value must be greater than or equal to 8.

receiver variable format

INPUT: CHAR(8)

The format name of Receiver variable data. The following values must be used:

- OGDG0100: Basic information format for an object dependency graph description.

server name

INPUT: CHAR(32)

The name used to identify the server for which information is retrieved (left justified and padded with blanks if necessary).

description name

INPUT: CHAR(32)

The name used to identify which object dependency graph description information is retrieved (left justified and padded with blanks if necessary).

error code

I/O: CHAR(*)

The structure in which to return error information.

OGDP0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Binary(4)	Bytes returned
4	4	Binary(4)	Bytes available
8	8	Char(32)	Server name
40	28	Char(32)	Object dependency graph description name
72	48	Binary(4)	Default property
76	52	Binary(4)	Allow API updates

Field descriptions:

Allow API updates

Indicates whether APIs are allowed to update the object dependency graph described by the object dependency graph description. The value returned is one of the special values described below.

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The object dependency graph is updated using APIs via the triggered cache manager server, as well as from trigger handler publish parsing.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The object dependency graph is not updated using APIs via the triggered cache manager server. Only updates as a result of trigger handler publish parsing are allowed.

Bytes available

The number of information bytes available for return to the caller of the API.

Note: If this value is greater than the value of Bytes returned, the receiver variable was not large enough to return all information.

Bytes returned

Number of bytes of information returned to the caller of the API.

Default property

Indicates whether the description is the current default object dependency graph description for the triggered cache manager server specified in Server name. The returned value equals one of the special values described below.

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The description is the current default for this server.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The description is not the current default.

Object dependency graph description name

The name of the object dependency graph description (left justified and padded with blanks if necessary).

Server name

The name of the triggered cache manager server with which the description is associated (left justified and padded with blanks if necessary).

*Error messages:***TCM7001 E**

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7032 E

Receiver variable format is not valid.

TCM7034 E

Length of receiver variable is not valid for the format specified.

TCM7035 E

Server parameter is not valid.

TCM7036 E

Description parameter is not valid.

TCM703E E

Error code parameter is not valid.

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is to authorized to the configuration file for triggered cache manager server &1.

TCM7190 E

A configuration file for triggered cache manager server &1 was not found.

TCM7290 E

&1 &2 was not found for triggered cache manager server &3.

Retrieve Triggered Cache Manager Publishing Rule (QzhtRtvTCMPublishingRule) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1	receiver variable	Output	Char(*)
2	length of receiver variable	Input	Binary(4)
3	receiver variable format	Input	Char(8)
4	server name	Input	Char(32)
5	description name	Input	Char(32)
6	error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

The QzhtRtvTCMPublishingRule API retrieves information from publishing rules associated with triggered cache manager servers. The API is a callable service implemented as an ILE entry point within QZHTINCONF *SRVPGM in QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

receiver variable

INPUT: CHAR(*)

The variable used to return publishing rule information.

length of receiver variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for receiver variable.

receiver variable format

INPUT: CHAR(8)

The format name of the receiver variable data. The following values must be used:

- PRDG0100: Basic information format for a publishing rule.

server name

INPUT: CHAR(32)

The name used to identify the server for which information is retrieved (left justified and padded with blanks if necessary).

description name

INPUT: CHAR(32)

The name used to identify the which publishing rule information is retrieved (left justified and padded with blanks if necessary).

error code

I/O: CHAR(*)

The structure in which to return error information.

PRDG0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Binary(4)	Bytes returned
4	4	Binary(4)	Bytes available
8	8	Char(32)	Server name
40	28	Char(32)	Publishing rule name
72	48	Binary(4)	Offset to file extensions
76	4C	Binary(4)	Length of file extensions
80	50	Binary(4)	Read from data source
84	54	Binary(4)	Send data source version
88	58	Binary(4)	Parse and assemble
92	5C	Binary(4)	Send assembled version
96	60	Binary(4)	Offset to new file extension
100	64	Binary(4)	Length of new file extension
		Char(*)	File extensions
		Char(*)	New file extensions

PRDG0100 format field descriptions:

Bytes available

The number of information bytes available for return to the caller of the API.

Note: If this value is greater than the value of Bytes returned, the receiver variable was not large enough to return all information.

Bytes returned

The number of bytes of information returned to the caller of the API.

File extensions

A list of file extensions used to identify files that are processed according to the publishing rule. File extensions are listed as a string of characters, where each extension is separated by one space (left justified and padded with blanks if necessary).

Note: File names are compared to file extensions starting at the last period in the file name.

Length of file extensions

The length of the information returned for the File extensions entry.

Length of new file extension

The length of the information returned for the New file extension entry.

New file extension

The string of characters used as a new extension to rename files after they have been assembled and before they are sent to the cache targets (left justified and padded with blanks if necessary).

Note: If the value is null, Offset to new file extension equals 0 (QZHT_NONE), indicating that the triggered cache manager server does not rename files prior to sending them to the cache targets.

Offset to new file extension

The offset from the beginning of the receiver variable to the New file extension data, in bytes. The value returned is greater than 0 or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NONE: The value is null for New file extensions.
---	---

Offset to file extensions

The offset from the beginning of the request variable to the File extensions data, in bytes.

Parse and assemble

Indicates if files matching this publishing rule are parsed for wrappers and includes, and possibly sent through the page assembler. The returned value equals one of the special values described below.

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: Files matching this publishing rule are parsed and possibly sent through the page assembler.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: Files matching this publishing rule are not parsed. Note: Files that are not parsed are not sent through the page assembler.

Publishing rule name

The name of the publishing rule, left justified and padded with blanks if necessary.

Read from data source

Indicates if files matching this publishing rule are read from the data source when triggered. The value returned is one of the special values described below.

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: Files matching this publishing rule are read from the data source when triggered.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: Files matching this publishing rule are not read from the data source.

Send assembled version

Indicates if the assembled version of files matching this publishing rule are sent to cache targets when triggered and processed. The returned value equals one of the special values described below.

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: Assembled versions of files matching this publishing rule are sent to cache targets when triggered and processed.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: Assembled versions of files matching this publishing rule are not sent to cache targets.

Send data source version

Indicates if the data source version of files matching this publishing rule (the version read from the data source prior to assembly) are sent to cache targets when triggered and processed. The value returned is one of the special values described below.

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: Data source versions of files matching this publishing rule are sent to cache targets when triggered and processed.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: Data source versions of files matching this publishing rule are not sent to cache targets.

Server name

The name of the triggered cache manager server with which the description is associated (left justified and padded with blanks if necessary).

Error messages:

TCM7001 E

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7032 E

Receiver variable format is not valid.

TCM7034 E

Length of receiver variable is not valid for the specified format.

TCM7035 E

Server parameter is not valid.

TCM7036 E

Description parameter is not valid.

TCM703E E

Error code parameter is not valid.

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 E

A configuration file for triggered cache manager server &1 was not found.

TCM7290 E

&1&2 was not found for triggered cache manager server &3.

Retrieve Triggered Cache Manager Rule Set (QzhtRtvTCMRuleSet) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 receiver variable	Output	Char(*)
2 length of receiver variable	Input	Binary(4)
3 receiver variable format	Input	Char(8)
4 server name	Input	Char(32)
5 description name	Input	Char(32)
6 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

The QzhtRtvTCMRuleSet API retrieves information from rule sets associated with triggered cache manager servers. The API is a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB. The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

receiver variable

INPUT: CHAR(*)

The variable used to return rule set information.

length of receiver variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for receiver variable. This value must be greater than or equal to 8.

receiver variable format

INPUT: CHAR(8)

The format name of the receiver variable data. The following values must be used:

- RSDG0100: Basic information format for a rule set.

server name

INPUT: CHAR(32)

The name used to identify the server for which information is retrieved (left justified and padded with blanks if necessary).

description name

INPUT: CHAR(32)

The name used to identify which rule set information is retrieved (left justified and padded with blanks if necessary).

error code

I/O: CHAR(*)

The structure in which to return error information.

RSDG0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Binary(4)	Bytes returned
4	4	Binary(4)	Bytes available
8	8	Char(32)	Server name
40	28	Char(32)	Rule set name
72	48	Binary(4)	Default property
76	52	Binary(4)	Offset to publishing rules
80	56	Binary(4)	Length of publishing rules

Offset		Type	Field
Dec	Hex		
		Char(*)	Publishing rules

RSDG0100 format field descriptions:

Bytes available

The number of information bytes available for return to the caller of the API.

Note: If this value is greater than the value of Bytes returned, the receiver variable was not large enough to return all information.

Bytes returned

BINARY(4)

The number of bytes of information returned to the caller of the API.

Default property

Indicates whether the description is the current default rule set for the triggered cache manager server specified in Server name. The returned value equals one of the special values described below.

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: The description is the current default for this server.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: The description is not the current default.

Length of publishing rules

The length of the information returned for the Publishing rules entry.

Offset to publishing rules

The offset from the beginning of the receiver variable to the Publishing rules data, in bytes. The value returned is greater than 0, or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NONE: The value is null for Publishing rules.
---	--

Publishing rules

The list of publishing rules currently referenced are listed by name, separated by one (left justified and padded with blanks if necessary).

Note: If the value is null, Offset to publishing rules equals 0 (QZHT_NONE), indicating that an empty rule set is described. An empty rule set referenced by a trigger handler causes it to process all triggers according to the default publishing rule.

Rule set object name

The name of the rule set (left justified and padded with blanks if necessary).

Server name

The name of the triggered cache manager server with which the description is associated (left justified and padded with blanks if necessary).

Error messages:

TCM7001 E

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7032 E

Receiver variable format is not valid.

TCM7034 E

Length of receiver variable is not valid for the specified format.

TCM7035 E

Server parameter is not valid.

TCM7036 E

Description parameter is not valid.

TCM703E E

Error code parameter is not valid

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 E

A configuration file for triggered cache manager server &1 was not found.

TCM7290 E

&1&2 was not found for triggered cache manager server &3.

Retrieve Triggered Cache Manager Trigger Handler Description (QzhtRtvTCMTriggerHandlerDesc) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1	receiver variable	Output	Char(*)
2	length of receiver variable	Input	Binary(4)
3	receiver variable format	Input	Char(8)
4	server name	Input	Char(32)
5	description name	Input	Char(32)
6	error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINCONF

Threadsafe: Yes

Use the QzhtRtvTCMTriggerHandlerDesc API to retrieve information from trigger handlers associated with triggered cache manager servers. The API is a callable service implemented as an ILE entry point within the QZHTINCONF *SRVPGM in the QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks:

- *IOSYSCFG special authority

Required parameter group:

receiver variable

OUTPUT: CHAR(*)

The variable used to return trigger handler description information.

length of receiver variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for Receiver variable. This value must be greater than or equal to 8.

receiver variable format

INPUT: CHAR(8)

The format name of Receiver variable data. The following values must be used:

- THDG0100: Basic information format for a trigger handler description.
- THDG0200: Detailed information format for an *UPDATE type trigger handler description.
- THDG0300: Detailed information format for a *PUBLISH type trigger handler description.

server name

INPUT: CHAR(32)

The name used to identify the server for which information is retrieved (left justified and padded with blanks if necessary).

description name

INPUT: CHAR(32)

The name used to identify which trigger handler description information is retrieved (left justified and padded with blanks if necessary).

error code

I/O: CHAR(*)

The structure in which to return error information.

THDG0100 format: When the trigger handler description type is unknown, use the THDG0100 format and the basic information (including trigger handler type) is returned.

Offset		Type	Field
Dec	Hex		
0	0	Binary(4)	Bytes returned
4	4	Binary(4)	Bytes available
8	8	Char(32)	Server name
40	28	Char(32)	Trigger handler description name
72	48	Char(10)	Trigger handler type
82	52	Char(2)	Reserved
84	54	Char(32)	Data source
116	74	Binary(4)	offset to cache targets
120	78	Binary(4)	Length of cache targets
124	7C	Binary(4)	Offset to ack targets
128	80	Binary(4)	Length of ack targets
132	84	Binary(4)	Offset to nack targets
136	88	Binary(4)	Length of nack targets
140	8C	Binary(4)	Number of threads

Offset		Type	Field
Dec	Hex		
		Char(*)	Cache targets
		Char(*)	Ack targets
		Char(*)	Nack targets

THDG0100 format field descriptions:

Ack targets

A list of acknowledgment target descriptions referenced by the new trigger handler description and used, at server startup, to obtain information as to where successful process completion messages are sent. Descriptions must be listed by name, where each name is separated by one or more spaces (left justified and padded with blanks if necessary).

Note: If a null list is returned, Offset to ack targets must equal 0 (QZHT_NONE), indicating no acknowledgment target descriptions are currently referenced. Messages concerning successful process completion of triggers referencing this handler are sent to all listed acknowledgment targets.

Bytes available

The number of information bytes available for return to the caller of the API.

Note: If this value is greater than the value of Bytes returned, the receiver was not large enough to return all information.

Bytes returned

The number of information bytes returned to the caller of the API.

Cache targets

The list of cache target descriptions referenced by the trigger handler descriptions and used, at server startup, to obtain information about the cache targets to which the trigger handler sends data. Returned descriptions must be listed by name, where each name is separated by one or more spaces, left justified, and padded with blanks if necessary.

Note: If a null list is returned, Offset to cache targets must equal 0 (QZHT_NONE), indicating no cache target descriptions are currently referenced. Data processed by this trigger handler is sent to all listed cache targets.

Data source

The name of a data source description referenced by the new trigger handler description and used, at server startup, to obtain information about the data source from which the trigger handler retrieves data. The value must be a data source description name (left justified and padded with blanks if necessary).

Length of ack targets

The length of the information returned for the Ack targets entry.

Length of cache targets

The length of the information returned for the Cache targets entry.

Length of nack targets

The length of the information returned for the Nack targets entry.

Nack targets

The list of acknowledgment target descriptions referenced by the new trigger handler description and used, at server startup, to obtain information as to where failed process completion messages are sent. Descriptions must be listed by name, where each name is separated by one or more spaces, left justified, and padded with blanks if necessary.

Note: If a null list is specified, Offset to nack targets must equal 0 (QZHT_NONE), indicating no acknowledgment target descriptions are currently referenced. Messages concerning failed trigger processes, referencing this handler, are sent to all listed acknowledgment targets.

Number of threads

The number of concurrent threads the triggered cache manager server spawns when processing triggers that are sent to this trigger handler. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9).

Offset to ack targets

The offset from the beginning of the receiver variable to the Ack targets data, in bytes. The value must be greater than 0, or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NONE: A null list is specified for Ack targets indicating no acknowledgment target descriptions are currently referenced.
---	--

Offset to cache targets

The offset from the beginning of the receiver variable to the Cache targets data, in bytes. The value must be greater than 0, or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NONE: A null list is specified for Cache targets indicating no cache target descriptions are currently referenced.
---	---

Offset to nack targets

The offset from the beginning of the receiver variable to the Nack targets data, in bytes. The value must be greater than 0, or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NONE: A null list is specified for Nack targets indicating no acknowledgment target descriptions are currently referenced.
---	---

Server name

The name used to identify the triggered cache manager server for which the description is associated (left justified and padded with blanks if necessary).

Trigger handler description name

The name of the trigger handler description (left justified and padded with blanks if necessary).

Trigger handler type

The trigger handler description type (left justified and padded with blanks if necessary). The value must be one of the special values described below.

Special values and their meanings are as follows:

- *UPDATE
QZHT_UPDATE_TYPE
- *PUBLISH
QZHT_PUBLISH_TYPE

THDG0200 format:

Offset		Type	Field
Dec	Hex		
0	0		Everything from THDP0100 format
144	90	Binary(4)	Cache request queue priority
148	94	Binary(4)	Trigger queue collapse policy

THDG0200 format field description:

Cache request queue priority

Specifies the trigger handler priority value when submitting requests to the cache targets. Lower values indicate higher priority. The value must be greater than 0 and less than 2^{31} (or 2.147×10^9).

Note: Triggered cache manager servers queue the trigger handler requests to the cache targets and process them according to queue priority. Requests from trigger handlers with higher priority are processed before requests from trigger handlers with lower priority. The trigger handler queue priority can be changed while servers are active by using the *-chspriority* command in a trigger message.

Trigger queue collapse policy

Specifies if identical triggers waiting on the request queue, for this trigger handler, are collapsed. The value must equal one of the special values described below.

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: Identical triggers, waiting on the request queue for this trigger, are collapsed.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: Identical triggers are not collapsed.

Note: Only triggers using the *-objects* keyword can be collapsed. Identical triggers are those having an identical set of listed objects. The order of the listed objects is not important. Once a trigger handler begins processing a trigger, it is not collapsed.

THDG0300 format:

Offset		Type	Field
Dec	Hex		
0	0		Everything from THDP0100 format
14	90	Char(32)	Object dependency graph
176	B0	Char(32)	Rule set
208	D0	Binary(4)	Offset to traversal edge name
212	D4	Binary(4)	Length of traversal edge name
216	D8	Binary(4)	Offset to default included file
220	DC	Binary(4)	Length of default included file

224	E0	Binary(4)	Include dependency information
228	E4	Binary(4)	Include triggered file information
22C	E8	Binary(4)	Include cached file information
		Char(*)	Traversal edge type
		Char(*)	Default included file

THDG0300 format field descriptions:

Default included file

The name of a file the trigger handler includes, by global default, as a replacement for included files that have not been triggered (when a local default file is not specified or available). The file name must be left justified and padded with blanks if necessary.

Note: If the value is null, Offset to default included file must equal 0 (QZHT_NONE), indicating that a global default file name is not specified for this trigger handler description. The file specified as a global default must be triggered (at runtime) before it can be used.

Include cached file information

Specifies if a list of names for all files sent to cache targets, as a result of handling original trigger requests, is included in successful process completion messages. The value must equal one of the special values described below.

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: A list of names is included.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: A list of names is not included.

Include dependency information

Specifies if information concerning all dependent files, assembled into triggered files as a result of handling original trigger request, is included in successful process completion messages. The value must equal one of the special values described below.

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: Information is included.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: Information is not included.

Include triggered file information

Specifies if a list of names for all files triggered, as a result of handling original trigger requests, is included in successful process completion messages. The value must equal one of the special values described below.

Special values and their meanings are as follows:

1	QZHT_YES, QZHT_TRUE, QZHT_ENABLED: A list of names is included.
0	QZHT_NO, QZHT_FALSE, QZHT_DISABLED: A list of names is not included.

Length of default included file

The length of information for the Default included file entry.

Length of traversal edge type

The length of the information returned for the Traversal edge type entry.

Object dependency graph

The name of an object dependency graph description referenced by the trigger handler description and used, at server startup, to identify which object dependency graph is used by the handler to record and obtain object dependency information. The value must be an object dependency graph description name (left justified and padded with blanks if necessary).

Offset to default included file

The offset from the beginning of the request variable to the Default included file data, in bytes. The value must be greater than 0, or equal to one of the special values described below.

Special values and their meanings are as follows:

0	QZHT_NONE: The value is null for Default included file.
---	---

Offset to traversal edge type

The offset from the beginning of the request variable to the Traversal edge type data, in bytes.

Rule set

The name of a rule set referenced by the trigger handler description and used, at server startup, to identify which publishing rules are used by the handler. The value must be a rule set name, or one of the special values described below (left justified and padded with blanks if necessary).

Special values and their meanings are as follows:

*NONE

QZHT_NONE_CHAR: No rule set is referenced by the trigger handler.

Note: Triggers that specify files that do not match any publishing rules, in the specified rule set, are processed according to the default publishing rule. That is, they are read from the data source and sent to all cache targets (they are not parsed). An empty rule set has the same affect as not specifying a rule set at all.

Traversal edge type

The name of an object dependency graph (ODG) edge type used by the trigger handler to determine object dependencies when assembling files. The edge type name must be left justified and padded with blanks if necessary.

Note: Only edges of the specified type are traversed by this trigger handler to determine object dependencies.

Error messages:

TCM7001 E

User &1 is not authorized.

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7032 E

Receiver variable format is not valid.

TCM7034 E

Length of receiver variable is not valid for the specified format.

TCM7035 E

Server parameter is not valid.

TCM7036 E

Description parameter is not valid.

TCM703E E

Error code parameter is not valid

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 E

A configuration file for triggered cache manager server &1 was not found.

TCM7290 E

A &1 and &2 was not found for triggered cache manager server &3.

Start Triggered Cache Manager Server (QzhtStrTCMServer) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Required Parameter Group:

1 request variable	Input	Char(*)
2 length of request variable	Input	Binary(4)
3 request variable format	Input	Char(8)
4 error code	I/O	Char(*)

Library Name/Service Program: QTCM/QZHTINOPS

Threadsafe: Yes

Use the *QzhtStrTCMServer* API to start the triggered cache manager servers. The API is a callable service implemented as an ILE entry point within the QZHTINOPS *SVRPGM in the QTCM *LIB.

The QTCM/H(QZHTINCONF) header file includes ILE C prototypes for this API.

Authorities and locks: None.

Required parameter group:**request variable**

INPUT: CHAR(*)

The variable used to pass information used to start existing triggered cache manager servers.

length of request variable

INPUT: BINARY(4)

The number of bytes that the calling program provides for Request variable.

request variable format

INPUT: CHAR(8)

The format name of *Request variable* data. The following values must be used:

- INNP0100: Basic server name format

error code

I/O:CHAR(*)

The structure in which to return error information.

INNP0100 format:

Offset		Type	Field
Dec	Hex		
0	0	Char(32)	Server name

Field description:

Server name

The name used to identify which triggered cache manager server is started (left justified and padded with blanks if necessary). The value must specify an server name, or be one of the special values described below.

Special values and their meanings are as follows:

***AUTOSTART**

QZHT_AUTOSTART_CHAR: All triggered cache manager servers having the autostart property set to '1' (QZHT_YES, QZHT_TRUE, QZHT_ENABLED) are started.

***ALL** QZHT_ALL_CHAR: All triggered cache manager servers are started.

Error messages:

TCM7030 E

Severe error occurred while addressing parameter list.

TCM7031 E

Request variable format is not valid.

TCM7033 E

Length of request variable is not valid for the format specified.

TCM703E E

Error code parameter is not valid

TCM70F0 E

Unknown error occurred while processing request.

TCM7101 E

User QTCM is not authorized to the configuration file for triggered cache manager server &1.

TCM7190 E

A configuration file for the triggered cache manager server &1 was not found.

TCM7390 E

Triggered cache manager server &1 not found.

TCM74C0 E

Triggered cache manager server name is not valid.

TCM73C3 E

Triggered cache manager server &1 is active.

Add Config Object (QzuiAddConfigObject) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Note: This API is for HTTP Server (powered by Apache)

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

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.

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 Δ where Δ , 3 = before Δ where Δ , 4 = at start of scope specified by Δ where Δ , 5 = at end of scope specified by Δ where Δ).

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:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Note: This API is for HTTP Server (powered by Apache)

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

Use the *QzuiChangeConfigObject* API to change the value portion of a scope or directive. The value is considered anything after the keyword. For example, in the directive Δ BrowserMatch Mozilla/2 nokeepalive Δ , the keyword is Δ BrowserMatch Δ and the value is Δ Mozilla/2 nokeepalive Δ .

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:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Note: This API is for HTTP Server (powered by Apache)

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

Use the *QzuiChangeInstanceData* API to change the information contained in the instance file. The idata information is retrieved in the format specified by INSD0110.

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 date.

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
10	Binary(4)	Threads
14	Binary(4)	CCSID
18	Char(10)	Outgoing table name
28	Char(10)	Outgoing table library
38	Char(10)	Incoming table name
48	Char(10)	Incoming table library
58	Char(512)	Config file (full path)
570	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 (powered by Apache) type instance.

Close Apache Config File (QzuiCloseConfig) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Note: This API is for HTTP Server (powered by Apache)

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

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 filename is specified, the configuration is written to that file, otherwise it is written to the original file.

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:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Note: This API is for HTTP Server (powered by Apache)

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

Use the *QzuiCreateInstance* API to create a new server instance file for the parameters passed to it.

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 431 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.

Find Config Object (QzuiFindConfigObject) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Note: This API is for HTTP Server (powered by Apache)

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

Use the *QzuiFindConfigObject* API to search HTTP Server (powered by Apache) configuration file for the object (and possibly value) specified and returns a handle to it. If `start` 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 `BrowserMatch` and the value is `Mozilla/2` the search would find `BrowserMatch Mozilla/2 nokeepalive`. Also, the `val` field would contain `Mozilla/2 nokeepalive`. You need only pass the object type and keyword. For example, to find a `Port` directive, set the object type to 1, the keyword to `Port` and `fdata_size` to 8. If the value field is not needed, set `val_size` 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 powered by Apache 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 &1 not valid.

CPF3C21 E

Format name &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:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Note: This API is for HTTP Server (powered by Apache)

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

Use the *QzuiGetInstanceData* API to retrieve the data from a specified server instance file. The data information is returned in the format specified by INSD0110. See “INSD0110 format” on page 431 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 431 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 &1 not valid.

CPF3C21 E

Format name &1 not valid.

CPF3CF1 E

Error code parameter not valid.

HTPA001 E

Input parameter &1 not valid.

HTPA101 E

Server instance &1 not found or is unreadable.

HTPA127 E

Server instance &1 is not a HTTP Server (powered by Apache) type instance.

Get Server Instance Names (QzuiGetInstanceNames) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Note: This API is for HTTP Server (powered by Apache)

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

Use the *QzuiGetInstanceNames* API to return a list of instance names, the type of instance and the running status of the instance.

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 *QzuiGetInstanceNames* HTTP Server (powered by Apache) API.

Offset	Type	Field
0	Char(10)	Instance name
10	Char(2)	Reserved
14	Binary(4)	Running status
18	Binary(4)	Instance type (0 = HTTP Server (original), 1 = HTTP Server (powered by Apache))

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.

CPF3C21 E
Format name &1 not valid.

HTPA001 E
Input parameter &1 not valid.

Get Instance Type (QzuiGetInstanceType) API:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Note: This API is for HTTP Server (powered by Apache)

Required Parameter Group:

1	name	Input	Char(10)
2	itype	Output	Binary(4)
3	errcode	I/O	Char(*)

Threadsafe: Yes

Use the *QzuiGetInstanceType* API to return the type of instance the specified instance corresponds to. If the file is not a valid instance file, a -1 is returned.

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:BINAR(4)

The type of instance (-1 = Invalid, 0 = Original, 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:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Note: This API is for HTTP Server (powered by Apache)

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

Use the *QzuiOpenConfig* API to cause the configuration file specified to be read into memory and a handle is returned. This handle is used in subsequent API calls to manipulate directives and scopes within the file.

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:

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Note: This API is for HTTP Server (powered by Apache)

Required Parameter Group:

1	cfg	Input	Binary(4)
2	object	Input	Binary(4)
3	errcode	I/O	Char(*)

Threadsafe: Yes

Use the *QzuiRemoveConfigObject* API to remove the specified directive or scope from the configuration. If a scope is removed, all the directives within it are removed also.

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.

Compatibility with other APIs

This topic provides information about API compatibility. The Server API is compatible with other APIs, such as CGI. You can run your existing CGI programs on all the server's operating systems.

Porting CGI programs: Here are a few guidelines for porting CGI applications that are written in ILE C to use the Server API:

- 1.
2. Remove your main() entry point or rename it so you can build a service program.
3. Eliminate global variables or protect them with a mutual exclusion semaphore.
4. Change the following calls in your programs:
 - v Change printf() header calls to HTTPD_set().
 - v Change printf() data calls to HTTPD_write().
 - v Change getenv() calls to HTTPD_extract().

Note: This returns deallocated memory, so you must free the result.

5. Remember, that the server runs in a multi-threaded environment and your application functions must be threadsafe. If the functions are re-entrant, performance will not decrease.
6. Do not forget to set the Content-Type header if you are using HTTPD_write() to send data back to the client.
7. Check your code for memory leaks.
8. Think about your error paths. If you generate error messages yourself and send them back as HTML, you should return HTTPD_OK from your service functions.

Authentication and Authorization: Authentication is the verification of the security tokens that are associated with this request. Authorization is the process using the security tokens to determine if the requester has access to the resource. In HTTP Server, authentication is part of the authorization process; it occurs only when authorization is required.

If your Server API application provides its own authorization process, it will override the default server authorization and authentication. Therefore, if you have Authorization directives in your configuration file, the application functions associated with them must also handle any necessary authentication. The predefined HTTPD_authenticate() function is provided to assist you with this.

There are three ways you can provide for authentication in your authorization application functions:

- Write your own separate authorization and authentication application functions. In your configuration file, use both the Authorization and the Authentication directives to specify these functions. Be sure to include HTTPD_authenticate() in your authorization application function.

When the Authorization step is run, it performs your authorization application function which, in turn, calls your authentication application function.

- Write your own authorization application function but have it call the default server authentication. In your configuration file, use the Authorization directive to specify your function. In this case, you will not need the Authenticate directive. Be sure to include HTTPD_authenticate() in your authorization application function.

When the Authorization step is run, it performs your authorization application function which, in turn, calls the default server authentication.

- Write your own authorization application function and include all your authentication processing right into it. Do not use HTTPD_authenticate() in your authorization application function. In your configuration file, use the Authorization directive to specify your function. In this case, you will not need the Authentication directive.

When the Authentication step is run, it performs your authorization application function and any authentication it included.

If your Server API application does not provide its own authorization process, you can still provide customized authentication. To do this, write your own authentication application function. In your configuration file, use the Authentication directives to specify your function. In this case, you do not need the Authorization directive.

Notes:

- 1.

2. If you do not have any Authorization directives in your configuration file, or their specified application functions decline to handle the request, the server's default authorization will occur.
3. If you do have Authorization directives in your configuration file and their application functions include HTTPD_authenticate(), the server calls any authentication functions specified in the Authentication directives. If you do not have any Authentication directives defined, or their specified application functions decline to handle the request, the server's default authentication will occur.
4. If you do have Authorization directives in your configuration file but their application functions do not include HTTPD_authenticate(), no authentication functions will be called by the server. You must code your own authentication processing as part of your authorization application functions or make your own calls to other authentication modules.
5. HTTP Server automatically generates the challenge (by prompting the browser to return user ID and password) if you return 401 or 407 from your authorization exit. However, you must still configure a protection setup so that this will occur correctly.

Environment variables: You can use environment variables in the predefined functions HTTPD_extract() and HTTPD_set(). They represent values you can extract from a client request or values you can set or create when processing a client request. The supported environment variables for HTTP Server are can be found in the IBM iSeries Information Center. The environment variables have been divided into two groups: Non-SSL and SSL and by HTTP Server type.

Overview of the Server API

This topic provides an overview of the Server API. The Server API allows you to extend the base functions of HTTP Server (original).

You can write extensions to do customized processing, such as:

- Enhance the basic authentication or replace it with a site-specific process.
- Add error handling routines to track problems or alert for serious conditions.
- Detect and track information that comes in from the requesting client, such as server referrals and user agent code.

General procedure for writing Server API programs: Before you start writing your Server API programs, you need to understand how the HTTP Server works. The server performs a sequence of steps for each client request that it processes. Your program can run and make function calls at any of these steps. You need to decide where in the basic server request process you want to add customized functions. For example, do you want the server to do something after it reads a client request but before it performs any other processing? Or, maybe you want the server to perform special routines during authentication and then after it sends the requested file.

You can instruct the server to call the application functions in your program at the appropriate processing step. You can do this by using the API directives in your server configuration file.

Guidelines: Use the following guidelines when creating Server API programs:

- If compiling on OS/400 with Integrated Language Environment (ILE) C, ensure that you include QHTTSPVR in the library list.
- Give each of your application functions a unique function name and call the server predefined functions as needed. Be sure to include HTAPI.h and to use the HTTPD_LINKAGE macro in your function definitions to avoid abending the server. This macro ensures that all functions use the same calling conventions.
- The server runs in a multi-threaded environment; therefore, your application functions must be threadsafe. If your application is re-entrant, performance will not decrease.
- Keep the actions in your applications to a thread scope. Do not perform any actions as a process scope, such as exit or change user ID.

- Eliminate global variables or protect them with a mutual exclusion semaphore.
- Do not forget to set the Content-Type header if you are using HTTPD_write() to send data back to the client.
- You also must take into consideration the Content-Encoding header when you use HTTPD_write() to send data back to the client.
- Always check your return codes and provide conditional processing where necessary.
- Compile and then create your service program by using CRTSRVPGM. When using CRTSRVPGM, you must bind to QZHBIAPI *SRVPGM that is in the QHTTPSVR library.
- Add Server API directives to your configuration file so that you can associate your program's application functions with the appropriate step. There is a separate directive for each server request processing step. You must stop and restart the server for the new directives take effect.
- Test your program rigorously. HTTP Server is a threaded server requiring more rigorous testing than you would for a forked server. HTTP Server calls your program directly making the server and program run in the same process space. Any errors in your program can stop the server.

Basic server request process: You can break down the basic server request process into a number of steps, based on the type of processing the server is performing during that phase. Each step includes a juncture at which a specified part of your program can run. The Server API directives in your configuration file, indicate which of your application functions you want called during a request process step.

Your compiled program exists as a service program. As the server proceeds through its request process steps, it calls the application functions associated with each step, until one of the functions indicates it has handled the request. If you have more than one of your application functions indicated for a particular step, your functions are called in the order in which they appear in the configuration file.

If the request is not handled by an application function (either you did not include a Server API directive or your application function for that step returned HTTP_NOACTION), the server performs the default action for that step.

Note: This is true for all steps except the Service step; the Service step does not have a default action.

The following list indicates the purpose of each step and defines the processing order.

Server Initialization

Performs initialization functions before any client requests are read.

PreExit

Performs processing after a request is read but before anything else is done. If this step returns an indication that the request was processed (HTTP_OK), only the Data Filter, Log and PostExit steps, in the request process are performed.

Authentication

Decodes, verifies, and stores security tokens.

Name Translation

Translates the virtual path (from URL) to the physical path.

Authorization

Uses stored security tokens to check the physical path (protections, acls) and generates the WWW-Authenticate headers required for basic authentication. If you write your own application function to replace this step, you must generate these headers yourself.

Object Type

Locates the file system object that is indicated by the path.

Service

Satisfies the request (such as, send the file or run the CGI)

Data Filter

Gives write access to the outgoing data stream.

Log Allows transaction logging.

Error Allows customized responses to error conditions.

PostExit

Allows cleanup of resources that are allocated for request processing.

Server Termination

Allows cleanup processing when an orderly shutdown or restart occurs.

Application functions: Use the following function prototype syntax to write your own program functions for the defined request steps.

Each of your functions must fill in the return code parameter with a value that indicates the action that is taken.

- HTTP_NOACTION (value of 0) means no action that is taken.
- Otherwise, one of the valid HTTP return codes is expected, indicating that the application function handled the step. As a result, no other application functions are called to handle that step of this request.

The function prototypes for each request step show the format to use and explain the type of processing they can perform. You must give your functions unique names and can choose your own naming conventions. For ease of association, these names relate to the server's request processing steps.

Server Initialization

```
void
HTTPD_LINKAGE ServerInit(
    unsigned char *handle, unsigned long *major_version,
    unsigned long *minor_version, long *return_code);
```

This function is called once when your module is loaded during server initialization. This is your opportunity to perform initialization before any requests have been accepted. Although all server initialization functions are called, error return codes from this step cause the server to ignore all other functions that are configured in this program.

PreExit

```
void
HTTPD_LINKAGE PreExit(
    unsigned char *handle, long *return_code);
```

This function is called after the request is read, but before any processing has occurred.

All server-predefined functions are valid during this step.

Authentication

```
void
HTTPD_LINKAGE Authentication(
    unsigned char *handle, long *return_code);
```

This function allows user verification of the security tokens. This step is performed based on the authentication scheme.

Only HTTP_extract() and HTTPD_set() are valid during this step.

Name Translation

```
void
HTTPD_LINKAGE NameTrans(
    unsigned char *handle, long *return_code);
```

This function provides a mechanism for mapping URLs to objects.

Only HTTPD_extract() and HTTPD_set() are valid during this step.

Authorization

```
void
HTTPD_LINKAGE Authorization(
    unsigned char *handle, long *return_code);
```

This function verifies that the identified object may be returned to the client. If you are doing basic authentication, you must generate the required WWW-Authentication headers.

Only HTTPD_extract() and HTTPD_set() are valid functions during this step.

Object Type

```
void
HTTPD_LINKAGE ObjType(
    unsigned char *handle, long *return_code);
```

This step checks to see if the object exists and performs object typing.

Only HTTPD_extract() and HTTPD_set() are valid during this step.

Service

```
void
HTTPD_LINKAGE Service(
    unsigned char *handle, long *return_code);
```

This function satisfies the request, if not satisfied in the PreExit.

All server-predefined functions are valid during this step. Refer to the Enable directive in the HTTP Server for iSeries Webmaster's Guide, GC41-5435 for information.

Data Filter

Filters data as a stream class, which means that each of its functions acts like a segment of pipe down which data flows. For this step, you must use three application functions:

```
void
HTTPD_LINKAGE open(
    unsigned char *handle, long *return_code);
```

This function performs any initialization (such as buffer allocation) that is required to process the data for this stream. An error return code causes this filter to abort.

```
void
HTTPD_LINKAGE write(
    unsigned char *handle, unsigned char *data,
    unsigned long *length, long *return_code);
```

This function processes the data and calls the server's write function with the new or changed data. The application must not attempt to free the buffer passed to it nor expect the server to free the buffer it receives.

```
void
HTTPD_LINKAGE close(
    unsigned char *handle, long *return_code);
```

This function performs any cleanup (such as flushing and freeing the buffer) required to complete processing the data for this stream.

Log

```
void
HTTPD_LINKAGE Log(
    unsigned char *handle, long *return_code);
```

This function is called after each request is processed and the communication to the client is closed, regardless of the success or failure of the request processing. Only HTTPD_extract() and HTTPD_set() are valid during this step.

Error

```
void
HTTPD_LINKAGE Error(
    unsigned char *handle, long *return_code);
```

This function is called only when an error is encountered and provides an opportunity to customize the response.

PostExit

```
void
HTTPD_LINKAGE PostExit(
    unsigned char *handle, long *return_code);
```

This function is called, regardless of the success or failure of the request, so that you can clean up any resources that are allocated by your application to process the request.

Server Termination

```
void
HTTPD_LINKAGE ServerTerm(
    unsigned char *handle, long *return_code);
```

This function is processed when an orderly shutdown or restart of the server occurs. It allows you to clean up resources that are allocated during the Server Initialization step. Do not call any HTTP_* functions in this step (the results are unpredictable). If you have more than one Server API directive in your configuration file for Server Termination, they will all be called.


HTTP return codes and values: These return codes follow the HTTP specification that is published by the World Wide Web Consortium at URL: <http://www.w3.org/pub/WWW/Protocols/>  . Your application functions should return one of these values.

Table 11. Return codes

Value	Return Code
0	HTTP_NOACTION
100	HTTP_CONTINUE
101	HTTP_SWITCHING_PROTOCOLS
200	HTTP_OK
201	HTTP_CREATED
202	HTTP_ACCEPTED
203	HTTP_NON_AUTHORITATIVE
204	HTTP_NO_CONTENT
205	HTTP_RESET_CONTENT
206	HTTP_PARTIAL_CONTENT
300	HTTP_MULTIPLE_CHOICES
301	HTTP_MOVED_PERMANENTLY
302	HTTP_MOVED_TEMPORARILY
303	HTTP_SEE_OTHER
304	HTTP_NOT_MODIFIED
305	HTTP_USE_PROXY

Table 11. Return codes (continued)

Value	Return Code
400	HTTP_BAD_REQUEST
401	HTTP_UNAUTHORIZED
403	HTTP_FORBIDDEN
404	HTTP_NO_FOUND
405	HTTP_METHOD_NOT_ALLOWED
406	HTTP_NOT_ACCEPTABLE
407	HTTP_PROXY_UNAUTHORIZED
408	HTTP_REQUEST_TIMEOUT
409	HTTP_CONFLICT
410	HTTP_GONE
411	HTTP_LENGTH_REQUIRED
412	HTTP_PRECONDITION_FAILED
413	HTTP_ENTITY_TOO_LARGE
414	HTTP_URI_TOO_LONG
415	HTTP_BAD_MEDIA_TYPE
500	HTTP_SERVER_ERROR
501	HTTP_NOT_IMPLEMENTED
502	HTTP_BAD_GATEWAY
503	HTTP_SERVICE_UNAVAILABLE
504	HTTP_GATEWAY_TIMEOUT
505	HTTP_BAD_VERSION

Predefined functions and macros: You can call the server's predefined functions and macros from your own application functions. You must use their predefined names and follow the format that is described in this section. Note that the parameter descriptions use the letter Δ to indicate input, the letter ∇ to indicate output, and Δ/∇ to indicate that a parameter is both input and output.

Each of these functions returns one of the HTTPD return codes, depending on the success of the request.

HTTPD_authenticate()

Authenticates a user ID and password. Valid only in PreExit, Authenticate, and Authorization steps.

```
void
HTTPD_authenticate(
    unsigned char *handle, /* i; handle */
    long *return_code); /* o; return code */
```

HTTPD_attributes

Extracts the attributes of a file. Valid in all steps.

The name of the file and the buffer containing the attributes are in the default CCSID of the job.

```
void
HTTPD_attributes(
    unsigned char *handle, /* i; handle (NULL right now) */
    unsigned char *name, /* i; name of the file */
    unsigned long *name_length, /* i; length of the name */
    unsigned char *value, /* o; buffer containing attributes*/
    unsigned long *value_length, /* i/o; size of buffer/length of attributes*/
    long *return_code); /* o; return code */
```

HTTPD_extract()

Extracts the value of a variable associated with this request. The valid variables you can use for the name parameter are the same as those used in the CGI. This function is valid in all steps, however not all variables are.

The CCSID of the name of the value to extract and the buffer in which to put the value depends upon the step and the CGI mode. For all steps except Service, these parameters are in the default CCSID of the job. For the Service step, the CGI mode determines the CCSID. For the %%MIXED%% CGI mode, these fields are in EBCDIC CCSID 37. For all other CGI modes, these fields are in the default CCSID of the job.

```
void
HTTPD_extract(
    unsigned char *handle,      /* i; handle */
    unsigned char *name,       /* i; name of value to extract */
    unsigned long *name_length, /* i; length of the name */
    unsigned char *value,      /* o; buffer in which to put value */
    unsigned long *value_length, /* i/o; buffer size/length of value */
    long *return_code);       /* o; return code */
```

If this function returns the HTTPD_BUFFER_TOO_SMALL return code, the buffer size you requested was not big enough for the extracted value. In this case, the function does not fill in the buffer but does update value_length with the buffer size you would need in order to successfully extract this value. Retry the extract with a buffer that is at least as big as the returned value_length.

Note: Server API programs gain access to information about a particular HTTP request by examining predefined variables. This information is obtained from the server using the HTTPD_extract() function. The PASSWORD variable used when basic authentication is requested. This password is associated with a user that is defined in a validation list (*VLDL) or the password for a user profile on the server. Encryption may not be the same for those VLDL's that must do two-way encryption and are in QUSRSYS. The two cannot be interchanged.

HTTP Server does not allow access to the PASSWORD variable if authorization is configured which uses user profiles and passwords for authentication.

To prevent an application from obtaining a user profile password, HTTPD_extract() is sensitive to the type of protect setups that are currently configured. If a protection setup is configured with a password file of %%SYSTEM%% (protection requiring user profile password), HTTP_extract() for PASSWORD returns HTTP_PARAMETER_ERROR and sets the value parameter to *CONFLICT. Otherwise, HTTP_extract() returns the appropriate value.

HTTPD_reverse_translate()

Translates a file system path to a URL. Valid in all steps.

The name of the file system object and the buffer containing the URL are in the default CCSID of the job.

```
void
HTTPD_reverse_translate(
    unsigned char *handle,      /* i; handle (NULL right now) */
    unsigned char *name,       /* i; name of the file system object */
    unsigned long *name_length, /* i; length of the name */
    unsigned char *value,      /* o; buffer which contains the URL */
    unsigned long *value_length, /* i/o; size of buffer/length of URL */
    long *return_code);       /* o; return code */
```

HTTPD_translate()

Translates a URL to a file system path. Valid in all steps.

Translates a URL to a file system path. Valid in all steps. The CCSID for QUERY_STRING depends upon the step and the CGI mode. For all steps except Service, these parameters are in the default CCSID of the job. For the Service step, the CGI mode determines the CCSID. For the %%MIXED%% CGI mode, these fields are in EBCDIC CCSID 37. For all other CGI modes, these fields are in the default CCSID of the job. The CCSID for QUERY_STRING depends upon the step and the CGI mode. For all steps except Service, these parameters are in the default CCSID of the job. For the Service step, the CGI mode determines the CCSID. For the %%MIXED%% CGI mode, these fields are in EBCDIC CCSID 37. For all other CGI modes, these fields are in the default CCSID of the job.

```
void
HTTPD_translate(
    unsigned char *handle,          /* i; handle (NULL right now) */
    unsigned char *name,           /* i; name of the URL */
    unsigned long *name_length,    /* i; length of the name */
    unsigned char *url_value,      /* o; buffer containing translated URL */
    unsigned long *url_value_length, /* i/o; buffer size/length of translated URL */
    unsigned char *path_trans,     /* o; buffer containing PATH_TRANSLATED */
    unsigned long *path_trans_length, /* i/o; size of buffer/length of PATH_TRANSLATED */
    unsigned char *query_string,   /* o; buffer containing QUERY_STRING */
    unsigned long *query_string_length, /* i/o; size of buffer/length of QUERY_STRING */
    long *return_code);           /* o; return code */
```

HTTPD_set()

Sets the value of a variable associated with this request. The valid variables you can use for the name parameter are the same as those are used by the CGI program.

Note that you can also create variables with this function. If any variables you create are prefixed by Δ HTTP_ Δ , they are sent as headers in the response, without the Δ HTTP_ Δ prefix. For example, if you want to see a Location header, use HTTPD_set() with the variable name HTTP_LOCATION.

This function is valid in all steps, however not all variables are.

The CCSID of the name of the value to set and the buffer which contains the value depends upon the step and the CGI mode. For all steps except Service, these parameters are in the default CCSID of the job. For the Service step, if you are not setting a variable with the Δ HTTP_ Δ prefix, then the default CCSID of the job is used. For setting variables prefixed by Δ HTTP_ Δ in the Service step, the CGI output mode determines the CCSID to be used. For %%MIXED%% mode, both are in EBCDIC 37. For other CGI modes, they are in the default CCSID of the job.

```
void
HTTPD_set(
    unsigned char *handle,          /* i; handle */
    unsigned char *name,           /* i; name of the value to set */
    unsigned long *name_length,    /* i; length of the name */
    unsigned char *value,         /* i; buffer containing the value */
    unsigned long *value_length,   /* i; length of value */
    long *return_code);           /* o; return code */
```

HTTPD_file()

Sends a file to satisfy this request. Valid only in PreExit, Service, NameTrans, Error, and DataFilter steps.

The name of the file to send is in the default CCSID of the job.

```
void
HTTPD_file(
    unsigned char *handle,          /* i; handle */
    unsigned char *name,           /* i; name of file to send */
    unsigned long *name_length,    /* i; length of the name */
    long *return_code);           /* o; return code */
```

HTTPD_exec()

Runs a script to satisfy this request. Valid in PreExit, Service, NameTrans, and Error steps.

The name of the script to run is in the default CCSID of the job.

```
void
HTTPD_LINKAGE
HTTPD_exec(
    unsigned char *handle,      /* i; handle */
    unsigned char *name,       /* i; name of script to run */
    unsigned long *name_length, /* i; length of the name */
    long *return_code);       /* o; return code */
```

HTTPD_read()

This function reads the body of the client's request. It uses HTTPD_extract for headers. This function is valid only in the PreExit, Service, and Data Filter steps.

For the Service step, the data CCSID is determined by the CGI mode. For %%MIXED%% the data is in the default job CCSID and any encoded sequences are the EBCDIC representation of the ASCII character. For %%EBCDIC%% and %%EBCDIC_JCD%% modes, the data is in the default CCSID of the job (including the escape sequences). For %%BINARY%% mode, no conversion is performed. For all other steps, this is the default job CCSID.

```
void
HTTPD_read(
    unsigned char *handle,      /* i; handle */
    unsigned char *value,      /* i; buffer in which to place data */
    unsigned long *value_length, /* i/o; buffer size/length of data */
    long *return_code);       /* o; return code */
```

HTTPD_write()

Writes the body of the response. Uses HTTPD_set and HTTPD_extract for headers. Valid in the PreExit, Service, NameTrans, Error, and Data Filter steps.

If you do not use HTTPD_set() to set the content type before the first time you call this function, the server assumes you are sending a CGI data stream.

You may need to use HTTPD_set() to set the CGI environment variable CONTENT_ENCODING to the appropriate code page for the content of your response before you send the data to the client.

For the Service step, the data to send is in the default job CCSID for %%MIXED%% and %%EBCDIC%% CGI output modes unless overridden by a character CCSID tag on the content_type header. The data is %%BINARY%% mode, so it is assumed to be binary.

```
void
HTTPD_write(
    unsigned char *handle,      /* i; handle */
    unsigned char *value,      /* i; data to send */
    unsigned long *value_length, /* i; length of the data */
    long *return_code);       /* o; return code */
```

HTTPD_log_error()

Writes a string to the server's error log.

The string passed to HTTPD_log_error must be EBCDIC (CCSID 37) data. The server converts the string to the CCSID of the error log file.

```
void
HTTPD_LINKAGE
HTTPD_log_error(
    unsigned char *handle, /* i; handle */
    unsigned char *value, /* i; data to write */
    unsigned long *value_length, /* i; length of the data */
    long *return_code); /* o; return code */
```

HTTPD_log_trace()

Writes a string to the server's trace log.

The string passed to HTTPD_log_trace must be EBCDIC (CCSID 37) data.

```

void
HTTPD_LINKAGE
HTTPD_log_trace
    unsigned char *handle,      /* i; handle (NULL right now) */
    unsigned char *value,      /* i; data to write */
    unsigned long *value_length, /* i; length of the data */
    long *return_code);        /* o; return code */

```

HTTPD_restart()

Restarts the server after all active requests have been processed. Valid only in ServerInit and ServerTerm steps.

```

void
HTTPD_restart(
    long *return_code); /* o; return code */

```

HTTPD_proxy()

Makes a proxy request. Valid in PreExit and Service steps.

Note: This is a completion function; the response is complete after this function.

The name of the URL for the proxy request and the body of the request are in the default CCSID of the job.

```

void
HTTPD_LINKAGE
HTTPD_proxy(
    unsigned char *handle,      /* i; handle (NULL right now) */
    unsigned char *url_name,    /* i; url for the proxy request */
    unsigned long *name_length, /* i; length of the url */
    unsigned char *request_body, /* i; body of the request */
    unsigned long *body_length, /* i; length of the body */
    long *return_code);        /* o; return code */

```

Note: Once an HTTPD_ function returns, it is safe for you to free any memory you passed with it.

Return codes: The server fills in the return code parameter with one of these values depending on the success of the request.

Table 12. Return code parameters

Return code parameter	Description
-1	HTTPD_UNSUPPORTED The function is not supported.
0	HTTPD_SUCCESS The function succeeded, and the output fields are valid.
1	HTTPD_FAILURE The function failed.
2	HTTPD_INTERNAL_ERROR The function encountered an internal error and cannot continue processing this request.
3	HTTPD_PARAMETER_ERROR You may have added one or more incorrect parameters. For example, the variable you tried to extract is unknown.
4	HTTPD_STATE_CHECK The function is not valid in this step.

Table 12. Return code parameters (continued)

Return code parameter	Description
5	HTTPD_READ_ONLY Returned only by HTTP_set(). The application can not set the variable.
6	HTTPD_BUFFER_TOO_SMALL Returned only by HTTPD_extract(). The provided buffer was too small.
7	HTTPD_AUTHENTICATION_FAILED Returned only by HTTPD_authenticate(). Examine the HTTP_RESPONSE and HTTP_REASON variables for more information.
9	HTTPD_EOF Returned only by HTTPD_read(). This return code indicates the end of the request body.
9	HTTPD_ABORT_REQUEST The request has been aborted because the client has provided an entity that did not match the condition that is specified by the request.
10	HTTPD_REQUEST_SERVICED Returned by HTTP_proxy. This return code indicates that the program that called the function completed the response for this request.
11	HTTPD_RESPONSE_ALREADY_COMPLETED The function failed because the response for that request completed processing.

Server API configuration directives

This topic provides information about the Server API. Each step in the request process has a configuration directive that allows you to indicate which of your application functions you want called and run during that step. You can add these directives to your server's configuration file by manually editing it or by using the Server API Request Processing form in the IBM Web Administration for iSeries interface.

Server API usage notes: Here are some notes to consider when using the Server API:

- When appropriate, you can indicate that you want your application function called for all URL requests or only for URL requests that match a specified mask.
- You can also have your Authentication functions called for every request or just for those with a type of Basic.
- The Server API directives, except for the Service and NameTrans directives, can be in any order in the configuration file. You do not need to include every Server API directive. If you do not have an application function for a particular step, just omit the corresponding directive.
- The Service and NameTrans directives work like the Exec directive and are dependent on its occurrence and placement relative to other mapping directives within the configuration file. This means that the server processes the Service, NameTrans, Map, Pass, Exec, Redirect, and Fail directives in their sequential order within the configuration file. When it successfully maps a URL to a file, it does not read or process any other of these directives.
- You can also have more than one configuration directive for a step. For example, you could include two NameTrans directives, each pointing to a different application function. When the server performs the name translation step, it will process your name translation functions in the order in which they appear in the configuration file.
- Multiple IP configuration (using a trailing IP address or template) is supported only for the Service and NameTrans directives.

- If the server fails to load a specific application function or you have a ServerInit directive that does not return an OK return code, no other application functions for that compiled Server API program will be called. Any processing specific to that program which was done up to this point is ignored. Other Server API programs that you include in these directives, and their application functions, are not affected.

Server API directives and syntax:

Table 13. Directives, variables, and access path

Directive	Variable	Access path
ServerInit		/QSYS.LIB/MYLIB/MYGWAPI.SRVPGM:function_name init_string
PreExit		/QSYS.LIB/MYLIB/MYGWAPI.SRVPGM:function_name
Authentication	type	/QSYS.LIB/MYLIB/MYGWAPI.SRVPGM:function_name
NameTrans	/URL	/QSYS.LIB/MYLIB/MYGWAPI.SRVPGM:function_name IP_address_template
Authorization	/URL	/QSYS.LIB/MYLIB/MYGWAPI.SRVPGM:function_name
ObjectType	/URL	/QSYS.LIB/MYLIB/MYGWAPI.SRVPGM:function_name
Service	/URL	/QSYS.LIB/MYLIB/MYGWAPI.SRVPGM:function_name* IP_address_template
Data Filter		/QSYS.LIB/MYLIB/MYGWAPI.SRVPGM:function_name: function_name:function_name
Log	/URL	/QSYS.LIB/MYLIB/MYGWAPI.SRVPGM:function_name
Error	/URL	/QSYS.LIB/MYLIB/MYGWAPI.SRVPGM:function_name
PostExit	/URL	/QSYS.LIB/MYLIB/MYGWAPI.SRVPGM:function_name
ServerTeam		/QSYS.LIB/MYLIB/MYGWAPI.SRVPGM:function_name

Server API directive variables: The variables in these directives have the following meanings:

type Used only with the Authentication directive. This variable determines if your application function is called. Valid values are:

Basic Application function is called only for basic authentication requests.

***** Application function is called for all requests.

/URL Determines for which URLs your application function is called. URL specifications in these directives are virtual (they do not include the protocol) but a slash (/) precedes them. For example, /www.ics.raleigh.ibm.com is correct, but http://www.ics.raleigh.ibm.com is not. Valid values are:

A specific URL

Application function is called only for that URL.

URL template

Application function is called only for URLs that match the template. You can specify a template as /URL*/,/*, or *.

Note: An URL template is required with the Service directive if you want path translation to occur.

/path/file

The fully qualified file name of your compiled program.

:function_name

The name you gave your application function within your program.

In the DataFilter directive, you must supply the names of the open, write, and close functions.

The Service directive requires an asterisk (*) after the :function_name, if you want to have access to path information.

init_string

Optional on the ServerInit directive, this can contain any arbitrary text you want to pass to your application function. Use HTTPD_extract() to extract the text from the INIT_STRING variable.

IP_address_template

Used only with the Service and NameTrans directives on servers that have more than one IP address. This variable determines if your application function is called only for requests that comes on a specific IP address or on a range of IP addresses.

Directives for HTTP Server

This topic provides information about the supported directives for HTTP Server (powered by Apache).

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

The supported modules can be found in the HTTP Server directive finder.

See “Directives no longer supported on HTTP Server (powered by Apache)” on page 466 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 iSeries interface to set up and manage your HTTP Server.

Directive list for HTTP Server

This topic provides information about the supported directives, in alphabetical order, for HTTP Server (powered by Apache).

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

For an alphabetical list of the modules, see “Directives for HTTP Server.” See the directives no longer supported by HTTP Server (powered by Apache) for directives missing from this list.

Note: These directives are provided for reference information only. Use the IBM Web Administration for iSeries interface to set up and manage your HTTP Server.

“A” - “B” on page 457 - “C” on page 457 - “D” on page 458 - “E” on page 458 - “F” on page 459 - “G” on page 459 - “H” on page 459 - “I” on page 459 - “J” on page 460 - “K” on page 460 - “L” on page 460 - “M” on page 461 - “N” on page 461 - “O” on page 461 - “P” on page 461 - “R” on page 462 - “S” on page 463 - “T” on page 464 - “U” on page 464 - “V” on page 464 - “W” on page 464

A

- “AcceptPathInfo” on page 531
- “AccessFileName” on page 532
- Action
- AddAlt

- AddAltByEncoding
- AddAltByType
- AddCharset
- AddClient
- “AddDefaultCharset” on page 532
- AddDescription
- AddEncoding
- AddHandler
- AddIcon
- AddIconByEncoding
- AddIconByType
- AddInputFilter
- AddLanguage
- AddOutputFilter
- “AddOutputFilterByType” on page 533
- AddType
- Alias
- AliasMatch
- Allow
- AllowCONNECT
- “AllowOverride” on page 534
- AlwaysDirectoryIndex
- “AppServer” on page 620
- AsAuthAuthoritative
- “AuthName” on page 534
- “AuthType” on page 535

B

- BrowserMatch
- BrowserMatchNoCase

C

- CacheDefaultExpire
- CacheDirLength
- CacheDirLevels
- CacheExpiryCheck
- CacheGcClean
- CacheGcDaily
- CacheGcInterval
- CacheGcMemUsage
- CacheGcUnused
- CacheIgnoreCacheControl
- CacheIgnoreNoLastMod
- CacheLastModifiedFactor
- CacheLocalFD
- CacheLocalFile

- CacheLocalFileMmap
- CacheLocalSizeLimit
- CacheMaxExpire
- CacheMaxFileSize
- CacheMinFileSize
- CacheNegotiatedDocs
- CacheRoot
- CacheSize
- CacheTimeMargin
- CGIConvMode
- "CgiInitialUrl" on page 522
- CGIMultiThreaded
- CGIRecyclePersist
- "CookieDomain" on page 712
- "CookieExpires" on page 712
- "CookieName" on page 713
- "CookieStyle" on page 713
- "CookieTracking" on page 713
- CustomLog

D

- Dav
- DavDepthInfinity
- DavLockDB
- DavMinTimeout
- DavQsysLockDB
- DefaultFsCCSID
- DefaultIcon
- DefaultLanguage
- DefaultNetCCSID
- "DefaultType" on page 536
- "DeflateBufferSize" on page 578
- "DeflateCompressionLevel" on page 579
- "DeflateFilterNote" on page 579
- "DeflateMemLevel" on page 580
- "DeflateWindowSize" on page 580
- Deny
- "<Directory>" on page 536
- DirectoryIndex
- "<DirectoryMatch>" on page 538
- "DocumentRoot" on page 538
- DynamicCache

E

- "ErrorDocument" on page 539
- "ErrorLog" on page 541

- Example
- ExpiresActive
- ExpiresByType
- ExpiresDefault

F

- "FileETag" on page 543
- "<Files>" on page 544
- "<FilesMatch>" on page 545
- ForceLanguagePriority
- "ForceType" on page 545
- "FRCACacheLocalFileRunTime" on page 491
- "FRCACacheLocalFileSizeLimit" on page 492
- "FRCACacheLocalFileStartUp" on page 492
- "FRCACacheLocalSizeLimit" on page 493
- "FRCACookieAware" on page 493
- "FRCAEnableFileCache" on page 493
- "FRCAEnableProxy" on page 494
- "FRCAEndofURLMarker" on page 494
- "FRCAMaxCommBufferSize" on page 495
- "FRCAMaxCommTime" on page 495
- "FRCAProxyCacheEntitySizeLimit" on page 495
- "FRCAProxyCacheExpiryLimit" on page 496
- "FRCAProxyCacheRefreshInterval" on page 496
- "FRCAProxyCacheSizeLimit" on page 497
- "FRCAProxyPass" on page 497
- "FRCARandomizeResponse" on page 498
- "FRCACustomLog" on page 651

G

- GroupFile

H

- HACGI
- HAModel
- Header
- HeaderName
- "HostNameLookups" on page 546
- "HotBackup" on page 547

I

- "IdentityCheck" on page 547
- "<IfDefine>" on page 548
- "<IfModule>" on page 548
- ImapBase
- ImapDefault

- Imaplist
- “Include” on page 549
- IndexIgnore
- IndexOptions
- IndexOrderDefault

J

- JkAsfTomcat
- JkLogFile
- JkLogLevel
- JkMount
- JkMountCopy
- JkWorkersFile

K

- “KeepAlive” on page 549
- “KeepAliveTimeout” on page 550

L

- LanguagePriority
- LDAPConfigFile
- LDAPRequire
- LDAPInclude
- ldap.AppId
- ldap.application.authType
- ldap.application.DN
- ldap.application.password.stashFile
- ldap.cache.timeout
- ldap.group.memberAttributes
- ldap.group.name.filter
- ldap.group.url
- ldap.idleConnection.timeout
- ldap.NTDomain
- ldap.ObjectClass
- ldap.realm
- ldap.search.timeout
- ldap.transport
- ldap.url
- ldap.user.authType
- ldap.user.name.fieldSep
- ldap.user.name.filter
- ldap.version
- ldap.waitForRetryConnection.interval
- “<Limit>” on page 550
- “<LimitExcept>” on page 551
- “LimitRequestBody” on page 551

- “LimitRequestFields” on page 552
- “LimitRequestFieldsize” on page 552
- “LimitRequestLine” on page 552
- “LimitXMLRequestBody” on page 553
- “Listen” on page 553
- “ListenBacklog” on page 554
- LiveLocalCache
- LmIntervalTime
- LmMaxReactivation
- LmResponseTime
- LmUrlCheck
- “LmUrlCheckBackup” on page 603
- LoadModule
- “<Location>” on page 554
- “<LocationMatch>” on page 555
- “LogCycle” on page 556
- LogFormat
- “LogLength” on page 557
- “LogLevel” on page 558
- “LogMaint” on page 558
- “LogTime” on page 560

M

- MapMatch
- MaxCGIJobs
- “MaxKeepAliveRequests” on page 560
- MaxPersistentCGI
- MaxPersistentCGITimeout
- MaxThreadedCGIJobs
- MetaDir
- MetaFiles
- MetaSuffix
- ModMimeUsePathInfo
- MultiviewsMatch

N

- “NameVirtualHost” on page 561
- NoProxy

O

- “Options” on page 561
- Order

P

- PassEnv
- PasswdFile

- PersistentCGITimeout
- “ProfileToken” on page 563
- <Proxy>
- ProxyBlock
- ProxyCacheOnly
- ProxyDomain
- ProxyErrorOverride
- <ProxyMatch>
- ProxyMaxForwards
- ProxyNoCache
- ProxyNoConnect
- ProxyPass
- ProxyPassReverse
- ProxyPreserveHost
- ProxyReceiveBufferSize
- ProxyRemote
- ProxyRequests
- ProxyReverse
- ProxyTimeout
- ProxyVia

R

- ReadmeName
- Redirect
- RedirectMatch
- RedirectPermanent
- RedirectTemp
- RequestHeader
- RemoveCharset
- RemoveClient
- RemoveEncoding
- RemoveHandler
- RemoveInputFilter
- RemoveLanguage
- RemoveOutputFilter
- RemoveType
- Require
- RewriteBase
- RewriteCond
- RewriteEngine
- RewriteLog
- RewriteLogLevel
- RewriteMap
- RewriteOptions
- RewriteRule

- “RuleCaseSense” on page 563

S

- “Satisfy” on page 564
- Script
- ScriptAlias
- ScriptAliasMatch
- ScriptLog
- ScriptLogBuffer
- ScriptLogLength
- “SendBufferSize” on page 564
- “ServerAdmin” on page 565
- “ServerAlias” on page 565
- “ServerName” on page 566
- “ServerPath” on page 567
- “ServerRoot” on page 567
- “ServerSignature” on page 568
- “ServerTokens” on page 568
- “ServerUserID” on page 569
- SetEnv
- SetEnvIf
- SetEnvIfNoCase
- “SetHandler” on page 570
- “SetInputFilter” on page 570
- “SetOutputFilter” on page 571
- SSIEndTag
- SSIErrorMsg
- SSIStartTag
- SSITimeFormat
- SSIUndefinedEcho
- SSLAppName
- SSLAuthType
- SSLCacheDisable
- SSLCacheEnable
- SSLCipherBan
- SSLCipherRequire
- SSLCipherSpec
- SSLClientAuth
- SSLClientAuthGroup
- SSLClientAuthRequire
- SSLClientCertDisable
- SSLClientCertEnable
- SSLDenySSL
- SSLDisable
- SSLEnable

- SSLRequireSSL
- SSLUpgrade
- SSLVersion
- SSLV2Timeout
- SSLV3Timeout
- StartCGI
- StartThreadedCGI
- SuffixCaseSense

T

- “ThreadsPerChild” on page 571
- “TimeOut” on page 571
- TransferLog
- TypesConfig

U

- UnsetEnv
- “UseCanonicalName” on page 572
- UseJCD
- UserDir
- UserID
- “UseShutdown” on page 573

V

- VirtualDocumentRoot
- VirtualDocumentRootIP
- “<VirtualHost>” on page 573
- VirtualScriptAlias
- VirtualScriptAliasIP

W

- “WASInstance” on page 620

Directive term definitions for HTTP Server (powered by Apache)

This topic provides information about the directive terms used for HTTP Server (powered by Apache).

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 (powered by Apache)" on page 7" 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 (powered by Apache)" on page 7" 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 (powered by Apache)" on page 7" 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 indicates if the directive is new for **iSeries** HTTP Server (powered by Apache), **modified** for iSeries HTTP Server (powered by Apache), or an unmodified **Apache** directive. Possible values for this attribute include:

iSeries

A new directive created for the iSeries HTTP Server (powered by Apache).

Modified

An Apache server directive modified to support the iSeries HTTP Server (powered by Apache).

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 (powered by Apache)

This topic provides information about what directives are no longer supported by HTTP Server (powered by Apache).

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

The following directives are no longer supported on HTTP Server (powered by Apache).

Directives

- "AddModule"
- "ClearModuleList" on page 467
- "IconPath" on page 467
- "Port" on page 468

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 466 directive.

IconPath:

Module: mod_auto_index

Syntax: IconPath

Default: IconPath /icons

Context: server config, virtual host, directory, .htaccess

Override: none

Origin: iSeries

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/HTTP/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 iSeries specific directive for Apache; therefore, precautions must be taken if the Apache configuration file is modified manually. On the iSeries, you would use:

```
Alias /icons /QIBM/ProdData/HTTP/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 553 directive is used as an alternative to Port.

Module mod_access

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Summary

The module mod_access provides access control based on a client's hostname or IP address.

Directives

- "Allow"
- "Deny" on page 470
- "Order" on page 470
- "Require" on page 471

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

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* 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,” 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 469 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.

Require:

Module: mod_access
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.

Module mod_actions

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Summary

The module mod_actions provides for executing CGI scripts based on media type or request method.

Directives

- “Action”
- “Script” on page 473

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 (powered by Apache)” on page 755 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

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Summary

The module mod_alias provides mapping for different parts of the host filesystem in the document tree and also for URL redirection.

Directives

- “Alias” on page 474
- “AliasMatch” on page 474
- “MapMatch” on page 475
- “Redirect” on page 475
- “RedirectMatch” on page 476
- “RedirectPermanent” on page 477
- “RedirectTemp” on page 478
- “ScriptAlias” on page 478

- “ScriptAliasMatch” on page 479

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 538. 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/iSeries/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 536 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 554 containers are run through once before aliases are performed, so they will apply.

See “ScriptAlias” on page 478 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,” 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 iSeries. 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 752 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 iSeries. 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 752 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 536 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 475, 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 a & 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 752 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/iSeries/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 752 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/iSeries/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 752 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 474 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 536 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 iSeries.

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 478, 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 iSeries. 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/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 mod_ap_charset

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Summary

The module mod_ap_charset provides support for performing ASCII to EBCDIC and EBCDIC to ASCII codepage conversions.

Directives

- “DefaultFsCCSID”
- “DefaultNetCCSID” on page 481
- “UseJCD” on page 481

DefaultFsCCSID:

Module: core

Syntax: DefaultFsCCSID *server-character-set-identification-number*

Default: dependent on server settings

Context: server config

Override: none

Origin: iSeries

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 *csid* 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 iSeries.
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
```

DefaultNetCCSID:

Module: mod_ap_charset

Syntax: DefaultNetCCSID *client-character-set-identification-number*

Default: DefaultNetCCSID 819

Context: server config

Override: none

Origin: iSeries

Example: DefaultNetCCSID 819

The DefaultNetCCSID directive specifies the client character set environment and defines the ASCII 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).

Parameter: *server-character-set-identification-number*

- The *server-character-set-identification-number* is an integer value.

A configuration file can contain more than one DefaultNetCCSID directive, but the last directive in the configuration file determines the CCSID.

UseJCD:

Module: mod_ap_charset

Syntax: UseJCD *On* | *Off*

Default: UseJCD Off

Context: server config

Override: none

Origin: iSeries

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

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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.

To enable ARM on IBM HTTP Server, 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 is installed and configured.
3. Ensure you have the latest required PTFs installed for EWLM to monitor the HTTP Server for iSeries application.
4. Add the following directives to the configuration file:

```
LoadModule arm4_module /QSYS.LIB/QHTTSPVR.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 iSeries interface.
- b. Click the **Manage** tab.
- c. Click the **HTTP Servers** subtab.
- d. Select your HTTP Server (powered by Apache) from the **Server** list.
- e. Expand **Tools**.
- f. 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.

- g. Click **OK** when you finish editing the configuration file.
- h. Stop and restart the HTTP Server.

For more information on eWLM, see Enterprise Workload Manager in the eServer Information Center.

Directives

- “`ArmApplicationName`”
- “`ArmInstrumentHandler`” on page 483
- “`ArmLoadLibrary`” on page 483
- “`ArmTransactionName`” on page 483

ArmApplicationName:

Module: `mod_arm4_ap20`

Syntax: `ArmApplicationName application_name`

Default: `ArmTransactionName "IBM Webserving Plugin"`

Context: `server`

Override: `none`

Origin: `iSeries`

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/QHTTSPVR.LIB/QZSRARM.SRVPGM`

Example: `ArmApplicationName "IBM Webserving Plugin"`

This directive specifies the application name registered with the ARM (Application Response Measurement) Agent.

ArmInstrumentHandler:

Module: mod_arm4_ap20

Syntax: ArmInstrumentHandler *on|off*

Default: ArmInstrumentHandler off

Context: server

Override: none

Origin: iSeries

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|unlock_transaction is called across content handlers.

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: iSeries

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 (powered by Apache). 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: iSeries

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

This directive specifies the transaction name registered with the ARM (Application Response Measurement) agent.

Module mod_as_auth

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Summary

The module `mod_as_auth` provides user authentication using iSeries system profiles, Internet users (through validation lists), or LDAP users.

Directives

- “AsAuthAuthoritative”
- “GroupFile”
- “PasswdFile” on page 485
- “UserID” on page 485

AsAuthAuthoritative:

Module: `mod_as_auth`

Syntax: `AsAuthAuthoritative On | Off`

Default: `AsAuthAuthoritative On`

Context: `directory`

Override: `none`

Origin: `iSeries`

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 the iSeries (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: `iSeries`

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 of the iSeries.

Note: The `GroupFile` directive is case-sensitive. If the filename is incorrectly cased, the `GroupFile` directive will not work properly. Since iSeries 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 535, and Require.

PasswdFile:

Module: mod_as_auth

Syntax: PasswdFile *passfile* [*passfile* *passfile* ...]

Default: none

Context: directory

Override: none

Origin: iSeries

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 iSeries 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 (powered by Apache) 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: iSeries

Example: UserID WEBUSER

Example: UserID %%SERVER%%

Example: UserID %%CLIENT%%

The UserID directive specifies the iSeries system profile to the server. For a protected resource (one for which Protection directives are defined), the UserID directive specifies which iSeries 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 iSeries 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 (powered by Apache) 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 iSeries system profile. If there is no iSeries 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 iSeries profiles, client certificates, and Kerberos.

The profile that issued the STRTCPSVR command to start HTTP Server (powered by Apache) 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 (powered by Apache) 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 iSeries authority checking for the requested resource is based on that profile.

When HTTP Server (powered by Apache) 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.

Module `mod_as_cache` for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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”
- “CacheLocalFile” on page 488
- “CacheLocalFileMmap” on page 488
- “CacheLocalFilePublic” on page 489
- “CacheLocalFileSizeLimit” on page 489
- “CacheLocalSizeLimit” on page 490
- “DynamicCache” on page 490
- “FRCACacheLocalFileRunTime” on page 491
- “FRCACacheLocalFileSizeLimit” on page 492
- “FRCACacheLocalFileStartUp” on page 492
- “FRCACacheLocalSizeLimit” on page 493
- “FRCACookieAware” on page 493
- “FRCAEnableFileCache” on page 493
- “FRCAEnableProxy” on page 494
- “FRCAEndofURLMarker” on page 494
- “FRCAMaxCommBufferSize” on page 495
- “FRCAMaxCommTime” on page 495
- “FRCAProxyCacheEntitySizeLimit” on page 495
- “FRCAProxyCacheExpiryLimit” on page 496
- “FRCAProxyCacheRefreshInterval” on page 496
- “FRCAProxyCacheSizeLimit” on page 497
- “FRCAProxyPass” on page 497
- “FRCARandomizeResponse” on page 498
- “LiveLocalCache” on page 498

CacheLocalFD:

Module: `mod_as_cache`

Syntax: `CacheLocalFD filename`

Default: none

Context: server config

Override: none

Origin: iSeries

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: iSeries

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: iSeries

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: iSeries

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: iSeries
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: iSeries
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 three gigabytes); 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: iSeries
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: iSeries

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: iSeries

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: iSeries

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: iSeries

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: iSeries

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: iSeries

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: iSeries

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 <*string*>

Default: none

Context: server config

Override: none

Origin: iSeries

Example: FRCAEndofURLMarker ###'

Example: FRCAEndofURLMarker eou

Example: FRCAEndofURLMarker #eou#

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.

Parameter: `<string>`

- The `<string>` parameter value specifies a string of one or more characters including #, \$.

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 (powered by Apache) 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 (powered by Apache). 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 (powered by Apache).

FRCAProxyCacheEntitySizeLimit:

Module: `mod_as_cache`

Syntax: `FRCAProxyCacheEntitySizeLimit size`

Default: `FRCAProxyCacheEntitySizeLimit 92160`

Context: `server config`

Override: `none`

Origin: `iSeries`

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: iSeries

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: iSeries

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: iSeries

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: iSeries

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: iSeries

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.

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: iSeries

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.

Module `mod_asis` for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Summary

This module allows file types to be defined such that HTTP Server sends them without adding HTTP headers. `mod_asis` 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` for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 582 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"
- "AddAltByEncoding" on page 501
- "AddAltByType" on page 501
- "AddDescription" on page 501
- "AddIcon" on page 502
- "AddIconByEncoding" on page 503
- "AddIconByType" on page 503
- "DefaultIcon" on page 503
- "HeaderName" on page 504
- "IndexIgnore" on page 504
- "IndexOptions" on page 505
- "IndexOrderDefault" on page 508
- "ReadmeName" on page 508

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 554 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 554 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 554 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 554 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 554 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 554 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 554 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 554 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 (powered by Apache)” on page 11 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 508.

Note: This directive is not supported in “<Location>” on page 554 containers.

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 554 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.

Parameter: *option*

- The *option* parameter can be one of the following:

Parameter	Description
DescriptionWidth= [n]	<ul style="list-style-type: none">• 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 bytes wide.• DescriptionWidth=* grows the column to the width necessary to accommodate the longest description string. See “AddDescription” on page 501 for information on truncating.
FancyIndexing	This 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. Note: When the display is sorted by “Size”, it is the actual size of the files that’s used, not the displayed value. Regardless of the ShowSmallFileBytes option setting, 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”.

Parameter	Description
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 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 HTTP Server software.
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. 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 HTTP Server will read the document for the value of the TITLE tag. This is CPU and disk intensive.
SelectiveDirAccess	This option causes the server to return directory listings only for directories that contain a wwwbrws file. The contents of the wwwbrws file are not important. The server only checks for its existence. The object is a member name of an iSeries 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. Note: SelectiveDirAccess is an iSeries specific option. This specific option works on a per directory basis. You must specify the ± SelectiveDirAccess on a Directory container.
ShowOwner	This determines whether directory listings should include the owner ID for each file.
SuppressColumnSorting	If specified, 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" on page 501 for information about setting the file description. See also the DescriptionWidth index option to limit the size of the description column.

Parameter	Description
SuppressHTMLPreamble	If the directory actually contains a file specified by the "HeaderName" on page 504 directive, the module usually includes the contents of the file after a standard HTML preamble (<HEAD>, <HEAD>, and so on). 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 suppresses the display of icons on directory listings. The default is that no options are enabled.
SuppressLastModified	This will suppress the display of the last modification date, in fancy indexing listings.
SuppressRules	This 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 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. Some UNIX systems do so, as do OS/2's JFS and Win32's NTFS volumes. OS/2® and Win32 FAT volumes, for example, do not. Once this feature is enabled, the client or proxy can track changes to the list of files when they perform a HEAD request. Note some operating systems correctly track new and removed files, but do not track changes for sizes or dates of the files within the directory. 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: <ul style="list-style-type: none"> • ibm-1.7 • ibm-1.7.2 • ibm-1.7.12 • ibm-1.8.2 • ibm-1.8.2a • ibm-1.12 If the number starts with a zero, then it is considered to be a fraction: <ul style="list-style-type: none"> • ibm-1.001 • ibm-1.002 • ibm-1.030 • ibm-1.04

Notes on IndexOptions directives:

- 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: This directive is not supported in "<Location>" on page 554 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 554 containers. The directive may be inherited in a "<Directory>" on page 536 context, but not in a "<VirtualHost>" on page 573 context.

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 504 directive, which uses the same mechanism as ReadmeName.

Note: This directive is not supported in “<Location>” on page 554 containers.

Module `mod_cern_meta` for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Summary

The module `mod_cern_meta` provides for CERN httpd meta file semantics.

Directives

- “MetaFiles”
- “MetaDir”
- “MetaSuffix” on page 510

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 509 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 for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 511.

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: 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 513
- "CacheExpiryCheck" on page 514

- “CacheIgnoreCacheControl” on page 514
- “CacheIgnoreNoLastMod” on page 515
- “CacheLastModifiedFactor” on page 516
- “CacheMaxExpire” on page 517
- “CacheMaxFileSize” on page 518
- “CacheMinFileSize” on page 518
- “CacheTimeMargin” on page 519

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.

CacheExpiryCheck:

Module: mod_cache

Syntax: CacheExpiryCheck *on* | *off*

Default: CacheExpiryCheck on

Context: server config, virtual host

Override: none

Origin: iSeries

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.

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/QHTTTPSVR.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 placing a maximum data size limit on individual cache entries. If the disk cache function is disabled (see CacheRoot), this setting has no affect.

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 affect.

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.

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 proxy 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 for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.


See IBM Service  for more information.

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 662 directive, or by being in a "ScriptAlias" on page 478 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 538 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 on HTTP Server” on page 766 for a list of environment variables.

REMOTE_HOST

This will only be set if “HostNameLookups” on page 546 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 547 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 iSeries 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 iSeries debugger. You can also use standard error (stderr) for debug information. The information written to STDERR is written to the ScriptLog if one is configured or to the ErrorLog if a ScriptLog is not configured.

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” on page 522
- “CgiInitialUrl” on page 522
- “CGIMultiThreaded” on page 523
- “CGIRecyclePersist” on page 524

- “MaxCGIJobs” on page 524
- “MaxPersistentCGI” on page 524
- “MaxPersistentCGITimeout” on page 525
- “MaxThreadedCGIJobs” on page 525
- “PersistentCGITimeout” on page 526
- “ScriptLog” on page 526
- “ScriptLogBuffer” on page 527
- “ScriptLogLength” on page 527
- “StartCGI” on page 527
- “StartThreadedCGI” on page 528
- “ThreadedCgiInitialUrl” on page 529

CGIConvMode:

Module: mod_cgi

Syntax: CGIConvMode *mode*

Default: CGIConvMode EBCDIC

Context: server config, virtual host, directory, .htaccess, Not in Limit

Override: FileInfo

Origin: iSeries

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:

Mode	Description
EBCDIC	<p>The server converts everything into the EBCDIC CCSID of the job. In addition, the server converts escaped octets from ASCII to EBCDIC.</p> <p>The server assumes the header output and encoded characters “%xx” are in the EBCDIC CCSID of the job or default fsCCSID. The server assumes that the body output is in the default EBCDIC CCSID of the server job unless specified otherwise using the Content-type header.</p>
EBCDIC_JCD	<p>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.</p>
BINARY	<p>The server performs no conversion on QUERY_STRING or STDIN data. Environment variables are encoded in the job’s EBCDIC CCSID.</p> <p>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.</p>

| CgiInitialUrl:

| **Module:** mod_cgi

| **Syntax:** CgiInitialUrl *url userid*

| **Default:** none

| **Context:** server config
| **Override:** none
| **Origin:** iSeries
| **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/pacecgi 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 iSeries userid or * where * means all of the
| userids specified on the StartCgi directive. To check for valid values, follow the rules for
| iSeries user profiles. The userid is optional.

| **CGIMultiThreaded:**

Module: mod_cgi

Syntax: CGIMultiThreaded *on* | *off*

Default: CGIMultiThreaded off

Context: server config, virtual host, directory, .htaccess, Not in Limit

Override: FileInfo

Origin: iSeries

Example: CGIMultiThreaded on

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 iSeries 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. This directive has no impact on CGI programs written in JAVA. JAVA CGI programs are always run in a job that is multiple thread capable. You should only use this directive in your non-Java CGI program that requires multiple threads.

CGIRecyclePersist:

Module: mod_cgi

Syntax: CGIRecyclePersist *on* | *off*

Default: CGIRecyclePersist off

Context: server config, virtual host, directory, .htaccess, Not in Limit

Override: FileInfo

Origin: iSeries

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.

MaxCGIJobs:

Module: mod_cgi

Syntax: MaxCGIJobs *number*

Default: MaxCGIJobs 40 or value of the ThreadsPerChild directive

Context: server config

Override: none

Origin: iSeries

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 iSeries profiles (profiles with no password) or use %%CLIENT%% (each user has their own iSeries 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 of the ThreadsPerChild directive, then the server will use a default of (value of ThreadsPerChild). If ThreadsPerChild directive is not present, then the default of this directive will be set to 40.

MaxPersistentCGI:

Module: mod_cgi

Syntax: MaxPersistentCGI *number*

Default: MaxPersistentCGI 40

Context: server config

Override: none

Origin: iSeries

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: iSeries

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: MaxThreadedCGIJobs 40 or the value of ThreadsPerChild directive

Context: server config

Override: none

Origin: iSeries

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 iSeries profiles (profiles with no password) or use %%CLIENT%% (each user has their own iSeries 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 of the ThreadsPerChild directive, then the server will use a default of (value of ThreadsPerChild). If ThreadsPerChild directive is not present, then the default of this directive will be set to 40.

PersistentCGITimeout:

Module: mod_cgi

Syntax: PersistentCGITimeout *number*

Default: PersistentCGITimeout 300

Context: server config

Override: none

Origin: iSeries

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.

All file systems supported by HTTP Server (original) for logs will be supported. The ScriptLog file will be created with the same CCSID as defaultFSCCSID. 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 defaultFSCCSID.

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: iSeries
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: iSeries
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 iSeries 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 iSeries 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 iSeries user profile), then it should be noted that %%CLIENT%% is not a valid value on this directive. Using iSeries profiles like this should only be done in an intranet or highly secure server because you would not want to give just anyone an iSeries 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: iSeries

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 iSeries 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 iSeries 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 iSeries user profile), then it should be noted that %%CLIENT%% is not a valid value on this directive. Using iSeries profiles like this should only be done in an intranet or highly secure server because you would not want to give just anyone an iSeries 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: iSeries

Example: ThreadedCgiInitialUrl /qsys.lib/cgi.lib/mycgi.pgm QTMHHTTP

Example: ThreadedCgiInitialUrl /QOpenSys/mypacedir/pacecgi

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 a Java CGI program is used, the server ignores it and log a message in the error log.

If there are no StartThreadedCgi directives, an error is posted and the server does not start.

Module core for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Summary

These directives control the core function of HTTP Server (powered by Apache).

Directives

- “AcceptPathInfo” on page 531
- “AcceptThreads” on page 531
- “AccessFileName” on page 532
- “AddDefaultCharset” on page 532
- “AddOutputFilterByType” on page 533
- “AllowEncodedSlashes” on page 533
- “AllowOverride” on page 534
- “AuthName” on page 534
- “AuthType” on page 535
- “DefaultType” on page 536
- “<Directory>” on page 536
- “<DirectoryMatch>” on page 538
- “DocumentRoot” on page 538
- “EnableSendfile” on page 539
- “ErrorDocument” on page 539
- “ErrorLog” on page 541
- “ErrorLogFormatDDS” on page 543
- “FileETag” on page 543

- “<Files>” on page 544
- “<FilesMatch>” on page 545
- “ForceType” on page 545
- “ForensicLog” on page 545
- “HostNameLookups” on page 546
- “HotBackup” on page 547
- “IdentityCheck” on page 547
- “<IfDefine>” on page 548
- “<IfModule>” on page 548
- “Include” on page 549
- “KeepAlive” on page 549
- “KeepAliveTimeout” on page 550
- “<Limit>” on page 550
- “<LimitExcept>” on page 551
- “LimitRequestBody” on page 551
- “LimitRequestFields” on page 552
- “LimitRequestFieldsize” on page 552
- “LimitRequestLine” on page 552
- “LimitXMLRequestBody” on page 553
- “Listen” on page 553
- “ListenBacklog” on page 554
- “<Location>” on page 554
- “<LocationMatch>” on page 555
- “LogCycle” on page 556
- “LogLength” on page 557
- “LogLevel” on page 558
- “LogMaint” on page 558
- “LogMaintHour” on page 560
- “LogTime” on page 560
- “MaxKeepAliveRequests” on page 560
- “NameVirtualHost” on page 561
- “Options” on page 561
- “ProfileToken” on page 563
- “RuleCaseSense” on page 563
- “Satisfy” on page 564
- “SendBufferSize” on page 564
- “ServerAdmin” on page 565
- “ServerAlias” on page 565
- “ServerName” on page 566
- “ServerPath” on page 567
- “ServerRoot” on page 567
- “ServerSignature” on page 568
- “ServerTokens” on page 568
- “ServerUserID” on page 569
- “SetHandler” on page 570

- “SetInputFilter” on page 570
- “SetOutputFilter” on page 571
- “ThreadsPerChild” on page 571
- “TimeOut” on page 571
- “UseCanonicalName” on page 572
- “UseShutdown” on page 573
- “<VirtualHost>” on page 573

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/nowhere.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: mod_core

Syntax: AcceptThreads *number*

Default: AcceptThreads 4

Context: server config

Override: none

Origin: iSeries

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 at startup time. The process never creates additional threads.

Parameter: *number*

- The *number* parameter 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 iSeries.

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 534.

AddDefaultCharset:

Module: core

Syntax: AddDefaultCharset *on* | *off* | *charset*

Default: AddDefaultCharset off

Context: server config, virtual host, directory, .htaccess, Not in Limit

Override: FileInfo

Origin: iSeries

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`

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 571 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.

AllowEncodedSlashes:

Module: mod_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 532) 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 485 and Require.
FileInfo	Allow use of the directives controlling document types such as AddEncoding, AddLanguage, AddType, "DefaultType" on page 536, "ErrorDocument" on page 539 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 469, "Deny" on page 470 and "Order" on page 470.
Options	Allow use of the directives controlling specific directory features such as "Options" on page 561.

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 535 *Basic*, and "Require" on page 471, and directives such as "PasswdFile" on page 485.

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 534 and "Require" on page 471. 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 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 534
- PasswdFile *SYSTEM*

- Require *validuser*, or Require user *kerbuser@DOMAIN.COM as400userid* or Require group *groupname* or GroupFile *pathtogroupfile*

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 *validuser* you must include both the kerberos client principal and the as400 userid to which it maps.
- To specify this directive, you must have your Microsoft Internet Explorer set to enable integrated windows authentication.

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 538 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 532 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 38 or “User profiles and required authorities for HTTP Server” on page 40 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 554 and “<LocationMatch>” on page 555 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 536.

DocumentRoot:

Module: core

Syntax: DocumentRoot *directory-path*

Default: DocumentRoot /QIBM/UserData/HTTP/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 iSeries.

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 can 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.

Apache on the iSeries 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 567. 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 556 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 556 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 556 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 556 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 556 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 38 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 558.

ErrorLogFormatDDS:

Module: core
Syntax: ErrorLogFormatDDS *on* | *off*
Default: ErrorLogFormatDDS *off*
Context: server, virtual host, not in limit
Override: none
Origin: iSeries
Example: ErrorLogFormatDDS *on*

The ErrorLogFormatDDS directive is used to specify whether the server should log information to the ErrorLog in the OS/400 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 iSeries logs 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 536 directive and "<Location>" on page 554 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 38 and "User profiles and required authorities for HTTP Server" on page 40 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 544 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: The core directives ForceType and “SetHandler” on page 570 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.

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: iSeries

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.

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` for HTTP Server (powered by Apache)" on page 648 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 '-DLLDAP'  
# in the instance configuration  
<IfDefine LDAP>  
    LoadModule ldap_module /QSYS.LIB/QHTTPSVR.LIB/MOD_LDAP.SRVPGM  
</IfDefine>
```

<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 560.

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 549, “MaxKeepAliveRequests” on page 560, and “TimeOut” on page 571.

<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 550 section and can be used to control both standard and nonstandard-unrecognized methods. See “<Limit>” on page 550 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
```

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 8190

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 8190 (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*
Default: none
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.

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
```

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.

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 536 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 544 section). However some directives, most notably the “AllowOverride” on page 534 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 536, `<Proxy>`, “`<Files>`” on page 544, and “AllowOverride” on page 534 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 570 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 554 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: iSeries

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: iSeries

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 541 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: iSeries

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 556 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_Tog 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: iSeries

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: iSeries

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 549 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 549, and “KeepAliveTimeout” on page 550.

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.

Option	Description
Follow SymLinks	The server will follow symbolic links in this directory. Note: 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 (powered by Apache)" on page 11 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. Note: 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: iSeries

Example: ProfileToken *on*

The Profile Tokens directive creates a 32-byte value called the Profile Token. 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 Profile Token value can be used on any of the iSeries security APIs that accept a Profile Token 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 iSeries user) to generate a Profile Token. A Profile Token is not generated if this parameter is set to *off*, or if basic authentication was not successful.

The Profile Token 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 Profile Token is not generated.

The Profile Token is accessible in HTTP Server (powered by Apache) 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 (powered by Apache) modules can then retrieve this Profile Token and either use it, or pass it on to another application. The AS_Auth_ProfileTkn header is not created if a Profile Token is not generated.

RuleCaseSense:

Module: core

Syntax: RuleCaseSense *on* | *off*

Default: RuleCaseSense *off*

Context: server config

Override: none

Origin: iSeries

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 "Require" on page 471 and 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 limit is 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* parameter is a number that is greater or equal to 512. The default of this directive is set to 0 (the default iSeries TCP value).

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 568 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 539 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 573 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 573 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 (powered by Apache) 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 568).

Note: TCP/IP must be properly configured to recognize all possible server hostnames. See the configuration chapter in TCP/IP Configuration and Reference  for more information.

See also “UseCanonicalName” on page 572, “NameVirtualHost” on page 561 and “ServerAlias” on page 565.

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/websvr

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: ServerAdmin www-admin@myserver.com

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 566 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 565 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 539 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/2.0"
- If *Minimal* is specified, the server sends: "Server: Apache"
- If *OS* is specified, the server sends: "Server: Apache (iSeries)"
- If *Full* is specified, the server sends: "Server: Apache (iSeries) 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 568).

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: iSeries

Example: ServerUserID webmaster

The ServerUserID directive specifies the user profile that the HTTP Server will run under.

Note: If directive `Userid` is set and authentication is performed in a context, and `Userid` specifies `%%SERVER%%`, then directive `ServerUserID` will be used for that context. `ServerUserID` is inherited through all contexts unlike `Userid`. If authentication is performed and directive `Userid` is set to any value other than `%%SERVER%%`, then `ServerUserID` is overridden by `Userid`. If authentication is not performed at all, then `ServerUserID` is used from the correct context.

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: The profile must be authorized to all the directories, files, and other server resources accessed by the HTTP Server unless the server is configured to swap to another profile for specific requests or directories. In order to start the server you must have authority to the specified profile.

See also "UserID" on page 485.

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 536 or “<Location>” on page 554 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 (powered by Apache)” on page 755 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: ThreadsPerChild 40

Context: server config

Override: none

Origin: Apache

Example: ThreadsPerChild

Use this directive to specify the maximum number of threads per server child process. If you do not specify a value for the directive, it inherits the global HTTP Server setting and uses the maximum listed value.

Parameter: *number*

- The *number* value is an integer value that specifies the maximum number of threads per server child process. The default global HTTP Server value is 40.

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.

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 566.

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 *address[:port]* ...> ... </VirtualHost>
Default: none
Context: server config, Not in Limit
Override: none
Origin: Apache
Example: <VirtualHost 10.22.33.44 > ... </VirtualHost>

The term Virtual Host refers to the practice of maintaining more than one server on a server, as differentiated by their apparent hostname (or server 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 any extra path information.

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.

HTTP Server (powered by Apache) supports two types of virtual hosting, they are IP-based Virtual Host and Name-based Virtual Host (the latter variant of virtual hosts is sometimes called host-based or non-IP virtual hosts). 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 server 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 it is difficult to implement on some machines). The benefits of using 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 the protocol. The latest browser versions (for example, HTTP 1.1) do, but there are still older browsers (for example, HTTP 1.0) in use that do not. This may 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 configurations is the `NameVirtualHost` directive. The directive specifies an IP address and 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`. You can add to one of HTTP Server (powered by Apache) configuration files 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; all that is needed is to 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>` tags. The "main server" never serves from the specified IP address. If you start to use virtual hosts, you should not use the "main server" as an independent server, but rather 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, you may want a server to be accessible by more than one name. For example, you may want a server to be accessible as `domain.tld`, or `www2.domain.tld` (assuming the IP addresses pointed to the same server). In fact, you may want all addresses at `domain.tld` assigned to 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.foobar"` then you will need to add `ServerAlias www www.foobar`. 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 pointing different hostnames to the same virtual host.

`<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 virtual host IP address
- A fully qualified domain name for the virtual host IP address. More than one IP address may be specified. This is used when a machine responds to the same name on two different interfaces.

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.

See "Fundamental directive, context, and server area concepts on HTTP Server (powered by Apache)" on page 7, or refer to module `mod_vhost_alias` for further details of how different sections are combined when a request is received.

Module `mod_dav` for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Summary

This module provides class 1 and class 2 WebDAV (Web-based Distributed Authoring and Versioning) functionality for HTTP Server (powered by Apache). 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, 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 576
- "DavLockDB" on page 577
- "DavMinTimeout" on page 577
- "DavQsysLockDB" on page 578

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 filesystems 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 filesystems); 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/QHTTPSVR.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/QHTTPSVR.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 for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 uncompressed output. Compression and decompression is implemented by the DEFLATE filter, located in module mod_deflate. See Apache HTTP Server Version

2.0 Documentation  for additional information and examples on configuring the Apache server to use compression.

Directives

- “DeflateBufferSize”
- “DeflateCompressionLevel” on page 579
- “DeflateFilterNote” on page 579
- “DeflateMemLevel” on page 580
- “DeflateWindowSize” on page 580

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/QHTTTPSVR.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/QHTTTPSVR.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` for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 582

AlwaysDirectoryIndex:

Module: `mod_dir`

Syntax: `AlwaysDirectoryIndex on | off`

Default: `Always DirectoryIndex on`

Context: server config, virtual host, directory, `.htaccess`

Override: Indexes

Origin: iSeries

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 iSeries 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 561 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.

Module mod_disk_cache for HTTP Server (powered by Apache)

This topic describes the module `mod_disk_cache` for HTTP Server (powered by Apache).

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 584
- “`CacheGcClean`” on page 585
- “`CacheGcDaily`” on page 587
- “`CacheGcInterval`” on page 588
- “`CacheGcMemUsage`” on page 589
- “`CacheGcUnused`” on page 590
- “`CacheRoot`” on page 591
- “`CacheSize`” on page 593

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 (powered by Apache) 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 /QOpenSys/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 /QOpenSys/QIBM/UserData/HTTPD/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 /QOpenSys/QIBM/UserData/HTTPD/CacheRoot/MyCache/x3/_9/ directory. For example two (above), the two files will be named vBA.data and vBA.header and will reside within the /QOpenSys/QIBM/UserData/HTTPD/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 (powered by Apache) does not support inheritance for the CacheDirLevels directive.

CacheGcClean:

Module: mod_disk_cache

Syntax: CacheGcClean *hash-key-criteria period*

Default: CacheGcClean *43200 (minutes, or 30 days)

Context: server config, virtual host

Override: none

Origin: iSeries

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 /* 21600

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 minutes. Multiple CacheGcClean directives are allowed. If disk cache maintenance is disabled, this setting has no effect 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 minutes) 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 minutes 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 minutes is not affected. Unmatched data is not affected. See “Two Phase Disk Cache Maintenance” on page 582 for details concerning the disk cache maintenance process.

Example 1: URL match expressions

```
CacheRoot serverCache
CacheGcClean *://ibm.com/* 43200
CacheGcClean ftp://server?.ibm.com/* 20160
```

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 43200 minutes (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 20160 minutes (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 7200
```

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 7200 minutes (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 (powered by Apache) 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: iSeries

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` directives 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 affect 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 582 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 (powered by Apache) 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 582 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 (powered by Apache) 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: `iSeries`

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 (powered by Apache) 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: iSeries

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 587 or “CacheGcInterval” on page 588), this setting has no affect 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 585 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 582 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 (powered by Apache) 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 587 or "CacheGcInterval" on page 588 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 path names based on hash keys (or URLs) provided for data stored in cache (see CacheDirLength and CacheDirLevels). Data is stored in the local file system using these path names, relative to the specified directory root. Since the algorithm generates case sensitive path names, CacheRoot must specify a directory within the QOpenSys (case sensitive) file system. For this reason the following limits are placed on the directory root:

Directory root limits:

- If the directory parameter specifies an absolute path it must start with /QOpenSys/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: /QOpenSys/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 "/QOpenSys/QIBM/UserData/HTTP/CacheRoot/ABC/def".

Example 1: Absolute Path

```
CacheRoot /QOpenSys/QIBM/UserData/HTTP/CacheRoot/proxyCache
ProxyRequests on
```

Example 2: Relative Path

```
CacheRoot proxyCache
ProxyRequests on
```

Example 3: Relative Path (with disk cache function unavailable for proxy data)

```
CacheRoot cache
ProxyRequests on
```

Example 4: Bad Path

```
CacheRoot /MyServerCache
```

For example one, CacheRoot enables the disk cache function (CacheRoot /QOpenSys/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 /QOpenSys/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 /QOpenSys/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 /QOpenSys/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 (powered by Apache) does not support inheritance for the CacheRoot directive.

CacheSize:

Module: mod_disk_cache

Syntax: CacheSize *size*

Default: CacheSize 50000000

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 587 or “CacheGcInterval” on page 588 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 (powered by Apache) does not support inheritance for the CacheSize directive.

Module `mod_env` for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Summary

This module allows the HTTP Server CGI and SSI environment to inherit environment variables.

Directives

- “PassEnv”
- “SetEnv”
- “UnsetEnv” on page 595

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 on HTTP Server” on page 766 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 on HTTP Server” on page 766 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 on HTTP Server” on page 766 for more information.

Parameter: *variable*

- The *variable* parameter is any valid environment variable.

Module mod_example for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 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.

Directives

- “ExpiresActive”
- “ExpiresByType”
- “ExpiresDefault” on page 598

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 is 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: *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.

You can also specify the expiration time calculation using the alternate interval syntax. For example:

```
#enable expirations
ExpiresActive on
#expire GIF images after a month in the clients cache
ExpiresByType image/fig 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.

Module mod_ha for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Summary

The module mod_ha contains directives that define support for the highly available HTTP Server function.

Directives

- "HACGI"
- "HAModel"
- "LmExitProgram" on page 600
- "LmIntervalTime" on page 601
- "LmMaxReactivation" on page 601
- "LmResponseTime" on page 602
- "LmUrlCheck" on page 602
- "LmUrlCheckBackup" on page 603

HACGI:

Module: mod_ha

Syntax: HACGI *on* | *off*

Default: HACGI off

Context: server config, virtual host, directory, .htaccess

Override: none

Origin: iSeries

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: 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: iSeries

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: 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 Web server cluster on HTTP Server” on page 17 for more information regarding highly available Web server models.

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
```

Note: When a server is configured as highly available (HAModel directive is specified), “HotBackup” on page 547 behaves as if it is set to ‘off’ and can not be overwritten.

LmExitProgram:

Module: mod_ha

Syntax: LmExitProgram *libraryname programname*

Default: none

Context: server config

Override: none

Origin: iSeries

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

The LmExitProgram directive configures a QSYS program 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 function 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 iSeries 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 also follow the iSeries rules for program names in a library.

Note: Only QSYS library/program can be used.

LmIntervalTime:

Module: mod_ha

Syntax: LmIntervalTime *interval*

Default: LmIntervalTime 15

Context: server config

Override: none

Origin: iSeries

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: 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/QHTTSPVR.LIB/QZSRCORE.SRVPGM
HAModel 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: iSeries

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: 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: iSeries

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
HAMode1 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: iSeries

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 599 and “Highly available Web server cluster on HTTP Server” on page 17 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: iSeries

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 http://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` for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Summary

The headers module allows for the customization of HTTP response headers. The module allows headers to be merged, replaced or removed.

Directives

- “Header”
- “RequestHeader” on page 606

Header:

Module: `mod_headers`

Syntax: Header [condition] *action header value* [env=[!]environment-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 first 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 third 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. It did not exist prior to this release on iSeries.

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.

Action	Description
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 append 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` for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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: iSeries

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` for HTTP Server (powered by Apache)

This module contains directives that allow HTTP Server to access an Lightweight Directory Access Protocol (LDAP) directory and to query the directory in a database fashion to obtain authentication information.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

These directives provide the server with information regarding the LDAP Servers in which HTTP Server configuration (see `mod_ibm_ldap`) 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"`
- `"ldap.application.authType"` on page 610
- `"ldap.application.DN"` on page 610
- `"ldap.application.password.stashFile"` on page 611
- `"ldap.cache.timeout"` on page 611
- `"ldap.group.memberAttributes"` on page 612
- `"ldap.group.name.filter"` on page 612
- `"ldap.group.url"` on page 612
- `"ldap.idleConnection.timeout"` on page 613
- `"ldap.NTDomain"` on page 614
- `"ldap.ObjectClass"` on page 614
- `"ldap.realm"` on page 615
- `"ldap.search.timeout"` on page 615
- `"ldap.transport"` on page 615
- `"ldap.url"` on page 616
- `"ldap.user.authType"` on page 616
- `"ldap.user.name.fieldSep"` on page 617
- `"ldap.user.name.filter"` on page 617
- `"ldap.version"` on page 618
- `"ldap.waitToRetryConnection.interval"` on page 618
- `"LDAPConfigFile"` on page 618
- `"LDAPRequire"` on page 619

`ldap.AppId`:

Module: `mod_ibm_ldap`

Syntax: `ldap.AppId application_ID`

Default: none

Context: `directory`, `.htaccess`

Override: AuthCfg

Origin: iSeries

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: iSeries

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: iSeries

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: `iSeries`

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: `iSeries`

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: iSeries

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: *attributes*

- The *attributes* parameter is the group attribute names used to extract users from an LDAP group entry.

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: iSeries

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: iSeries

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: iSeries

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: iSeries

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 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: iSeries

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: iSeries

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: iSeries

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: `iSeries`

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: `iSeries`

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: iSeries

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: iSeries

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)(|(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: iSeries

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.waitToRetryConnection.interval:

Module: mod_ibm_ldap

Syntax: ldap.waitToRetryConnection.interval *seconds*

Default: ldap.waitToRetryConnection.interval 30

Context: directory, .htaccess

Override: AuthCfg

Origin: iSeries

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.waitToRetryConnection.interval 60

If an LDAP server is down, HTTP Server may have degraded performance because it will be continually trying to connect. The ldap.waitToRetryConnection.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: iSeries

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.prop

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: iSeries

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.

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 for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Summary

The module mod_ibm_si specifies behavior between associated HTTP Server (powered by Apache) servers and WebSphere Application Servers.

Directives

- "AppServer" on page 620
- "WASInstance" on page 620

AppServer:

Module: mod_ibm_si

Syntax: AppServer *server_name* [*ALL start | nostart] [end | noend]

Default: none

Context: server config

Override: none

Origin: iSeries

Usage Considerations: A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule isi_module /QSYS.LIB/QHTTPSVR.LIB/QZISI.SRVPGM

Example: AppServer ITD start end

Example: AppServer ADMIN start noend

Example: AppServer NODO nostart noend

The AppServer directive is only effective if directive WASInstance declares the WAS instance name for the application server. The AppServer directive specifies the load module to start the application server when the HTTP Server (powered by Apache) is started. The AppServer directive also ends the application server when the same HTTP Server (powered by Apache) is stopped. More than one directive is allowed but with different HTTP Server (powered by Apache) names.

Note: For duplicate names, the last one is specified and used by the server configuration.

Parameter One: *server_name*

- The *server_name* parameter value specifies the HTTP Server (powered by Apache) to be associated with the application server.

Parameter Two: *ALL

- The *ALL parameter specifies that this directive applies to all application servers associated with the instance named on the WASInstance directive. For example, AppServer *ALL start end

Parameter Three: start | nostart |

- The *start* parameter value specifies the application server to start when the associated HTTP Server (powered by Apache) starts.
- The *nostart* parameter value specifies the application server to not start when the associated HTTP Server (powered by Apache) starts.

Parameter Four: end | noend

- The *end* parameter value specifies the application server to end when the associated HTTP Server (powered by Apache) ends.
- The *noend* parameter value specifies the application server to not end when the associated HTTP Server (powered by Apache) ends.

WASInstance:

Module: mod_ibm_si

Syntax: WASInstance *name ProductID ProductID_option*

Default: none

Context: server config

Override: none

Origin: iSeries

Usage Considerations: A LoadModule is required in the configuration file prior to using the directive. The statement should be as follows: LoadModule isi_module /QSYS.LIB/QHTTPSVR.LIB/QZISI.SRVPGM

Example: WASInstance SYSINST 5722XXX 2

Example: WASInstance ITD 5722IWE 2

Example: WASInstance DEFAULT 5733WS5 2

The WASInstance directive specifies the application server instance and the associated product and product option. The directive typically has an accompanying AppServer directive.

Parameter One: *name*

- The *name* parameter value is the name of the application server. Mixed case is allowed.

Parameter Two: *ProductID*

- The *ProductID* parameter value specifies the product. Valid values include:
 - 5722IWE
 - 5722WS5
 - 5722E51
 - 5722XXX (intended for system instance application server, SYSINST)

Parameter Three: *ProductID_option*

- The *ProductID_options* parameter value specifies any valid product options. The option number is a number from 0 to 99 where 0 is *BASE. Valid values include:
 - 2 for *ProductID* parameter value 5722IWE (WAS Express, V5R3)
 - 2 for *ProductID* parameter value 5733WS5 (WAS Base, V5R3)
 - 2 for *ProductID* parameter value 5722E51
 - 5 for *ProductID* parameter value 5733WS5 (WAS ND; future release)

Module `mod_ibm_ssl` for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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" on page 623
- "SSLAuthType" on page 623
- "SSLCacheDisable" on page 624
- "SSLCacheEnable" on page 624
- "SSLCipherBan" on page 625
- "SSLCipherRequire" on page 626
- "SSLCipherSpec" on page 627
- "SSLClientAuth" on page 628
- "SSLClientAuthGroup" on page 628
- "SSLClientAuthRequire" on page 630
- "SSLClientCertDisable" on page 631
- "SSLClientCertEnable" on page 631
- "SSLDenySSL" on page 632
- "SSLDisable" on page 632
- "SSLEnable" on page 632
- "SSLEngine" on page 633
- "SSLProxyAppName" on page 633
- "SSLProxyEngine" on page 634
- "SSLProxyVerify" on page 634
- "SSLProxyVersion" on page 635
- "SSLRequireSSL" on page 636
- "SSLUpgrade" on page 636
- "SSLVersion" on page 637
- "SSLV2Timeout" on page 637
- "SSLV3Timeout" on page 638

SSLAppName:

Module: mod_ibm_ssl

Syntax: SSLAppName *server_application_name*

Default: none

Context: server config, virtual host

Override: none

Origin: iSeries

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 iSeries 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: 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: iSeries

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 iSeries 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 (powered by Apache) 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 iSeries 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: iSeries

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: iSeries

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. An 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: iSeries

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 14. 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

Table 14. Version 2 ciphers (continued)

Long name	Short name
SSL_EXPORT_WITH_RC2_CBC_40_MD5	24

Table 15. 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: iSeries

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 16. 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 17. 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

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: iSeries

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 18. 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 19. 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

Table 19. Version 3 and TLS ciphers (continued)

Long name	Short name
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

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: iSeries

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: iSeries

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 20. 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: iSeries

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 21. 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: iSeries

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: iSeries

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: iSeries

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: iSeries

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: iSeries

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: iSeries

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.

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: iSeries

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: iSeries

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: iSeries

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 22. 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: iSeries

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: iSeries

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 for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 640 for more information on `Imagemaps`.

Directives

- `ImapBase`
- `ImapDefault`
- `Imaplist`

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 `Imaplist` 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 URL. 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 Imaplist 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 URL. Relative URLs can contain '..' syntax and will be resolved relative to the base value . The base value itself will not resolved according to the current value. However, the statement base mailto: will work properly.

Imaplist:

Module: mod_imap

Syntax: Imaplist *none* | *formatted* | *semiformatted* | *unformatted*

Default: Imaplist formatted

Context: server config, virtual host, directory, .htaccess

Override: Indexes

Origin: Apache

Example: Imaplist formatted

The Imaplist directive determines the action taken if an imagemap file is called without valid coordinates.

Parameter: *none* | *formatted* | *semiformatted* | *unformatted*

- The *none* parameter means no list is generated and the default action is performed
- The *formatted* parameter generates a formatted list is the simplest list. 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 list has a consistent, plain look close to that of a directory listing.
- The *semiformatted* parameter generates a semiformatted list, 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 list is the same as a formatted list.
- The *unformatted* parameter generates an unformatted list, 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 lists, 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 "list text" [x,y ...]

directive value x,y ... "list 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 any of the following:

- URL - The URL can be relative or absolute URL. 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 Imaplist is set to none.
- list - 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 for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 561 directive for more information.

Note: The iSeries system does not support XBitHack to enable server-side includes.

Directives

- “SSIEndTag”
- “SSSErrorMsg”
- “SSIStartTag”
- “SSITimeFormat” on page 644
- “SSIUndefinedEcho” on page 644

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.

SSSErrorMsg:

Module: mod_include

Syntax: SSSErrorMsg *string*

Default: SSSErrorMsg "[an error occurred while processing this directive]"

Context: server config, virtual host, directory, .htaccess

Override: none

Origin: Apache

Example: SSSErrorMsg "This is the default error message"

This SSSErrorMsg 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:
SSSErrorMsg "This is the default error message"

SSIStartTag:

Module: mod_include

Syntax: SSIStartTag *string*

Default: SSIStartTag "<!--#"

Context: server config, virtual host

Override: none

Origin: Apache

Example: SSIStartTag "<!--#"

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 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.

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 782 for a list of supported server-side include directives.

Note: HTTP Server (powered by Apache) 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_jk for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Summary

The modules `mod_jk` provides five directives used to communicate with the ASF Tomcat Servlet Engine. These directives go into the HTTP Server (powered by Apache) server configuration file (`httpd.conf`). Any `mod_jk` directive usage must be preceded by loading the Service Program that supports `mod_jk`. The following `LoadModule` must be in the configuration file prior to the use of any `mod_jk` directives.

```
LoadModule mod_jk /QSYS.LIB/QHTTPSVR.LIB/QZTCJK.SRVPGM
```

Directives

- “`JkAsfTomcat`”
- “`JkLogFile`”
- “`JkLogLevel`” on page 646
- “`JkMount`” on page 646
- “`JkMountCopy`” on page 647
- “`JkWorkersFile`” on page 647

JkAsfTomcat:

Module: `mod_jk`

Syntax: `JkAsfTomcat On | Off`

Default: `JkAsfTomcat On`

Context: `server config`

Override: `none`

Origin: `iSeries`

Usage Considerations: A `LoadModule` is required in the config file prior to using the directive. The statement should be as follows: `LoadModule mod_jk /QSYS.LIB/QHTTPSVR.LIB/QZTCJK.SRVPGM`

Example: `JkAsfTomcat Off`

The `JkAsfTomcat` directive allows ASF Tomcat to be turned off without deleting particular ASF Tomcat directives from the HTTP Server (powered by Apache) configuration file.

Parameter: `On | Off`

- When set to `On`, ASF Tomcat is enabled.
- When set to `Off`, it appears to the user as if ASF Tomcat was never enabled.

URL requests are not redirected to the ASF Tomcat servlet engine.

JkLogFile:

Module: `mod_jk`

Syntax: `JkLogFile directory-filename`

Default: `none`

Context: `server config`

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 mod_jk /QSYS.LIB/QHTTPSVR.LIB/QZTCJK.SRVPGM`

Example: `JkLogFile /QIBM/UserData/HTTPPA/tomcat/logs/tomcat.log`

The `JkLogFile` directive is used to describe the full path and file name of the `mod_jk` log file. For example:

```
JkLogFile /QIBM/UserData/HTTPPA/tomcat/logs/tomcat.log
```

The log file describes the flows of header and data between the HTTP Server (powered by Apache) and the ASF Tomcat servlet engine. It does not contain information relative to what happens after a request is forwarded to the servlet engine. This information is obtained by looking at the ASF Tomcat servlet engine log files. The specified log file is never purged or wrapped. The file may need to be periodically purged by the administrator.

Parameter: *directory-filename*

- The *directory-filename* parameter specifies the directory path and filename for the log file that describes the flows of header and data between the HTTP Server (powered by Apache) and the ASF Tomcat servlet engine.

The directive supports both the Root and QOpenSys file systems.

JkLogLevel:

Module: mod_jk

Syntax: JkLogLevel *log-level*

Default: JkLogLevel emerg

Context: server config

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 mod_jk /QSYS.LIB/QHTTPSVR.LIB/QZTCJK.SRVPGM

Example: JkLogLevel error

The JkLogLevel directive is used to describe what detail of logging should occur to the log file defined by JkLogFile.

Parameter: *log-level*

- The *log-level* parameter describes what detail of logging should occur to the log file defined by JkLogFile. The possible log level values are:
 - debug
 - info
 - error
 - emerg

These values correspond to the HTTP Server (powered by Apache) logging level descriptions.

JkMount:

Module: mod_jk

Syntax: JkMount *URL-path workername*

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 mod_jk /QSYS.LIB/QHTTPSVR.LIB/QZTCJK.SRVPGM

Example: JkMount /servlet/* workerajp12

The JkMount directive specifies which URI contexts are sent to a ASF Tomcat worker. The mod_jk module supplies the code to enable the different workers.

Parameter One: *URL-path*

- The *URL-path* parameter used by this directive is compared to the incoming URL from the client browser to determine if the client URL must be forward to the ASF Tomcat servlet engine.

Parameter Two: *workername*

- The *workername* can include the following characters: alpha, numeric, dash (-), underscore (_), percent (%), and question mark (?). The JkWorkersFile directive points to a file that provides a mapping between a worker name and a valid worker type. The worker type determines whether the worker is an in-process or out-of-process worker.

The directive URL path can assume three different forms: exact match, context match and suffix match. All URL path forms must be preceded by a forward slash (/) character (for instance, /Example, /Example/* or /Example/*.jsp).

- Exact match is used when JkMount must match exactly to a URL path in the client URL. For example, for an incoming client URL of `http://systemname:port/example/SampleServlet`, JkMount identifies the URL path then the HTTP Server (powered by Apache) forwards the URL request to the ASF Tomcat worker `jniworker`. The directive displays in the form:

```
JkMount /example/SampleServlet jniworker
JkMount /SampleServlet jniworker
```

- Context match forwards client URL requests that have URL paths matching the directive URL path. This form supports the use of the wildcard character (*). For example, with an incoming client URL of `http://systemname:port/server1/example/samples/ExampleServlet`, JkMount identifies the URL path then the HTTP Server (powered by Apache) forwards all URL requests with a */example* subcontext to the ASF Tomcat worker **System1worker**. The directive displays in the form:

```
JkMount /example/* System1worker
```

- Suffix match forwards client URL requests that have a particular suffix. The form is used to support JSPs that are given in the client URL with a suffix of `.jsp`. For example:

```
JkMount/*.jsp System1worker
```

All client URL requests that have a `.jsp` suffix are forwarded to worker `System1worker` for processing.

JkMountCopy:

Module: `mod_jk`

Syntax: `JkMountCopy On | Off`

Default: `JkMountCopy Off`

Context: 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 mod_jk /QSYS.LIB/QHTTPSVR.LIB/QZTCJK.SRVPGM`

Example: `JkMountCopy On`

The JkMountCopy directive indicates whether the base server mount points should be copied to the virtual server. Any mount points defined outside `<VirtualHost>` `</VirtualHost>` are inherited by the virtual host.

Parameter: `On | Off`

- When set to *On*, the base server mount points are copied to the virtual server.
- When set to *Off*, the base server mount points are not copied to the virtual server.

JkWorkersFile:

Module: `mod_jk`

Syntax: `JkWorkersFile directory-filename`

Default: none

Context: server config

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 mod_jk /QSYS.LIB/QHTTPSVR.LIB/QZTCJK.SRVPGM

Example: JkWorkersFile /home/Tomcat/conf/workers.properties

The JkWorkersFile directive is used to define the name of a file that contains configuration information (that describes how mod_jk module attaches to the ASF Tomcat servlet engine). The two interfaces, used to support the connection between HTTP Server (powered by Apache) and ASF Tomcat, are JNI (Java Native Interface) and sockets connections. The JNI interface is used for an in-process ASF Tomcat server. An in-process ASF Tomcat server will startup at the same time as the HTTP Server (powered by Apache). The sockets interface is for use with an out-of-process ASF Tomcat server. With an out-of-process ASF Tomcat server, the server could be on the same system or on a different system.

Parameter: *directory-filename*

- The *directory-filename* parameter defines the directory path and filename for the file that contains configuration information (that describes how mod_jk module attaches to the ASF Tomcat servlet engine).

There is no default; this directive must be specified or ASF Tomcat will not function. The typical file name is workers.properties.

This directive supports both the Root and QOpenSys file systems.

Module mod_log_config for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 (powered by Apache)” on page 29 for information on the log file formats supported on HTTP Server (powered by Apache).

Use with virtual hosts

If a “<VirtualHost>” on page 573 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 38 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 651
- “LogFormat” on page 653
- “TransferLog” on page 654

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 a CCSID of DefaultNetCCSID for IFS.

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 affect 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 556 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 556 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 556 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/Qcymmddhh.MBR.

"LogCycle" on page 556 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 556 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 38 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: iSeries

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 a CCSID of DefaultNetCCSID for IFS.

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 FSCSID 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 556 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 556 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 556 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/Qcymmddhh.MBR`.

`"LogCycle"` on page 556 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/Qcymmdd`.

If `"LogCycle"` on page 556 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 QHTTPSVR/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 (powered by Apache)" on page 29 for information on the log file formats supported on HTTP Server (powered by Apache).

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 CustomLog will be created as a CCSID of DefaultNetCCSID for IFS.

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 the FSCSID that is 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 556 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 556 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 556 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 556 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 38 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 649 in place of TransferLog.

Module mod_log_io for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.


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 for HTTP Server (powered by Apache)" on page 668 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 current list is maintained at <http://www.isi.edu/in-notes/iana/assignments/media-types/media-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 (powered by Apache) 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"` on page 658
- `"AddClient"` on page 658
- `"AddEncoding"` on page 659

- “AddHandler” on page 659
- “AddInputFilter” on page 660
- “AddLanguage” on page 661
- “AddOutputFilter” on page 661
- “AddType” on page 662
- “DefaultLanguage” on page 662
- “ModMimeUsePathInfo” on page 662
- “MultiviewsMatch” on page 663
- “RemoveCharset” on page 664
- “RemoveClient” on page 664
- “RemoveEncoding” on page 665
- “RemoveHandler” on page 665
- “RemoveInputFilter” on page 666
- “RemoveLanguage” on page 666
- “RemoveOutputFilter” on page 666
- “RemoveType” on page 667
- “SuffixCaseSense” on page 667
- “TypesConfig” on page 668

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 for HTTP Server (powered by Apache)” on page 668 for more information.

AddClient:

Module: mod_mime

Syntax: AddClient *user-agent extension*

Default: none

Context: server config, virtual host, directory, .htaccess

Override: none

Origin: iSeries

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 (powered by Apache) 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 (powered by Apache)" on page 755 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 for HTTP Server (powered by Apache)" on page 668 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 for HTTP Server (powered by Apache)" on page 668 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: iSeries

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 658 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: iSeries

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 iSeries 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 567. This restricts the file to the IFS file system.

Module mod_negotiation for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 561) where the server does an implicit filename pattern match and makes a choice from the results.

See “Content negotiation for HTTP Server (powered by Apache)” on page 11 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>Parameter One: <i>level</i></p> <ul style="list-style-type: none"> • The <i>level</i> parameter is an integer specifying the version of the media type. For text/html, this defaults to '2', otherwise '0'. <p>Parameter Two: <i>qs</i></p> <ul style="list-style-type: none"> • 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
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 671

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 for HTTP Server (powered by Apache)

This module contains directives that define support for the HTTP Proxy function.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries. It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 (powered by Apache)” on page 31.

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 673
- “<Proxy>” on page 674
- “ProxyBadHeader” on page 676
- “ProxyBlock” on page 676
- “ProxyCacheOnly” on page 677
- “ProxyDomain” on page 678
- “ProxyErrorOverride” on page 679
- “<ProxyMatch>” on page 680
- “ProxyMaxForwards” on page 681
- “ProxyNoCache” on page 683
- “ProxyNoConnect” on page 684
- “ProxyPass” on page 685
- “ProxyPassReverse” on page 687
- “ProxyPreserveHost” on page 688
- “ProxyReceiveBufferSize” on page 689
- “ProxyRemote” on page 690
- “ProxyRequests” on page 691
- “ProxyReverse” on page 692
- “ProxyTimeout” on page 692
- “ProxyVia” on page 693

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/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: 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: iSeries

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) 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.

<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 "^/v[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 "iSeries 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 685 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 539).

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: iSeries

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* | !]

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/
```

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 692).

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” may be used to handle HTTP redirect responses from remote servers.
- Setting “ProxyReverse” on page 692 to *off* negates this directive.
- Setting “ProxyRequests” on page 691 to *off* does not negate this directive. It is available regardless of the forward proxy state.

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 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. The server adjusts the values for URI, Location, and Content-Location response headers according to the specified values.

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 685 directive.
- Setting "ProxyReverse" on page 692 to *off* negates this directive.
- Setting "ProxyRequests" on page 691 to *off* does not negate this directive. It is available regardless of the proxy state.

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 687 may be used to handle HTTP redirect responses from remote servers.
- Setting “ProxyReverse” on page 692 to *off* negates this directive.
- Setting “ProxyRequests” on page 691 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/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: 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 676 may be used to block incoming requests prior to consideration for this directive.
- Requests matching a “NoProxy” on page 673 directive are not chained.
- This directive is commonly used in conjunction with “NoProxy” on page 673 and “ProxyDomain” on page 678 for directing proxy requests within intranets.
- Setting ProxyNoConnect to on negates this directive.

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 684 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 514 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 691 to *off* negates all directives for the mod_proxy module, except for the “ProxyPass” on page 685 and “ProxyPassReverse” on page 687 directives.

Note: CacheRoot is required when ProxyNoConnect is set to *on*.

ProxyReverse:

Module: mod_proxy

Syntax: ProxyReverse *on* | *off*

Default: ProxyReverse *on*

Context: server config, virtual host

Override: none

Origin: iSeries

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 685 and “RewriteRule” on page 702 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 685 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/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: 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 571 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 687 may be used to handle HTTP redirect responses from remote servers.
- Setting “ProxyReverse” on page 692 to *off* negates this directive.
- Setting “ProxyRequests” on page 691 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 (powered by Apache) 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 690 for more details about proxy chains).

Module `mod_rewrite` for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries. It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 695
- "RewriteEngine" on page 699
- "RewriteLog" on page 699
- "RewriteLogLevel" on page 699
- "RewriteMap" on page 700
- "RewriteOptions" on page 701
- "RewriteRule" on page 702

RewriteBase:

Module: `mod_rewrite`

Syntax: RewriteBase *Base_URL*
Default: RewriteBase physical directory path
Context: directory, .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 700 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 on HTTP Server” on page 766 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.

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
<code>^/somepath(*) otherpath\$1</code>	not supported
<code>^/somepath(*) otherpath\$1 [R]</code>	not supported
<code>^/somepath(*) otherpath\$1 [P]</code>	not supported
<code>^/somepath(*) /otherpath\$1</code>	<code>/otherpath/pathinfo</code>
<code>^/somepath(*) /otherpath\$1 [R]</code>	<code>http://thishost/otherpath/pathinfo</code> via external redirection
<code>^/somepath(*) /otherpath\$1 [P]</code>	not supported
<code>^/somepath(*) http://thishost/otherpath\$1</code>	<code>/otherpath/pathinfo</code>
<code>^/somepath(*) http://thishost/otherpath\$1 [R]</code>	<code>http://thishost/otherpath/pathinfo</code> via external redirection
<code>^/somepath(*) http://thishost/otherpath\$1 [P]</code>	not supported
<code>^/somepath(*) http://otherhost/otherpath\$1</code>	<code>http://otherhost/otherpath/pathinfo</code> via external redirection
<code>^/somepath(*) http://otherhost/otherpath\$1 [R]</code>	<code>^/somepath(*) http://otherhost/otherpath\$1 [R]</code>
<code>^/somepath(*) http://otherhost/otherpath\$1 [P]</code>	<code>http://otherhost/otherpath/pathinfo</code> via internal proxy
<code>^localpath(*) otherpath\$1</code>	<code>/somepath/otherpath/pathinfo</code>
<code>^localpath(*) otherpath\$1 [R]</code>	<code>http://thishost/somepath/otherpath/pathinfo</code> via external redirection
<code>^localpath(*) otherpath\$1 [P]</code>	not supported
<code>^localpath(*) /otherpath\$1</code>	<code>/otherpath/pathinfo</code>

Given rule	Resulting substitution
^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 for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 707
- “SetEnvIf” on page 708
- “SetEnvIfNoCase” on page 709

BrowserMatch:

Module: mod_setenvif

Syntax: BrowserMatch *regex* *envar*[=*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 on HTTP Server” on page 766 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.” BrowserMatch and “BrowserMatchNoCase” are special cases of “SetEnvIf” on page 708 and “SetEnvIfNoCase” on page 709. 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 on HTTP Server” on page 766 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 706. However, it provides for case-insensitive matching. For example:

```
BrowserMatchNoCase mac platform=ibm
BrowserMatchNoCase win platform=windows
```

“BrowserMatch” on page 706 and BrowserMatchNoCase are special cases of “SetEnvIf” and “SetEnvIfNoCase” on page 709. 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 on HTTP Server” on page 766 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 on HTTP Server” on page 766 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 708, 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 on HTTP Server” on page 766 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** for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Summary

The module `mod_so` provides for loading of executable code and modules into the HTTP Server at startup or restart time. On the iSeries 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 iSeries file.

Parameter Two: *filename*

- The *filename* parameter must be an iSeries 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** for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Summary

The module `mod_userdir` provides for user-specific directories.

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 iSeries 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 iSeries 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 38 for more information.

Module `mod_usertrack` for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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”
- “CookieName” on page 713
- “CookieStyle” on page 713
- “CookieTracking” on page 713

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 (powered by Apache) 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` for HTTP Server (powered by Apache)

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 (powered by Apache) 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 (powered by Apache).

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/l/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/
```

There are examples of more complicated VirtualDocumentRoot settings in the mod_vhost_alias documentation. Using 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>
```

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 572 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 number.
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 653 for more information.

Directives

- “`VirtualDocumentRoot`”
- “`VirtualDocumentRootIP`” on page 720
- “`VirtualScriptAlias`” on page 721
- “`VirtualScriptAliasIP`” on page 722

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 538 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" 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
# all the preliminary stuff is the same as above, then
# 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 572 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.

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.

Parameter: *interpolated-directory*

- The *interpolated-directory* parameter the full path to a directory.

```
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 719 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.

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.

Parameter: *interpolated-directory*

- The *interpolated-directory* parameter the full path to a directory.

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 719 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 721.

The VirtualScriptAliasIP directive is like the “VirtualScriptAlias” on page 721 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.

Common Gateway Interface

This topic provides information about Common Gateway Interfaces (CGI).

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Overview of CGI

This topic provides information about CGI. Common Gateway Interface (CGI) is a standard, supported by almost all web servers, that defines how information is exchanged between a web server and an external program (CGI program).

The CGI specification dictates how CGI programs get their input and how they produce any output. CGI programs process data that is received from browser clients. For example, the client fills out a form and sends the information back to the server. Then the server runs the CGI program.

Programs that are called by the server must conform to the server CGI interface in order to run properly. We will describe this in further detail later in this chapter.

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. Depending on the server directives, the server calls that program on behalf of the client browser.

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

You need to compile programs that are written in programming languages. Compiled programs typically run faster than programs that are written in scripting languages. On the other hand, those programs that are written in scripting languages tend to be easier to write, maintain, and debug.

The functions and tasks that CGI programs can perform range from the simple to the very advanced. In general, we call those that perform the simple tasks CGI scripts because you do not compile them. We often call those that perform complex tasks gateway programs. In this manual, we refer to both types as *CGI programs*.

Given the wide choice of languages and the variety of functions, the possible uses for CGI programs seem almost endless. How you use them is up to you. Once you understand the CGI specification, you will know how servers pass input to CGI programs and how servers expect output.

There are many uses for CGI programs. Basically, you should design them to handle dynamic information. Dynamic in this context refers to temporary information that is created for a one-time use and not stored as a static Web page. This information may be a document, an e-mail message, or the results of a conversion program.

For detailed information about CGI APIs, see Chapter 8, “HTTP Server Application Programming Interfaces” on page 51.

CGI and Dynamic Documents: There are many types of files that exist on the web. Primarily they fall into one of the following categories:

- Images
- Multimedia

- Programs
- HTML documents

Servers break HTML documents into two distinct types:

- Static
- Dynamic

Static documents exist in non-changing source form on the web server. You should create *Dynamic documents* as temporary documents to satisfy a specific, individual request.

Consider the process of Δ serving Δ these two types of documents. Responding to requests for static documents is fairly simple. For example, Jill User accesses the Acme web server to get information on the Pro-Expert gas grill. She clicks on Products, then on Grills, and finally on Pro-Expert. Each time Jill clicks on a link, the web browser uses the URL that is attached to the link to request a specific document from the web server. The server responds by sending a copy of the document to Jill's browser.

What if Jill decides that, she wants to search through the information on the Acme web server for all documents that contain information on Acme grills? Such information could consist of news articles, press releases, price listings, and service agreements. This is a more difficult request to process. This is not a request for an existing document. Instead, it is a request for a dynamically generated list of documents that meet certain criteria. This is where CGI comes in.

You can use a CGI program to parse the request and search through the documents on your web server. You can also use it to create a list with hypertext links to each of the documents that contain the specified word or string.

Uses for CGI: HTML allows you to access resources on the Internet by using other protocols that are specified in the URL. Examples of such protocols are `mailto`, `ftp`, and `news`. If you code a link with `mailto` that is followed by an e-mail address, the link will result in a generic mail form.

What if you wanted your customers to provide specific information, such as how often they use the web? Or how they heard about your company? Rather than using the generic `mailto` form, you can create a form that asks these questions and more. You can then use a CGI program to interpret the information, include it in an e-mail message, and send it to the appropriate person.

You do not need to limit CGI programs to processing search requests and e-mail. You can use them for a wide variety of purposes. Basically, anytime you want to take input from the reader and generate a response, you can use a CGI program. The input may even be apparent to the reader. For example, many people want to know how many other people have visited their home page. You can create a CGI program that keeps track of the number of requests for your home page. This program can display the new total each time someone links to your home page.

Environment variables

Before you begin writing your CGI program, you need to understand the format in which the server will pass the data. The server receives the URL-encoded information and, depending on the type of request, passes the information to the CGI program. The server does this by using environment variables, command line arguments, or standard input.

A CGI application program should be able to handle a NULL value when getting an environment variable. For example, when the CGI program is trying to do a `getenv(ΔCONTENT_LENGTHΔ)` and the method is GET, the value would be returned NULL. This is because `CONTENT_LENGTH` is only defined in method POST (to describe the length of standard input).

The supported environment variables for HTTP Server are can be found in the IBM iSeries Information Center. The environment variables have been divided into two groups: Non-SSL and SSL and by HTTP Server type.

Requests from Standard Search (ISINDEX) Documents: *ISINDEX* is an HTML tag that identifies the document as a standard search document and causes the browser to automatically generate an entry field. When information is sent from an ISINDEX document, the server takes the appended data (the information following the ?), breaks it at the pluses (+), and sends the data to the CGI program as command line arguments (argv). For example:

```
<ISINDEX>
```

Note: It is possible to write CGI programs that display all environment variables. At times these variables may include sensitive data such as user IDs and passwords for various products. Consequently, you must be careful about displaying environment variables in your CGI programs, and you must be careful about who has access to them.

CGI program considerations for HTTP Server (original)

You need to understand that the security environment defined by the server configuration directives that apply to your CGI programs. If the CGI program is covered by a protection directive that calls for basic authentication, the user must supply a user ID and password before the CGI program is allowed to run.

The other protection subdirectives determine the following:

- How the server validates the user ID and password
- What security environment the CGI program runs in

The subdirectives might tell the browser to treat the user ID as a user profile and to validate the password against it. In addition, the Userid subdirective might be used to cause the server job to run under a specified user profile or the one the user entered. The following example protection setup would cause the user ID to be treated as a user profile, and to switch to that profile when starting the CGI program:

```
Protection example1 {  
  AuthType Basic  
  Userid %%CLIENT%%  
  PasswdFile %%SYSTEM%%  
}
```

If Userid %%SERVER%% had been specified, the CGI program will run under the QTMHHTTP1 user profile. If Userid FRED had been specified, the CGI program would run under the FRED user profile.

Alternatively, the PasswdFile subdirective can identify a validation list. For example:

```
PasswdFile qgpl/valist1
```

Internet User Lists contain a set of user IDs, their associated password, and optionally other application-specific information. In this example, the server would authenticate the user by comparing the specified user ID and password against the specified Internet User List. If the user ID exists in the Internet User List and the password matches, the CGI program would run under the QTMHHTTP1 user profile.

Internet User Lists can be created through the CRTVLDL command. CGI or other programs can add, remove, find, or change entries through a set of APIs documented in the programming topic in the iSeries 400 Information Center. By using Internet User Lists, the CGI program can “register” users and associate other information with each entry while at the same time using the basic authentication functions of the server to authenticate requests.

Guidelines for writing cluster-enabled CGIs

By understanding the guidelines for writing cluster-enabled CGIs, you can avoid potential problems with your CGI code.

All CGI programs existing today are cluster-disabled. A CGI program developer should follow the following rules when writing cluster-enabled 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.
- Use only cluster-safe mechanisms. For example, if a CGI program uses shared memory (which is currently not a cluster-safe mechanism), the macro is almost certainly cluster-disabled.

Table 23. 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` API. To retrieve any stateful data that has been stored use the `QzhbCgiRecvState` API. The `QzhbCgiRecvState` 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` 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` API and restore the static variables.

The Web server still limits the total number of persistent CGIs (both cluster-enabled and cluster-disabled) using the `MaxPersistentCGI` directive.

For the Peer model, follow these additional guidelines:

The stateful data is saved by the high availability CGI program by calling the `QzhbCgiSendState` API. To retrieve any stateful data that has been stored use the `QzhbCgiRecvState` API. The `QzhbCgiRecvState` 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. Consider a cluster-enabled persistent CGI program. The concept of persistency is kept, but its implementation is absolutely changed. A cluster-enabled persistent CGI program does not stay in the CGI job during a session; it is invoked each time a request comes in.

However, the concept of persistency is kept due to CGI's state (and the state of the Web server regarding the CGI) stored in the CHT and always passed to the CGI program with subsequent requests.

Overview of High Availability CGI

High availability CGI programs use APIs to preserve the CGI's state between successive client requests. A High availability CGI program stores its state data on the Web server and can retrieve its state even after a failure or switchover of the HTTP Server or system.

Although maintaining a CGI program's state 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 significantly different and a High Availability CGI program should not be confused with a Persistent CGI program.

During the configuration of an 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. 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 Web server associates a HA 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 HA CGI program to be returned to the client.

An HA CGI program will use two APIs to maintain its state. To store state information on the Web server, the CGI calls the API `QzhhCgiSendState`. To receive its previous state from the Web server, the CGI should call the API `QzhhRecvState`.

Using directives for security and access control for HTTP Server (original)

The server administrator controls the behavior of the server. The server will not do anything that the server administrator has not explicitly configured it to do.

Several features of the server ensure that the administrator maintains this control:

- The default fail rule means that only requests that are authorized by the web administrator are honored; other requests will fail.
- Explicit CGI enablement means that no CGI programs will run unless specifically authorized.
- Only CGI programs are run.
- Only the read HTTP methods GET, POST, and HEAD are supported.

The default fail rule: The server rejects, by default, all incoming requests unless the URL, as translated by any preceding Map directives, matches a Pass, Redirect, or Exec directive that has been explicitly coded by the server administrator:

- A match with a Pass directive enables the server to serve a document.
- A match with a Redirect directive causes the server to return a 302 response, found in the HTTP response to the client application. This HTTP response header field contains a location with the redirect request. The HTTP request that matches a Redirect directive causes no data to be accessed. A subsequent request generated by a client could cause data to be accessed.
- A match with an Exec directive enables the server to run a CGI program on behalf of the client.
- A match with a Service directive enables the server to run a server API program on behalf of the client.

Explicit CGI enablement: The server will not run a user-defined CGI program unless the server administrator has explicitly enabled it by coding an Exec directive. The server administrator can, for example, limit CGI requests to a specific library in QSYS.LIB.

Important: It is the server administrator's responsibility to verify that any CGI program that is enabled does not violate the customer's security policies for the system on which the server is running. IBM recommends that the HTTP administrator move the DB2WWW *PGM (the Net.Data CGI program) from the QHTTPSVR library to its own CGI library. This allows users to run the CGI program while limiting access to the QHTTPSVR library. Do not move any Include files from the QHTTPSVR library.

Server runs only CGI programs: To run properly, programs that are called by the server must conform to the server CGI interface. When the server is enabled to call a particular program on behalf of a remote HTTP client application, the program is called and the output is returned through the server CGI interface.

Overview of Persistent CGI

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-HTSession␣` 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.

Named Activation Groups: CGI programs can be built using named activation groups by specifying a name on the ACTGRP parameter of the CRTPGM or CRTSRVPGM commands. In doing this, the initial call to the program within the job will still have the startup cost of activating the program. However, an activation group is left active after the program has exited normally. All storage associated with that program is still allocated and in `␣last-used␣` state. The program is not initialized when it is called again. In addition, for the ILE C runtime, all settings are in `␣last-used␣` state, such as `signal()`, `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. All ILE languages running on the server can use this mechanism to enable persistence for their CGI programs.

For additional information about activation groups see the, ILE Concepts, SC41-5606 book.

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.

HTTTimeout CGI Header: The *HTTTimeout* 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.

```
HTTTimeout = "HTTTimeout" ":" minutes
```

The *time-out* value is a non-negative decimal integer, representing the time in minutes. This header must be preceded by an `␣Accept-HTTPSession␣` 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 time-out, 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, Country, 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 program example located under **Developer Resources**, <http://www.ibm.com/eserver/iseres/http> , displays a counter that is increased each time the Persistent CGI program is called.

Common gateway interface (CGI) programs for the HTTP Server

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Common Gateway Interface (CGI) is a standard that defines how information is exchanged between a Web server and an external program (CGI program).

The functions and tasks that CGI programs can perform range from the simple to the very advanced and come in two different types, CGI script and gateway program. CGI scripts are used to perform simple tasks and do not require to be compiled. The more complex tasks are most commonly referred to as gateway programs. In general, both types are referred to as CGI programs.

Regardless of the type of CGI program, both are designed to handle dynamic information. Dynamic, in this context, refers to temporary information that is created for a one-time use and not stored as a static Web page. This information may be a document, an e-mail message, or the results of a conversion program. You must compile gateway programs before using them. Compiled programs typically run faster than programs that are written in scripting languages. On the other hand, those programs that are written in scripting languages tend to be easier to write, maintain, and debug.

The CGI specification dictates how CGI programs receive input and how output is produced. A common use of CGI programs is to process data that is received from browser clients. For example, the client fills out a form and sends the information back to the server. Then the server runs the CGI program. Programs that are called by the server must conform to the server CGI interface in order to run properly.

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. Depending on the server directives, the server calls that program on behalf of the client browser. The server supports CGI programs that are written in C++, REXX, Java, ILE C, ILE RPG, and ILE COBOL. It also supports multi-thread CGI programs in all of these languages capable of multiple threads.

For detailed information about CGI APIs, see “Application programming interfaces (APIs) for HTTP Server” on page 218.

CGI and dynamic documents

There are many types of files that exist on the Web. Primarily they fall into one of the following categories:

- images
- multimedia
- programs
- HTML documents

Servers break HTML documents into two distinct types:

- static
- dynamic

Static documents exist in non-changing source form on the web server. You should create *Dynamic documents* as temporary documents to satisfy a specific, individual request.

Consider the process of $\hat{\text{-serving}}$ these two types of documents. Responding to requests for static documents is fairly simple. For example, Jill User accesses the Acme web server to get information on the Pro-Expert gas grill. She clicks on Products, then on Grills, and finally on Pro-Expert. Each time Jill clicks on a link, the web browser uses the URL that is attached to the link to request a specific document from the web server. The server responds by sending a copy of the document to Jill’s browser.

What if Jill decides that she wants to search through the information on the Acme web server for all documents that contain information on Acme grills? Such information could consist of news articles, press releases, price listings, and service agreements. This is a more difficult request to process. This is

not a request for an existing document. Instead, it is a request for a dynamically generated list of documents that meet certain criteria. This is where CGI comes in.


CGI programs may be used to parse the request and search through the documents on your web server. You can also use it to create a list with hypertext links to each of the documents that contain the specified word or string.

Uses for CGI

HTML allows you to access resources on the Internet by using other protocols that are specified in the URL. Examples of such protocols are `mailto`, `ftp`, and `news`. If you code a link with `mailto` that is followed by an e-mail address, the link will result in a generic mail form.

What if you wanted your customers to provide specific information, such as how often they use the web? Or how they heard about your company? Rather than using the generic `mailto` form, you can create a form that asks these questions and more. You can then use a CGI program to interpret the information, include it in an e-mail message, and send it to the appropriate person.

You do not need to limit CGI programs to processing search requests and e-mail. You can use them for a wide variety of purposes. Basically, anytime you want to take input from the reader and generate a response, you can use a CGI program. The input may even be apparent to the reader. For example, many people want to know how many other people have visited their home page. You can create a CGI program that keeps track of the number of requests for your home page. This program can display the new total each time someone links to your home page.

For general information on CGI programs see <http://hoohoo.ncsa.uiuc.edu/cgi/> .

The CGI Process

Usually CGI programs are referred to from within HTML documents. In general, the HTML document format defines the environment variables that specify the passing of information. When you design the layout of your HTML document, you must keep in mind how a CGI program might affect the look of your document. Developing the CGI program along with the HTML document will help you avoid many design mistakes.

Overview: The CGI process involves three players: The web browser, the web server, and the CGI program. To exemplify how CGI programs for online forms work, let us assume that the web browser has already requested and obtained an HTML form.

1. The user clicks buttons or enters information in fields, and then clicks on an HTML button to submit the request.
2. The web browser then sends the data to the web server in an encoded format. For example, the data might consist of responses on an HTML form.
3. When the web server receives data, it converts the data to a format compliant with the CGI specification for input and sends it to the CGI program.
4. The CGI program then decodes and processes the data.
5. The system sends this response back to the web server in a form that is compliant with the CGI specification for output.
6. The web server then interprets the response and forwards it to the web browser.

Note: If a CGI application program must change HTTP Server job attributes while processing, the CGI program must restore the attributes to their initial values before the CGI program ends. Failure to restore job attributes that are changed in the CGI program will result in unpredictable responses to future server requests. For example, a CGI program that requires a library in the library list needs to add the library to the library list. The CGI program **must** remove the library list before the CGI program ends.

The following HTML form illustrates the various types of fields:

Note: The CGIXMP.EXE program referred to in this sample is just an example; it is not shipped with the server product.

```
<HTML>
<HEAD>
<TITLE>CGIXMP Test Case</TITLE>
</HEAD>
<BODY>
<H1>CGI Sample Test Case</H1>
Fill in the following fields and press APPLY.
The values you enter will
be read by the CGIXMP.EXE program and displayed in a simple HTML
form which is generated dynamically by the program.
<P> <HR>
<form method=POST action="/cgi-bin/cgixmp">
<P>
<H3>Checkbox Field</H3>
<P>
<PRE>
<input type="checkbox" name="var1" value="123">
Check to set variable VAR1 to 123<BR>
<input type="checkbox" name="var2" value="XYZ" checked>
Check to set variable VAR2 to XYZ<BR>
</PRE>
<P>
<H3>Radio Button Field</H3>
<P>
<PRE>
<input type="radio" name="var3" value="1">
Select to set variable VAR3 to 1<BR>
<input type="radio" name="var3" value="2">
Select to set variable VAR3 to 2<BR>
<input type="radio" name="var3" value="3" checked>
Select to set variable VAR3 to 3<BR>
<input type="radio" name="var3" value="4">
Select to set variable VAR3 to 4<BR>
</PRE>
<P>
<H3>Single selection List Field</H3>
<P>
<PRE>
Select a value for variable VAR4 <select size=1 name="var4">
<option>0<option>1<option>2<option>3
<option>4<option>5</select>
</PRE>
<P>
<H3>Text Entry Field</H3>
<P>
<PRE>
Enter value for variable VAR5 <input type="text" name="var5"
size=20 maxlength=256 value="TEST value">
</PRE>
<P>
<H3>Multiple selection List Field</H3>
<P>
<PRE>
Select a value for variable VAR6
<select multiple size=2 name="var6">
<option>Ford<option>Chevrolet<option>Chrysler<option>
Ferrari<option>Porsche
</select>
</PRE>
<P>
<H3>Password Field</H3>
<P>
```

```

<PRE>
Enter Password
<input type="password" name="pword" size=10 maxlength=10>
</PRE>
<P>
<H3>Hidden Field</H3>
<P>
<input type="hidden" name="hidden" value="Text not shown on form...">
<P>
<PRE>
<input type="submit" name="pushbutton" value="Apply">
<input type="reset" name="pushbutton" value="Reset">
<HR>
</PRE>
</FORM>
</BODY>
</HTML>

```

Sending Information to the Server: When you fill out a form, the web browser sends the request to the server in a format that is described as *URL-encoded*. The web browser also performs this function whenever you enter a phrase in a search field and click on the submission button. In URL-encoded information:

- The URL starts with the URL of the processing program.
- A question mark (?) precedes attached data, such as name=value information from a form, which the system appends to the URL.
- 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.

Note: The method attribute specifies how the server sends the form information to the program. You use the GET and POST methods in the HTML file to process forms. The *GET* method sends the information through environment variables. You will see the information in the URL after the $\Delta\Delta$ character. The *POST* method passes the data through standard input.

The main advantage of using the GET method is that you can access the CGI program with a query without using a form. In other words, you can create canned queries that pass parameters to the program. For example, if you want to send the previous query to the program directly, you can do the following:

```

<A HREF="/cgi-bin/program.pgm?Name=John&LName=Richard&user=Smith%Company=IBM">
YourCGI Program</a>

```

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 4 KB.

Data Conversions on CGI Input and Output: 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 iSeries 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. You must provide data to a CGI program through environment variables and standard-input (stdin). 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.

For more information related to CCSIDs, national language support, and international application development, see the Globalization topic in the iSeries Information Center.

HTTP Server (powered by Apache): You can configure HTTP Server (powered by Apache) 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

Note: For compatibility, HTTP Server (powered by Apache) also supports HTTP Server (original) modes. If you are writing new CGI programs for HTTP Server (powered by Apache), use BINARY, EBCDIC, or EBCDIC_JCD.

When using one of HTTP Server (original) modes with HTTP Server (powered by Apache), you must specify both the input and output modes in the directive.

CGI Environment Variables: In addition, the system provides the following CGI environment variables to the CGI program:

- CGI_MODE - which input conversion mode the server is using (%%MIXED%%, %%EBCDIC%%, %%BINARY%%, %%EBCDIC_JCD%%, EBCDIC, BINARY, or EBCDIC_JCD)
- CGI_ASCII_CCSD - from which ASCII CCSID was used to convert the data
- CGI_EBCDIC_CCSD - which EBCDIC CCSID the data was converted into
- CGI_OUTPUT_MODE - which output conversion mode the server is using (%%MIXED%%, %%EBCDIC%%, %%BINARY%%, EBCDIC, or BINARY)

The supported environment variables for HTTP Server are can be found in the IBM iSeries Information Center. The environment variables have been divided into two groups: Non-SSL and SSL and by HTTP Server type.

CGI Input Conversion Modes:

Table 24. Conversion action for text in CGI Stdin. This table summarizes the type of conversion that is performed by the server for each CGI mode.

CGI_MODE	Conversion	Stdin encoding	Environment variable	Query_String encoding	argv encoding
BINARY or %%BINARY%%	None	No conversion	FsCCSID	No conversion	No conversion
EBCDIC or %%EBCDIC%%	NetCCSID to FsCCSID	FsCCSID	FsCCSID	FsCCSID	FsCCSID
%%EBCDIC%% or %%EBCDIC_JCD%% with charset tag received	Calculate target EBCDIC CCSID based on received ASCII charset tag	EBCDIC equivalent of received charset	FsCCSID	FsCCSID	FsCCSID
EBCDIC_JCD or %%EBCDIC_JCD%%	Detect input based on received data. Convert data to FsCCSID	Detect ASCII input based on received data. Convert data to FsCCSID.	FsCCSID	Detect ASCII input based on received data. Convert data to FsCCSID.	Detect ASCII input based on received data. Convert data to FsCCSID

Table 24. Conversion action for text in CGI Stdin (continued). This table summarizes the type of conversion that is performed by the server for each CGI mode.

CGI_MODE	Conversion	Stdin encoding	Environment variable	Query_String encoding	argv encoding
%%MIXED%% (Compatibility mode)	NetCCSID to FsCCSID (receive charset tag is ignored)	FsCCSID with ASCII escape sequence	CCSID 37	CCSID 37 with ASCII escape sequence	CCSID 37 with ASCII escape sequence
EBCDIC or EBCDIC_JCD	Calculate target EBCDIC CCSID based on received ASCII charset tag	EBCDIC equivalent of received charset	FsCCSID	EBCDIC equivalent of received charset	EBCDIC equivalent of received charset

HTTP Server (original)

%%MIXED%%

This mode is the default mode of operation for HTTP Server (original). The system converts values for CGI environment variables to EBCDIC CCSID 37, including QUERY_STRING. The system converts stdin data to the CCSID of the job. However, the system still represents the encoded characters "%xx" by the ASCII 819 octet. This requires the CGI program to convert these further into EBCDIC to process the data. For more information, see symptom, Special characters are not being converted or handled as expected in Chapter 6, "Troubleshooting your CGI programs" on page 43.

%%EBCDIC%%

In this mode, the server will convert everything into the EBCDIC CCSID of the job. The server checks the Entity bodies for a charset tag. If found, the server will convert the corresponding ASCII CCSID to the EBCDIC CCSID of the job. If the server does not find a charset tag, it uses the value of the DefaultNetCCSID configuration directive as the conversion CCSID. In addition, the system converts escaped octets from ASCII to EBCDIC, eliminating the need to perform this conversion in the CGI program.

%%BINARY%%

In this mode, QueryString and stdin data is left in ASCII as it was sent by the browser. The CGI program must be written to convert all data as needed. All other environment variables are in the DefaultFSCCID, the default CCSID of the job.

%%EBCDIC_JCD%%

Japanese browsers can potentially send data in one of three code pages, JIS (ISO-2022-JP), S-JIS (PC-Windows), or EUC (UNIX). In this mode, the server uses a well-known JCD utility to determine which codepage to use (if not explicitly specified by a charset tag) to convert stdin data.

HTTP Server (powered by Apache)

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 DefaultFSCCID, the default job CCSID including the name QUERY_STRING =.

EBCDIC

The EBCDIC mode, delivers all of the information to the CGI program in the DefaultFsCCSID, the default CCSID of the job. This mode gives the CGI programmer a more consistent EBCDIC mode than was present in HTTP Server (original). The ASCII CCSID of the QueryString or stdin data is determined from a charset tag on the content type header if present, otherwise the DefaultNetCCSID 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).

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 server instance are overridden. The DefaultFsCCSID directive or the -fscsid startup parameter specifies the default conversions. The startup FsCCSID must be 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-2022JP (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).

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.

Startup FsCCSID Startup NetCCSID 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 FsCCSID, ensure that you are consistent between the FsCCSID and the Content-Type header's CCSID.

CGI Output Conversion Modes: The CgiConv mode includes an output mode. This section explains CGI output conversion modes in more detail.

Table 25. Conversion action and charset tag generation for text in CGI Stdout. This table summarizes the type of conversion that is performed and the charset tag that is returned to the browser by the server.

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 - FsCCSID 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 - FsCCSID 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 - FsCCSID to NetCCSID	None (compatibility mode)
Invalid	CGI error 500 generated by server	

HTTP Server (original) and HTTP Server (powered by Apache) also set an environment variable CGI_OUTPUT_MODE to reflect the setting for the CGI output mode. It contains the CGI output conversion mode the server is using for this request. Valid values are %%EBCDIC%%, %%MIXED%%, %%BINARY%%, EBCDIC, BINARY, or EBCDIC_JCD. The program can use this information to determine what conversion, if any, the server performs on CGI output.

HTTP Server (powered by Apache)

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 in DefaultFsCCSID. 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 DefaultFsCCSID, default CCSID of the job, unless specified otherwise in a charset or CCSID tag by the CGI program.

EBCDIC_JCD

In this mode HTTP header output is in DefaultFsCCSID. 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 DefaultFsCCSID, default CCSID of the job, unless specified otherwise in a charset or CCSID tag by the CGI program.

Returning Output from the Server: When the CGI program is finished, it passes the resulting response to the server by using standard output (stdout). The server interprets the response and sends it to the browser.

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 browser. If the host name is not the local host, the HTTP processes it as a redirect to the web browser. 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>\n  The data entered must be valid numeric digits for id number\n<br></pre>\n</body></html>
```

How CGI Programs Work: Most CGI programs include the following three stages:

- Parsing CGI programs
- Data manipulation within a CGI program
- Response generation by a CGI program

Note: Any CGI program with a name that ends in _nph is considered a no parse header CGI program. This means that the server does no conversions on the data and sends 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: Parsing is the first stage of a CGI program. In this stage, the program takes the data from QUERY_STRING environment variable, command line arguments using argv() or standard input. When the method is GET, the system reads the data from the QUERY_STRING environment variable or command line arguments by using argv(). There is no way to determine the length of data in QUERY_STRING. The system encodes the QUERY_STRING data in the request header.

An example of data read in the QUERY_STRING variable (%%MIXED%% mode):

```
NAME=Eugene+T%2E+Fox&ADDR=etfox%40ibm.net&INTEREST=RCO
```

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.

When the method is post, the system reads the data from standard input. Before the CGI attempts to read standard input, it must check environment variables REQUEST_METHOD and CONTENT_LENGTH. Read standard input only when the REQUEST_METHOD is POST. The read must specify no more than CONTENT_LENGTH bytes. Attempts to specify more than CONTENT_LENGTH bytes on reading standard input are not defined.

Data manipulation: Data manipulation 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.

Response generation: Response generation is the final stage of a CGI program. In this stage, the program formulates its response to the web server, which forwards it to the browser. The response contains MIME headers that vary depending on the type of response. With a search, the response might be the URLs of all the documents that met the search value. With a request that results in e-mail, the response might be a message that confirms that the system actually sent the e-mail.

CGI Program Interface

A Web server passes information to CGI programs using environment variables. If the HTTP Method is Δ POST Δ the CGI also obtains information from the Web Server through stdin. The CGI program uses stdout to send its response back to the Web server. The response consists of a set of headers (such as Content-Length and Content-Type) followed by the response body which is frequently HTML data.

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

The Δ Cluster-Enabled Δ and Δ Accept-HTSession Δ headers should be returned in each response from a High Availability CGI program. Cluster-Enabled:1

An error will result if the Δ Cluster-Enabled Δ header is returned by a CGI program with a value of Δ 1 Δ , 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.

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. 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 iSeries 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

Run-time 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.

***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, ILE Concepts, SC41-5606 book.

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 moving CGI programs that rely on end processing to function 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 Problem Examples:

Note: The following examples are not general CGI programming examples. For general CGI programming examples, see Chapter 7, "Sample programs (in Java, ILE C, and ILE RPG)" on page 49.

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 Δ Document Contains No Data Δ 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 stdio 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");
/*-----*/
}
```

```

/* FIX: 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 with a 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 \triangle Document Contains No Data \triangle 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);
stdinBufferpbytesRead+1p = '\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);
/*-----*/
/* FIX 2: 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);
/*-----*/
/* FIX 3: Free allocated memory. */
/*-----*/
free(stdinBuffer);
} else
printf("Error getting content length\n");
/*-----*/
/* FIX 1: Flush stdout. */
/*-----*/
fflush(stdout);
return;
}

```

Set up CGI programs for HTTP Server (powered by Apache)

This topic describes how to set up your HTTP Server to use CGI programs.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

The following discusses how to set up your HTTP Server (powered by Apache) to use your CGI programs.

CGI settings:

You can set values associated with CGI jobs.

To set CGI settings, do the following:

1. Click the **Manage** tab.
2. Click the **HTTP Servers** subtab.
3. Select your HTTP Server (powered by Apache) 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. Depending on the server area you selected, enter the values associated with your CGI jobs.
9. Click **OK**.

Persistent CGI settings:

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 (powered by Apache) 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. Depending on the server area you selected, enter the values associated with the CGI jobs.
9. Click **OK**.

See the Developer Resources topic at <http://www.ibm.com/eserver/iserries/http>  for sample CGI programs.

CGI in PASE for HTTP Server

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

If your CGI programs currently run on a Unix platform, they may run on the iSeries HTTP Server in PASE. To do this, store your CGI programs in the QOpenSys file system. You should then verify that your program will run in PASE. For more information on how to prepare your code and insure that it will run effectively in PASE, see the Prepare programs to run in PASE 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.

Note: CGI programs that reside in PASE must have file names that do not include the following extensions which are reserved for non-PASE CGI programs:

- .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/

How to enable the server to run CGI programs

The iSeries server stores some CGI programs in QSYS.LIB. You can write the programs in C++, REXX, Java, ILE C, ILE RPG, or ILE COBOL. If the UserID directive is not active, the server profile QTMHHTTP1 needs access to the *PGM object and all objects the program accesses. If the UserID directive is active, the UserID profile needs access to the *PGM object and all objects the program accesses. The Exec directive is required in HTTP Server (original) configuration to run a CGI program on the server. The ScriptAlias directive is required in HTTP Server (powered by Apache) configuration to run a CGI program on the server.

Here is a summary of the steps you need to take to enable your server to run CGI programs:

1. Decide for which CGI mode you will write your program.
2. Write the C++, REXX, Java, ILE C, ILE RPG, or ILE COBOL program.
3. Compile your program, if required.
4. Create a directory structure for your program using MKDIR.
5. Transfer your program to the new directory.
6. If you are using HTML, set the source type using CHGPFM.
7. Create the program object using CRTCMOD and CRTPGM.
8. Ensure that your program has the correct authority using PUBLIC, QTMHHTTP or QTMHHTTP1.
9. Create a new server instance to associate with your program.
10. Make any changes to your server configuration file that are needed. For example, you need to add the Exec or ScriptAlias directive at a minimum.
11. Start or restart the server.
12. Point your web browser to the URL for the HTML document on the server where hostname is the fully qualified host domain name of your server.

`http://hostname/sample`

Note: For REXX programs, you only need to indicate the path and the file name in the EXEC or ScriptAlias directive. REXX CGI execs must reside in database files named REXX or QREXSRC. For example:

```
EXEC /REXX/* /QSYS.LIB/AS400CGI.LIB/QREXSRC.FILE/*
```

Or in HTTP Server (powered by Apache):

```
ScriptAlias /REXX /QSYS.LIB/AS400CGI.LIB/QREXSRC.FILE/*
```

The URL is :

```
http://hostname/REXX/samplecgi.REXX
```

Troubleshooting your CGI programs

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 might be similar to Figure 1:

```
Work with Active Jobs                AS400SYS
                                     10/22/97 16:35:38
CPU %: .4      Elapsed time: 00:02:01   Active jobs: 98
Type options, press Enter.
  2=Change 3=Hold 4=End 5=Work with 6=Release 7=Display message
  8=Work with spooled files 13=Disconnect ...

Opt  Subsystem/Job  User      Type   CPU %  Function      Status
-----
      DEFAULT      QTMHHTTP  BCH    .0     PGM-QZHBHTTP  TIMW
      DEFAULT      QTMHHTTP  BCI    .0     TIMW
      DEFAULT      QTMHHTTP  BCI    .0     TIMW
      DEFAULT      QTMHHTTP  BCI    .0     TIMW

                                           Bottom

Parameters or command
====>
F3=Exit F5=Refresh F10=Restart statistics F11=Display elapsed data
F12=Cancel F23=More options F24=More keys
```

Figure 1. WRKACTJOB SBS(QHTTSPVR) Job(DEFAULT)

To find out if server jobs have ended abnormally, check the spooled files that contain the job logs (QPJOBLOG) for the user profile QTMHHTTP.

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
\n
This data is returned to the client
```

Cause: CGI program caused an exception message that was not handled by the CGI program.

Solution: Look at the active server job logs in Figure 1 on page 747. 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)`

Three active jobs are displayed for this server instance. The first job in the list is always the server instance server job (Function PGM-QZHBHTTP). The second and third jobs in the list are the CGI program jobs. One is for single thread-capable CGI programs, and the other is for multithread-capable CGI programs (Java).

Select **option 10** to display the job log.

If your CGI program is single thread capable (not written in Java language), message HTP2001 will be in the job log. If your CGI program is multithread capable (Java), 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.
The `Pass` and `Exec` directives that request the document and run a CGI program must be in `WRKHTTPCFG` for HTTP Server (original).
- 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) JOBserver_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 403: Forbidden - Path not valid for this server

Whenever a "Forbidden - Path not valid for this server" occurs when running a CGI program, the configuration directives have not been specified correctly.

Cause when a CGI program is requested: When a CGI program is requested on HTTP Server (original), a `Pass` directive appears before an `Exec` directive. For example:

```
Pass /qsys.lib/htmlcgi.lib/*
Exec /qsys.lib/htmlcgi.lib/*
```

In this example any programs in library `htmlcgi` will not run because the `Pass` occurred before the `Exec`. Once a `Pass` condition is true, then the server does not go further.

Solution when a CGI program is requested: The best way to avoid this problem using HTTP Server (original) is to use one of the following:

1. Use Exec and Pass directives with mapping:

```
Pass /doc/* qsys.lib/html.lib/*
Exec /cgi-bin/* qsys.lib/html.lib/*
```

2. Put the CGI programs in a separate library:

```
Exec /qsys.lib/htmlcgi.lib/*
Pass /qsys.lib/htmldoc.lib/html.file/*
```

The best way to avoid the same problem using HTTP Server (powered by Apache) is to use one of the following:

1. Use Exec and Pass directives with mapping:

```
Alias /doc/ /qsys.lib/html.lib/
```

```
<Directory /qsys.lib/html.lib>
  Allow From All
  Order allow,deny
</Directory>
```

```
ScriptAlias /cgi-bin/ /qsys.lib/html.lib/
```

Example 2:

```
Alias /cgi-bin/ /qsys.lib/html.lib/
```

```
<Directory /qsys.lib/html.lib/>
  Allow From All
  order allow,deny
  AddHandler cgi-script .PGM
  Options +ExecCGI
</Directory>
```

2. Put the CGI programs in a separate library:

```
ScriptAlias /qsys.lib/htmlcgi.lib/ /qsys.lib/htmlcgi.lib/
<Directory /qsys.lib/htmlcgi.lib/>
  Allow From All
  order allow,deny
</Directory>

Alias /qsys.lib/htmldoc.lib/ /qsys.lib/html.file/
<Directory /qsys.lib/html.file/>
  Allow From All
  order allow,deny
</Directory>
```

Cause when a document is requested: When a document is requested on HTTP Server (original), an Exec directive appears before a Pass directive. For example:

```
Exec /qsys.lib/html.lib/*
Pass /qsys.lib/html.lib/*
```

In this example any documents in library html will not be found because the Exec occurred before the Pass.

Solution when a document is requested: Change the order of the directives to correct the problem. For example:

```
Pass /qsys.lib/html.lib/*
Exec /qsys.lib/html.lib/*
```

Note: Since the value mapped /qsys.lib/html.lib/ is the same for both Pass and Exec, the combination above would correct a problem with using an incorrect directive. It also would leave a directive in the file that could never be used.

The best way to avoid this problem is to use one of the following:

1. Use Exec and Pass directives with mapping:

```
Pass /doc/* qsys.lib/html.lib/*
Exec /cgi-bin/* qsys.lib/html.lib/*
```

2. Put the CGI programs in a separate library:

```
Exec /qsys.lib/htmlcgi.lib/*
Pass /qsys.lib/htmldoc.lib/html.file/*
```

Symptom: Error 500: Bad script request -- script '/qsys.lib/qsyscgi.lib/progname.pgm' not found or not executable

Cause: Incorrect match of Exec rule.

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: See Description of “Forbidden - Path not valid for this server”.

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%%.

HTTP Server (powered by Apache) and Apache portable runtime application programming interfaces

This topic provides information about the Apache portable runtime (APR) and application programming interfaces (APIs).

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

The HTTP Server (powered by Apache) and Apache portable runtime (APR) application programming interfaces (APIs) can be used to write cross-platform Apache modules. For information on how to compile and configure an HTTP Server (powered by Apache) module, see [Compile and configure modules on HTTP Server \(powered by Apache\)](#).

Links to the IBM HTTP Server (powered by Apache) and APR APIs are listed below. To write, compile, and configure an HTTP Server (powered by Apache) module, you will need to use both APR Core and HTTP Server (powered by Apache) APIs.

- APR Core APIs - The APR APIs are not application specific and may be used with different server types and applications.
- HTTP Server (powered by Apache) APIs - The HTTP Server (powered by Apache) APIs are HTTP Server (powered by Apache) specific and are not part of the APR APIs.

“Module mod_example for HTTP Server (powered by Apache)” on page 595 provides a simple example of the use of the Apache APIs.

For general information on the Apache portable runtime, see <http://apr.apache.org/> .

Regular expression notation for HTTP Server

This topic provides a general overview of regular expression notation.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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 table below 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.

Pattern	Description
\	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.
\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)

Pattern	Examples of strings that match
^192.\.168\.\.10\.\.*\$	(All addresses on class C subnet 192.168.10.0)

Set up third party modules for HTTP Server (powered by Apache)

This topic provides information about how to set up third party modules for your HTTP Server.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

The HTTP Server (powered by Apache) 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 (powered by Apache). Before the module can be used by your HTTP Server (powered by Apache), 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. 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 iSeries (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.)
  LOCALETYPE(*LOCALE)
  SRCSTMF('IFS style path name to the file to be compiled.')
  DEFINE(Preprocessor macros that take affect before the file is compiled.)
  INCDIR('Directory or directories to add to the search path used by the
  compiler to find include files.')
  LANGLVL(*EXTENDED)
```

Note: The LOCALETYPE must have *LOCALE as the value parameter.

At a minimum, you must specify AS400 in the DEFINE field so the compiler will handle AS/400 uniqueness.

At a minimum, you must specify `'/qibm/proddata/httpa/include'` in the INCDIR field. This is the default directory for HTTP Server (powered by Apache) header files.

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 (powered by Apache) APIs that may be used when building the service program.

4. Add LoadModule to HTTP Server (powered by Apache) configuration file

Using the IBM Web Administration for iSeries interface, manually add the LoadModule directive to your HTTP Server (powered by Apache) configuration.

1. Start the IBM Web Administration for iSeries interface.
2. Click the **Manage** tab.
3. Click the **HTTP Servers** subtab.
4. Select your HTTP Server (powered by Apache) from the **Server** list.
5. Expand **Tools**.
6. Click **Edit Configuration File**.
7. Add the following, where **PATH** is the directory in the QSYS directory and **MODULE** is the name of your module:

```
LoadModule Module /QSYS.LIB/PATH/MODULE.SRVPGM
```

Note: Replace Module with the name module you named in CRTSRVPGM under the EXPORT field. Replace the `/DIRECTORY/PATH/MODULE.SRVPGM` with the directory path and the name of your compiled service program.

It is suggested that you place the LoadModule directive as close to the beginning of the configuration file as possible. If the new compiled directive is called before the LoadModule directive, you will have an error.


8. 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.

9. Click **OK**.

When writing your own modules, the same steps above may be used. The Apache Software Foundation (ASF)  provides basic information for writing your own modules.

Handler for HTTP Server (powered by Apache)

This topic provides a general overview of handlers.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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)

Net.Data programs for the HTTP Server

This topic provides a general overview of Net.Data.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Net.Data is an application that runs on a server and allows you to easily create dynamic Web documents that are called Web macros. Web macros that are created for Net.Data have the simplicity of HTML with the functionality of CGI-BIN applications. Net.Data makes it easy to add live data to static Web pages. Live data includes information that is stored in databases, files, applications, and system services.

Net.Data is a comprehensive web development environment for the creation of simple dynamic Web pages or complex Web-based applications. These applications enable browser clients to access data from a variety of sources, such as databases, applications, and system services. Net.Data consists of a program, the Web macro processor, and one or more dynamic libraries, called language environments. The executable input to Net.Data is the Web macro.

The Web macro processor communicates with the HTTP Server through its CGI-BIN interface. The server uses TCP/IP to connect to the Internet. 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 Web macro processor operable, DB2WWW, in the CGI-BIN directory.

Language environments are the Web macro processor's interface to your data and applications. Each language environment provides a specific interface to a particular resource. For example, Net.Data provides language environments to access DB2 databases, REXX, and other applications through the SYSTEM language environment.


A Web macro is a file that contains a series of statements that are defined by the Net.Data Web macro language. These statements can include standard HTML and language environment-specific statements (for example, SQL statements) as well as macro directives. These statements act as instructions to the Web macro processor, telling it how to construct dynamic Web pages.

When a URL is received by the server that refers to the Web macro processor program, the server starts an instance of the Web macro processor. It then passes essential information, including the name of the requested Web macro and the section of the macro to use. The Web macro processor then:

1. Reads and parses through the Web macro.
2. Interprets all the macro statements.
3. Dynamically builds the HTML page.

When a Web macro language %FUNCTION statement is encountered, the Web macro processor loads the requested language environment-dynamic library (service program). It then passes language-specific information to the language environment to be processed. The language environment processes the information and returns the results to the Web macro processor.

After all parsing is done and language environment processing is completed, all that remains is pure HTML text. This text can then be interpreted by any browser. The Web macro writer has complete control over the level of HTML it uses and what HTML tags are applied. The Web macro processor imposes no restrictions. The pure HTML text is passed back to the server, and the Web macro processor ends. The resulting HTML text is passed to the 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 Net.Data for iSeries home page .


Java servlets and JSPs for the HTTP Server (powered by Apache)


This topic provides a general overview of Java servlets and Java server pages (JSPs).

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Java servlets and JSPs are Java programs that run on the server and extend the capabilities of the Web server by creating a framework for providing requests and or response services over the Web.

The IBM HTTP Server for iSeries provides a basic servlet engine for running entry level Java servlets and JSPs. This servlet engine is a port of the Apache Software Foundation (ASF) Jakarta Tomcat. For general information on Java servlets, JSPs, and Tomcat, see <http://jakarta.apache.org/tomcat/index.html> .

WebSphere Application Server is IBM's strategic Web application server and provides enterprise level support for Java servlets, JSPs, and EJBs (Enterprise Java Beans). For more information on the IBM WebSphere Application Server, see the IBM WebSphere Application Server for iSeries home page .

For more in-depth technical information about Java servlet technology, see <http://java.sun.com/products/servlet/> .

Troubleshoot

This topic lists common problems and solutions for the HTTP Server, the IBM Web Administration for iSeries interface, and other features associated with the product.


Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

List of symptoms:

- “Symptom: Out-of-process ASF Tomcat server does not start or appears to start, but then stops”
- “Symptom: Error 404 on HTTP Server” on page 759
- “Symptom: HTTP Server has a slow response” on page 759
- “Symptom: Error 500 on HTTP Server” on page 759
- “Symptom: Cannot read or write to QUSRSYS/QATMHINSTC or QUSRSYS/QATMHASFT” on page 760
- “Symptom: HTTP Server on port 80 does not start” on page 760
- “Symptom: Web browser problems with HTTP Server” on page 761
- “Symptom: ADMIN server will not start” on page 762
- “Symptom: HTTP Server will not start or functions will not work” on page 762
- “Symptom: Unknown server type when working with HTTP Servers in ADMIN” on page 763
- “Symptom: All servers show status ‘Stopped’” on page 763
- “Symptom: Cannot access ADMIN or some functions do not work” on page 763
- “Symptom: User Profile does not have *IOSYSCFG” on page 763
- “Symptom: Tomcat options not shown in the IBM Web Administration for iSeries interface” on page 763
- “Symptom: Error 500 when Tomcat settings in the IBM Web Administration for iSeries interface is accessed” on page 763
- “Symptom: Cannot create new HTTP Server instance” on page 764
- “Symptom: Net.Data error” on page 764
- “Symptom: Error occurred opening file” on page 764
- “Symptom: Databases fail to deploy when configuring with the IBM Web Administration for iSeries interface” on page 764
- “Symptom: WebSphere Portal authentication performance problems” on page 764

Note: Additional troubleshooting information may be found in the following documentation:

- WebSphere Portal Product Documentation  Web site
- WebSphere Application Server topic
- WebSphere Application Server - Express topic
- Business Applications topic

Symptom: Out-of-process ASF Tomcat server does not start or appears to start, but then stops

Cause

A number of possible causes could be causing your out-of-process ASF Tomcat to not start. Check the job log for the cause.

Solution

Check the job log for a report on the possible problem and enable tracing.

Check the job log:

1. On an iSeries command line, enter **WRKACTJOB** immediately after the server is started.
If the job is active, enter **WRKACTJOB** to work with the job and display the job log.
If the job is not active, enter **WRKSPLF SELECT(QTMHHTTP)** to find the name of the server and display the spool file.
2. Check the ASF Tomcat logs. Inspect the tomcat.log, jvmstdout.txt, and jvmstderr.txt files for possible information concerning the error.

Enable/disable tracing:

1. On an iSeries command line, enter **ADDENVVAR ENVVAR(QIBM_ASFTOMCAT_TRACE) VALUE(10) LEVEL(*SYS)**.
2. Restart the ADMIN server.
3. Start the out-of-process ASF Tomcat server.
4. Additional debug information is created for the job. If this information does not appear in a spool file associated with the started job, then it may be accessed using the Dump User Trace (DMPUSRTRC) command against the ADMIN server and the out-of-process ASF Tomcat server jobs.
To disable tracing, continue with the next steps.
5. On an iSeries command line, enter **RMVENVVAR ENVVAR(QIBM_ASFTOMCAT_TRACE)**.
6. Restart the ADMIN server.
7. Start the out-of-process ASF Tomcat server.

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:

- “Manage server performance for HTTP Server (powered by Apache)” on page 140

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 error log, script error log, and CGI job logs for information on CGI failures.

- If you have not used the IBM Web Administration for iSeries interface to create an HTTP Server configuration, a required directive may be missing from the configuration file. View the configuration file with the IBM Web Administration for iSeries interface for possible errors.

Symptom: Cannot read or write to QUSRSYS/QATMHINSTC or QUSRSYS/QATMHASFT

Cause

The IBM Web Administration for iSeries interface uses the Java Toolbox for iSeries. When reading and writing files in QSYS, the Java Toolbox sometimes uses the DDM server. In the IBM Web Administration for iSeries interface, this results in problems reading or writing the QUSRSYS/QATMHINSTC file containing HTTP Server definitions, or QUSRSYS/QATMHASFT file containing out-of-process ASF Tomcat server definitions.

Solution

On an iSeries command line, enter STRTCPSVR *DDM.

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.

3. If you have configured ASF Tomcat, then there may be error information logged in the ASF Tomcat logs. Inspect the tomcat.log, jvmstdout.txt, and jvmstderr.txt for possible information concerning the error

If these steps do not help, then try starting the server with verbose tracing. See Manage server performance for HTTP Server (powered by Apache) 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.

For additional information see the Apache Software Foundations list of list of known problems with clients  .

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 40 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

Current group PTF for DG1 product is applied (SE01464 - Dup of SE02177).

Error CEE0200

Verify that 57XXJV1 Opt. 3 is installed (must be Option 3).

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 OS/400 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 not disabled.

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: Tomcat options not shown in the IBM Web Administration for iSeries interface

Solution

Do the following:

- Verify latest DG1 PTF's
- Verify that AdminTc.jar is in: /QIBM/UserData/HTTPA/admin/webapps/HTTPAdmin/WEB-INF/lib. This file can be found in /QIBM/ProdData/HTTPA/admin/pgm/. If so, the problem can be circumvented by adding a SYMLNK.
- On the OS/400 command line, type the following:
ADDLNK OBJ('/QIBM/ProdData/HTTPA/admin/pgm/AdminTc.jar')+
NEWLNK('/QIBM/UserData/HTTPA/admin/webapps/httpadmin/web-inf/lib/AdminTc.jar')
Press Enter, then type the following:
CHGOWN OBJ('/QIBM/USerData/HTTPA/admin/webapps/HTTPAdmin/WEB-INF/lib/AdminTc.jar')+
NEWOWN(QTMHHTTP) SYMLNK(*YES)

Symptom: Error 500 when Tomcat settings in the IBM Web Administration for iSeries interface is accessed

Solution

Verify latest DG1 PTF's and SI01832 are applied.

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 IBM Web Administration for iSeries 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 iSeries 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.

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 IBM

Web Administration for iSeries 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 `"(&=%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 IBM Web Administration for iSeries 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.
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 LDAP directory service documentation and the WebSphere Portal Product Documentation  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.

Reference documentation for HTTP Server

This topic provides additional reference documentation for HTTP Server and the IBM Web Administration for iSeries interface.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

See "Related information on HTTP Server" on page 788 for additional reference documentation.

CL commands for HTTP Server

This topic provides information about common command line (CL) commands for a 5250 session.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

The following table summarizes the CL commands associated with the HTTP Server.

Note: As of V5R3, HTTP Server (original) is no longer supported.

Command	Description
CHGHTTTPA	Change HTTP Attributes - change the HTTP Server (original) global data.
CFGHTTTPSCH	Configure HTTP Search - perform various search administration tasks.
CFGTCPHTTP	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.
ENDHTTPCRL	End HTTP Crawl - stops or pauses an active crawling session.
ENDTCP	End TCP/IP - stop TCP/IP.
ENDTCPSPVR	End TCP/IP Server - stop an HTTP Server.
RSMHTTPCRL	Resume HTTP Crawl - start a paused crawling session.
STRCHTSVR	Start Clustered Hash Table Server - start a job for a clustered hash table (CHT) server instance on each of the specified node systems.
STRHTTPCRL	Start HTTP Crawl - start a crawling session.
STRTCP	Start TCP/IP - autostart an HTTP Server.
STRTCPSPVR	Start TCP/IP Server - start an HTTP server.
TRCTCPAPP	Trace TCP/IP Application - capture trace information for the HTTP Server (powered by Apache).

The following table summarizes the CL commands associated with the highly available Web server cluster function. See “Highly available Web server cluster on HTTP Server” on page 17 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 on HTTP Server

This topic provides information about environment variables

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

The supported environment variables for HTTP Server are listed here. The environment variables have been divided into two groups: Non-SSL and SSL.

You can provide secure Web serving when you run HTTP traffic over the SSL protocol. To use SSL, your server must have a digital certificate. This is how a retail company on the Internet allows users to look through the merchandise without security. These same users then fill out order forms and send their credit card numbers by using SSL. A browser that does not support HTTPS cannot request URLs by using HTTP over SSL. The Non-SSL browsers will not allow the submission of forms that need secure submission.

Not all environment variables are supported by both HTTP Server types. Make sure to check that the environment variable is supported by your HTTP Server type.

Note: All headers sent by the client (such as Set-Cookie) are prefixed by "HTTP_", and their value can be extracted. To access variables that are headers, prefix the variable name with "HTTP_". You can also create new variables using the HTTP_set() predefined function. As of V5R3, HTTP Server (original) is no longer supported.

Variable Name	Type	HTTP Server Type	Description
ACCEPT_RANGES	Non-SSL	original	Server API only. Used to accept ranges other than bytes. Example: bytes
ALL_VARIABLES	Non-SSL	original (server API only)	All the CGI environment variables.
AUTH_TYPE	Non-SSL	all	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_CCSD	Non-SSL	all	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_CCSD	Non-SSL	all	Contains the EBCDIC CCSID under which the current server job is running (DefaultFsCCSID configuration directive). It also represents the current job CCSID that is used during server conversion (if any) of CGI input data. Example: 37

Variable Name	Type	HTTP Server Type	Description
CGI_MODE	Non-SSL	all	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	all	Determines which output conversion mode the server is using. Example: %%EBCDIC%%
ClassPath	Non-SSL	Apache	Used with the SetEnv or SetEnvIf directive to provide the JavaClassPath for Java CGI programs. Example: /directory1/directory1a:/directory2/://directory3/
CLIENT_ADDR	Non-SSL	original (server API only)	The IP address for the client. Example: 10.10.2.3
CLIENT_AUTH	Non-SSL	original (server API only)	Defines client authentication as on or off. Example ON.
CLIENTMETHOD	Non-SSL	original (server API only)	The HTTP method that is used in the request.
CLIENT_NAME	Non-SSL	original (server API only)	The host name of the machine making the request. Example: SMITH
CLIENT_PROTOCOL	Non-SSL	original (server API only)	The name and version of the protocol the client is using to make the request. Example: HTTP
CONNECTIONS	Non-SSL	original (server API only)	The number of connections being served, or number of active requests. Example: 15
CONTENT_CHARSET	Non-SSL	original (server API only)	The character set of the response for text/*. Example: US ENGLISH
CONTENT_LENGTH	Non-SSL	all	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	all	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
CONTENT_TYPE_PARAMETERS	Non-SSL	original (server API only)	The other MIME attributes, but not the character set.

Variable Name	Type	HTTP Server Type	Description
DATE_GMT	Non-SSL	Apache (SSI only)	The current date and time in Greenwich Mean Time. Example: 2000/12/31:03:15:20
DATE_LOCAL	Non-SSL	Apache (SSI only)	The current date and time in the local time zone. Example: 2000/08/14:15:40:10
DOCUMENT_NAME	Non-SSL	all	The file name of the document requested by the user. Example: /www/myserver/htdocs/html/hello.html
DOCUMENT_PATH_INFO	Non-SSL	Apache (SSI only)	Contains the additional path information as sent by the Web browser for SSI. Example: /wizard
DOCUMENT_ROOT	Non-SSL	all	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	all	The URI of the document requested by the user. Example: /html/hello.html Note: The DOCUMENT_URI and DOCUMENT_URL environment variables are identical
DOCUMENT_URL	Non-SSL	all	The URL of the document requested by the user. Example: /html/hello.html Note: The DOCUMENT_URI and DOCUMENT_URL environment variables are identical.
DTW_IS_ALLOWED_CLUSTER_ENABLED	Non-SSL	original	This is a single symbol (flag) containing TRUE or FALSE . TRUE means the macro is allowed to be cluster-enabled (its allowed to run across a cluster) and FALSE means the macro is not allowed to be cluster-enabled (its not allowed to run across a cluster). This variable is initialized by Net.Data and cannot be modified. The macro is cluster-enabled only if DTW_IS_ALLOWED_CLUSTER_ENABLED is set to "TRUE" and the macro is persistent. Example: TRUE
ERRORINFO	Non-SSL	original (server API only)	Specifies the error code to determine the error page. Example: 401
EXPIRES	Non-SSL	original (server API only)	Defines the expiration for documents that are stored in a proxy's cache. Example: Thu, 01 Dec 2002 16:00:00 GMT
FSCP	Non-SSL	all	The EBCDIC CCSID used to translate the data. Example: 37
GATEWAY_INTERFACE	Non-SSL	all	Contains the version of CGI that the server is using. Example: CGI/1.1

Variable Name	Type	HTTP Server Type	Description
HTTP_ACCEPT	Non-SSL	all	Contains the list of MIME types the browser accepts. Example: image/gif,image/x-xbitmap,image/jpeg,image/pjpeg,image/png,*/* Note: 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_ACCEPT_CHARSET	Non-SSL	all	Contains the list of character sets the browser accepts. Example: iso-8859-1,*,utf-8 Note: 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_ACCEPT_ENCODING	Non-SSL	all	Contains the list of encoding protocols the browser accepts. Example: gzip Note: 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_ACCEPT_LANGUAGE	Non-SSL	all	Contains the list of languages the browser accepts. Example: de,fr,en Note: 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	all	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 to the environment variable. Example: Keep-Alive
HTTP_COOKIE	Non-SSL	all	User defined cookie for the response. Example: w3ibmTest=true
HTTP_HOST	Non-SSL	all	Contains the HTTP host URL. Example: IBM.COM Note: 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_REASON	Non-SSL	original (server API only)	Sets the reason string in the HTTP response header. Example: "Not Modified"
HTTP_RESPONSE	Non-SSL	original (server API only)	Sets the response code in the HTTPS response header. Example: 304

Variable Name	Type	HTTP Server Type	Description
HTTP_USER_AGENT	Non-SSL	all	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) Note: 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.
IBM_CCSID_VALUE	Non-SSL	all	The CCSID under which the current server job is running. Example: 37
INIT_STRING	Non-SSL	original (server API only)	The string specified on the ServerInit directive.
LAST_MODIFIED	Non-SSL	original (server API only) and Apache	The last modification date of the document requested by the user. Example: 2000/12/31:09:45:20
LOCAL_VARIABLES	Non-SSL	original (server API only)	All the user-defined variables.
NETCP	Non-SSL	all	The default ASCII CCSID used to translate the data. Example: 819
PASSWORD	Non-SSL	original (server API only)	For authentication, contains the decoded password. Example: password Note: (iSeries passwords are not returned) The HTTP Server (original) does not allow access to the PASSWORD variable if the authorization is configured which uses user profiles and passwords for authentication. To prevent an application from obtaining a user profile password, HTTPD_extract() is sensitive to the type of protect setups that are currently configured. If a protection setup is configured with a password file of %%SYSTEM%% (protection requiring user profile password), HTTP_extract() for PASSWORD returns HTTP_PARAMETER_ERROR and sets the value parameter to *CONFLICT. Otherwise, HTTP_extract() returns the appropriate value.
PATH_INFO	Non-SSL	all	Contains the additional path information as sent by the web browser. Example: /wizard
PATH_TRANSLATED	Non-SSL	all	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
PEAKCONNECTIONS	Non-SSL	original (server API only)	Defines the peak number of connections the server allows. Example: 45
PPATH	Non-SSL	original (server API only)	The partially translated path. Example: /wwwhome/wizard

Variable Name	Type	HTTP Server Type	Description
PROXY_ACCESS	Non-SSL	original (server API only)	Defines whether the request is a proxy request or not. Example: NO
PROXY_CONTENT_LENGTH	Non-SSL	original (server API only)	The Content-Length header of the proxy request that is made through HTTPD_proxy. When information is sent with the method of POST, 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 required, you can use the CONTENT_LENGTH value to determine the end of the input string. Example: 7034
PROXY_CONTENT_TYPE	Non-SSL	original (server API only)	The Content-Type header of the proxy request that is made through HTTPD_proxy. 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
PROXY_METHOD	Non-SSL	original (server API only)	The method for the request that is made through HTTPD_proxy. Example: GET
QUERY_STRING	Non-SSL	all	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 Note: The supported maximum size of QUERY_STRING is 4K for HTTP Server (original) and 8K for HTTP Server (powered by Apache).
QZHBHA_MODEL	Non-SSL	original	Model of the highly available Web server. Example: PRIMARYBACKUP
QZHBIS_FIRST_REQUEST	Non-SSL	original	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

Variable Name	Type	HTTP Server Type	Description
QZHBIS_CLUSTER_ENABLED	Non-SSL	original	<p>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.</p> <p>Example: 1</p>
QZHBNEXT_SESSION_HANDLE	Non-SSL	original	<p>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.</p> <p>Example: 8B739003AB741824899F0004AC009021</p>
QZHBRECOVERY .	Non-SSL	original	<p>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</p>
REDIRECT_QUERY_STRING	Non-SSL	Apache	<p>Contains QUERY_STRING from a re-directed request.</p> <p>Example: NAME=Eugene+T%2E+Fox=etfox %7Cibm.net=xyz</p>

Variable Name	Type	HTTP Server Type	Description
REDIRECT_QUERY_URL	Non-SSL	Apache	<p>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.</p> <p>Example: 4D868803AB731824899F0004AC009021</p>
REFERRER	Non-SSL	Apache	<p>Contains the referrer.</p> <p>Example: http://www.myserver.com/cgi-bin/</p>
REFERRER_URL	Non-SSL	all	<p>Contains the referrer URL.</p> <p>Example: http://WWW.MYSERVER.COM:8080/perlSetEnv/</p>
REMOTE_ADDR	Non-SSL	all	<p>Contains the IP address of the remote host (web browser) that is making the request, if available.</p> <p>Example: 10.10.2.3</p>
REMOTE_HOST	Non-SSL	original	<p>Contains the host name of the web browser that is making the request, if available.</p> <p>Example: www.mybusiness.com</p>
REMOTE_PORT	Non-SSL	Apache	<p>Contains the remote user port number.</p> <p>Example: 3630</p>
REMOTE_IDENT	Non-SSL	all	<p>Contains the user ID of the remote user.</p> <p>Example: MyIdentityx</p>
REMOTE_USER	Non-SSL	all	<p>If you have a protected script and the server supports client authentication, this environment variable contains the user name that is passed for authentication.</p> <p>Example: SMITH</p>
REQHDR	Non-SSL	original (server API only)	<p>Contains all of the headers received from the client separated by Carriage Return/Line Feed (\r\n).</p>
REQUEST_CONTENT_LENGTH	Non-SSL	original (server API only)	<p>When information is sent with the method of POST, this variable contains the number of characters. The server typically does not send an end-of-file flag when it forwards the information by using stdin. If required, you can use the CONTENT_LENGTH value to determine the end of the input string.</p> <p>Example: 7034</p>

Variable Name	Type	HTTP Server Type	Description
REQUEST_CONTENT_TYPE	Non-SSL	original (server API only)	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
REQUEST_METHOD	Non-SSL	all	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	Apache	Specifies URI to be requested. Example: /cgi-bin/hello.pgm
RULE_FILE	Non-SSL	all	Specifies rule file to be used. Example: /www/myserver/conf/httpd.conf
SCRIPT_FILENAME	Non-SSL	Apache	The file name of the document requested by the user. Example: /QSYS.LIB/CGI.LIB/HELLO.PGM
SCRIPT_NAME	Non-SSL	all	A virtual path to the program being run. Use this for self-referring URLs. Example: /cgi-bin/hello.pgm
SERVER_ADDR	Non-SSL	all	Contains the address of the server. Example: 10.10.2.3
SERVER_ADMIN	Non-SSL	Apache	Contains information about the server administrator. Example: [no address given]
SERVER_NAME	Non-SSL	all	Contains the server host name or IP address of the server. Example: 10.9.8.7
SERVER_PORT	Non-SSL	all	Contains the port number to which the client request was sent. Example: 2001
SERVER_PROTOCOL	Non-SSL	all	Contains the name and version of the information protocol that is used to make the request. Example: HTTP/1.0
SERVER_ROOT	Non-SSL	original	Sets the directory in which the server lives. It will typically contain the subdirectories like conf/ and logs/. Paths for other configuration fields are taken as relative to this directory. Example: /myserver/main/
SERVER_SIGNATURE	Non-SSL	all	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

Variable Name	Type	HTTP Server Type	Description
SERVER_SOFTWARE	Non-SSL	all	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	all	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	all	The file name of the current file. Example: ssi_parent.shtml
SSI_INCLUDE	Non-SSL	all	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	all	The path and file name of the include, relative to SSI_ROOT. Example: ssi_parent.shtml
SSI_ROOT	Non-SSL	all	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 -> Note: You can use echo to display a value set by the set or global directives.
UNIQUE_ID	Non-SSL	Apache	Provides a unique magic token and acts as the identifier across all requests under very specific conditions. Example: aK8YOakFBZkAABsuEC4AAACB
URI	Non-SSL	original (server API only)	The URL of the document requested by the user. Example: /cgi-bin/hello.pgm
URL	Non-SSL	original (server API only)	The URL of the document requested by the user. Example: /cgi-bin/hello.pgm
USERID	Non-SSL	original (server API only)	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
USERNAME	Non-SSL	original (server API only)	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
HTTPS	SSL	all	Returns ON if the system has completed an SSL handshake. It returns OFF if the exchange of signals to set up communications between two modems has failed. Example: OFF
HTTPS_CIPHER	SSL	all	This is the cipher that is used to negotiate with the client on the SSL handshake. Example: SSL_RSA_WITH_RC4_128_MD5

Variable Name	Type	HTTP Server Type	Description
HTTPS_CLIENT_CERT	SSL	all	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...
HTTPS_CLIENT_CERT_COMMON_NAME	SSL	all	The common name from the client certificate's distinguished name. Example: SMITH
HTTPS_CLIENT_CERT_COUNTRY	SSL	all	The country code from the client certificate's distinguished name. Example: US
HTTPS_CLIENT_CERT_DN	SSL	all	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	Apache	The email of the client owning the certificate. Example: me@mycompany.com
HTTPS_CLIENT_CERT_ISSUER_COMMON_NAME	SSL	all	The common name of the certificate authority that issued the client's certificate. Example: SMITH
HTTPS_CLIENT_CERT_ISSUER_COUNTRY	SSL	all	The country code of the certificate authority that issued the client's certificate. Example: US
HTTPS_CLIENT_CERT_ISSUER_DN	SSL	all	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	all	The e-mail address of the certificate authority that issued the client's certificate. Example: me@mydomain.net
HTTPS_CLIENT_CERT_ISSUER_LOCALITY	SSL	all	The locality or city of the certificate authority that issued the client's certificate. Example: New York
HTTPS_CLIENT_CERT_ISSUER_ORG_UNIT	SSL	all	The organizational unit of the certificate authority that issued the client's certificate. Example: bird watchers
HTTPS_CLIENT_CERT_ISSUER_ORGANIZATION	SSL	all	The organization name of the certificate authority that issued the client's certificate. Example: dove

Variable Name	Type	HTTP Server Type	Description
HTTPS_CLIENT_CERT_ISSUER_POSTAL_CODE	SSL	Apache	The postal code of the certificate authority that issued the client's certificate. Example: 12344-6789
HTTPS_CLIENT_CERT_ISSUER_STATE_OR_PROVINCE	SSL	all	The state or province of the certificate authority that issued the client's certificate. Example: North Dakota
HTTPS_CLIENT_CERT_LEN	SSL	all	The length of the certificate passed in HTTPS_CLIENT_CERT. Example: 968
HTTPS_CLIENT_CERT_LOCALITY	SSL	all	The locality or city of the client certificate's distinguished name. Example: New York
HTTPS_CLIENT_CERT_ORG_UNIT	SSL	all	The organization unit name from the client certificate's distinguished name. Example: Pack234
HTTPS_CLIENT_CERT_ORGANIZATION	SSL	all	The organization name from the client certificate's distinguished name. Example: Scouts
HTTPS_CLIENT_CERT_POSTAL_CODE	SSL	Apache	The postal code assigned by the issuing certificate authority. Example: 80525
HTTPS_CLIENT_CERT_SERIAL_NUM	SSL	all	The serial number assigned by the issuing certificate authority. Example: 3F:E4:83:81:02:D5:58
HTTPS_CLIENT_CERT_STATE_OR_PROVINCE	SSL	all	The state or province from the client certificate's distinguished name. Example: Alberta
HTTPS_CLIENT_ISSUER_EMAIL	SSL	Apache	Contains the email address of the Certificate Authority that issued the certificate. Example: jones@mydomain.net
HTTPS_KEYSIZE	SSL	all	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]

Variable Name	Type	HTTP Server Type	Description
HTTPS_PORT SSL	original	original	For HTTP Server (original), if a valid security product is installed and the SSLMode directive is SSLMode=ON, this environment variable contains the SSL port number the server is listening on. Example: 443 Note: HTTP Server (powered by Apache) must set this environment variable in the configuration file. Add to your config file "SetEnv HTTPS_PORT nnnnn" where nnnnn is the https port matching the secure port for the context that it applies to (server config or specific to virtual host).
HTTPS_SESSION_ID	SSL	all	Set to NULL by default when used with HTTP Server (powered by Apache).
HTTPS_SESSION_ID_NEW	SSL	all	If the value is TRUE , it indicates that a full handshake was performed for this SSL session. If the value is FALSE , it indicates that an abbreviated handshake was performed for this SSL session. Example: True
SSL_CIPHER	SSL	Apache	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	Apache	The country code from the client certificate's distinguished name. Example: USA
SSL_CLIENT_CERTBODY	SSL	Apache	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: MIIC0DCC big IB gIHOL2Yx...
SSL_CLIENT_CERTBODYLEN	SSL	Apache	The length of the certificate passed in SSL_CLIENT_CERT. Example: 828
SSL_CLIENT_CERT_EMAIL	SSL	Apache	The email of the client owning the certificate. Example: me@mycompany.com
SSL_CLIENT_CN	SSL	Apache	The common name from the client certificate's distinguished name. Example: SMITH
SSL_CLIENT_DN	SSL	Apache	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

Variable Name	Type	HTTP Server Type	Description
SSL_CLIENT_ICN	SSL	Apache	The common name of the certificate authority that issued the client's certificate. Example: SMITH
SSL_CLIENT_IC	SSL	Apache	The country code of the certificate authority that issued the client's certificate. Example: CA
SSL_CLIENT_IDN	SSL	Apache	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	Apache	The e-mail of the certificate authority that issued the client's certificate. Example: me@mycompany.com
SSL_CLIENT_IL	SSL	Apache	The locality of the certificate authority that issued the client's certificate. Example: New York
SSL_CLIENT_IO	SSL	Apache	The organization name of the certificate authority that issued the client's certificate. Example: bird watchers
SSL_CLIENT_I OU	SSL	Apache	The organizational unit of the certificate authority that issued the client's certificate. Example: bird watchers
SSL_CLIENT_IPC	SSL	Apache	The postal code of the certificate authority that issued the client's certificate. Example: 55901
SSL_CLIENT_IST	SSL	Apache	The state or province of the certificate authority that issued the client's certificate. Example: MNA
SSL_CLIENT_L	SSL	Apache	The locality or city of the client certificate's distinguished name. Example: New York
SSL_CLIENT_NEWSESSIONID	SSL	Apache	If the value is TRUE , it indicates that a full handshake was performed for this SSL session. If the value is FALSE , it indicates that an abbreviated handshake was performed for this SSL session. Example: True
SSL_CLIENT_O	SSL	Apache	The organization name from the client certificate's distinguished name. Example: bird watchers

Variable Name	Type	HTTP Server Type	Description
SSL_CLIENT_OU	SSL	Apache	The organizational unit name from the client certificate's distinguished name. Example: bird watchers
SSL_CLIENT_PC	SSL	Apache	The postal code from the client certificate's distinguished name. Example: 58401
SSL_CLIENT_SERIALNUM	SSL	Apache	The serial number assigned by the issuing certificate authority. Example: 3F:E4:83:81:02:D5:58
SSL_CLIENT_SESSIONID	SSL	Apache	If the value is TRUE , it indicates that a full handshake was performed for this SSL session. If the value is FALSE , it indicates that an abbreviated handshake was performed for this SSL session. Example: True
SSL_CLIENT_ST	SSL	Apache	The state or province from the client certificate's distinguished name. Example: North Dakota
SSL_PROTOCOL_VERSION	SSL	Apache	The SSL protocol version negotiated on the SSL handshake with the client. Example: SSLV3
SSL_SERVER_C	SSL	Apache	The country where the server is located in. Example: Denmark
SSL_SERVER_CN	SSL	Apache	The common name from the server certificate's distinguished name. Example: WWW.MYDOMAIN.COM
SSL_SERVER_DN	SSL	Apache	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	Apache	The e-mail address of the server certificate. Example: me@mydomain.net
SSL_SERVER_L	SSL	Apache	The locality of the server certificate's distinguished name. Example: New York
SSL_SERVER_OU	SSL	Apache	The organization unit name from the server certificate's distinguished name. Example: bird watchers
SSL_SERVER_O	SSL	Apache	The organization name from the server certificate's distinguished name. Example: bird watchers

Variable Name	Type	HTTP Server Type	Description
SSL_SERVER_ST	SSL	Apache	The state or province from the server certificate's distinguished name. Example: North Dakota
HTTP_AS_AUTH_PROFILETKN	SSL and Non-SSL	Apache	A 32-bit value used to identify or authenticate the user. See the ProfileToken directive for more information.
QIBM_CGI_LIBRARY_LIST	Non-SSL	Apache	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.

Lotus Domino plug-in for HTTP Server (powered by Apache)

This topic provides information about the Lotus Domino plug-in for HTTP Server (powered by Apache).

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

With Domino 6 for iSeries, customers can take advantage of the HTTP Server (powered by Apache) to forward HTTP traffic to their Domino 6 servers. A new plug-in service program QZSRVIHSAH is provided with the HTTP Server product (5722DG1) as a PTF in OS/400 releases V5R2 and V5R1 to provide this function.

See HTTP Server (powered by Apache) plug-in for Domino 6 on iSeries for more information 

Server-side include commands for HTTP Server

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

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

Server-side includes allow you to insert information into CGI programs and HTML documents that the server sends to the client.

Server-side include commands for HTTP Server (powered by Apache)

HTTP Server (powered by Apache) SSI commands have the following format:

```
<--#command parameter="value -->
```

Note: There is a space before -->. The parameter value is normally enclosed in double quotes, but for compatibility with the HTTP Server (original), double quotes are not required.

The following describes the SSI commands for HTTP Server (powered by Apache).

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 on HTTP Server” on page 766 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 752 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" -->
```

For compatibility with HTTP Server (original), text can be supplied without 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.

Note: For compatibility with HTTP Server (original), a variable name can also begin with an ampersand (&).

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 ê is a client side variable, so the & is escaped to create the value ":frêd" 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

Supported OS/400 file systems for Web content served by HTTP Server

This topic provides information about supported OS/400 file system for Web content.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

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

- root (/)
- QSYS.LIB
- QOpenSys
- QDLS
- NFS
- QFileSvr.400
- QNetWare
- 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.

See File systems in the integrated file system for more information about the integrated file system.

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 IBM HTTP Server for iSeries:

- 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.

Time formats for HTTP Server

This topic provides information about time formats for server-side includes.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.

See IBM Service  for more information.

The following table contains formatting values used to specify time with server-side includes. See “Log formats for HTTP Server (powered by Apache)” on page 29 and “Server-side include commands for HTTP Server” on page 782 for proper use of the server-side include time format.

Value	Description	Example
%%	Replace with %.	%
%a	Replace with the abbreviated weekday name.	Mon
%A	Replace with the full weekday name.	Monday
%b	Replace with the abbreviated month name.	Apr

Value	Description	Example
%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[cCzyY]	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.	
%OU	Replace with the week number of the year (Sunday as the first day of the week, rules corresponding to %U) using alternative numeric symbols.	
%Ow	Replace with the weekday (Sunday=0) using the alternative numeric symbols.	
%OW	Replace with the week number of the year (Monday as the first day of the week) using the alternative numeric symbols.	

Value	Description	Example
%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
%Z	Valid only for HTTP Server (original). Replace with the name of the time zone or no characters if the time zone is not known.	

Related information on HTTP Server


Find additional information about the HTTP Server and Web serving.

Important: Information for this topic supports the latest PTF levels for HTTP Server for iSeries . It is recommended that you install the latest PTFs to upgrade to the latest level of the HTTP Server for iSeries. Some of the topics documented here are not available prior to this update.




See IBM Service  for more information.





Listed below are the iSeries manuals and IBM Redbooks™ (in PDF format), Web sites, and Information Center (categories or) topics that relate to the IBM HTTP Server. You can view or print any of the PDFs.

Manuals

- iSeries Performance Capabilities Reference in the V5R1 iSeries Information Center 

Redbooks








- HTTP Server Performance and Capacity Planning 
- AS/400 Internet Security: Developing a Digital Certificate Infrastructure 
- Implementation and Practical User of LDAP on the IBM eServer iSeries Server 

- Who Knew You Could Do That with RPG IV? A Sorcerer's Guide to System Access and More 
- Building AS/400 Internet-Based Applications with Java 
- Porting UNIX Applications Using AS/400 PASE 
- HTTP Server (powered by Apache): An Integrated Solution for IBM eServer iSeries Servers 

Web sites


- IBM HTTP Server for iSeries product page 
- IBM Net.Data for iSeries product page 
- Apache HTTP Server Project 
- IBM AlphaWorks 
- IBM developerWorks® 
- IBM PartnerWorld® e-business 
- IBM Easy400 CGI WEB development tools for iSeries 
- Ignite/400 e-business user group 

Programming resources

- Who Knew You Could Do That with RPG IV? A Sorcerer's Guide to System Access and More 
- Building AS/400 Internet-Based Applications with Java 
- Porting UNIX Applications Using AS/400 PASE 
- IBM AlphaWorks 
- IBM developerWorks tools, code, and tutorials 
- IBM PartnerWorld e-business 
- WebSphere Development Tools for iSeries 

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

1. Open the PDF in your browser (click the link above).
2. In the list of your browser, click **File**.
3. Click **Save As...**
4. Navigate to the directory in which you would like to save the PDF.
5. Click **Save**.

If you need Adobe Acrobat Reader to view or print these PDFs, you can download a copy from the Adobe Web site (<http://www.adobe.com/prodindex/acrobat/readstep.html>)  .

Legal notices for Apache Software Foundation on HTTP Server

The Apache Software Foundation has specific licensing agreements for the ASF Apache Web Server and the ASF Jakarta Tomcat servlet engine.

- Apache license
- Apache Software Foundation Jakarta Tomcat servlet engine 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:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

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 49XA
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 information 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 is for planning purposes only. The information herein is subject to change before the products described become available.

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.

Trademarks

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

AIX
AS/400
DB2
DB2 Universal Database
Domino

e(logo)server
eServer
i5/OS
IBM
iSeries
OS/400
Redbooks
Sametime
Tivoli
WebSphere
zSeries

Lotus, Freelance, and WordPro are trademarks of International Business Machines Corporation and Lotus Development Corporation 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.

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

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

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

Terms and conditions for downloading and printing information

Permissions for the use of the information you have selected for download are granted subject to the following terms and conditions and your indication of acceptance thereof.

Personal Use: You may reproduce this information for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of this information, or any portion thereof, without the express consent of IBM.

Commercial Use: You may reproduce, distribute and display this information solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of this information, or reproduce, distribute or display this information 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 information or any 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 information 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 THIS INFORMATION. THE INFORMATION IS 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.

All material copyrighted by IBM Corporation.

By downloading or printing information from this site, you have indicated your agreement with these terms and conditions.



Printed in USA