WebSphere Application Server

Caching Proxy Administration Guide

Version 6.0.2

GC31-6857-02

WebSphere Application Server

Caching Proxy Administration Guide

Version 6.0.2

GC31-6857-02

Note
Before using this information and the product it supports, be sure to read the general information under "Notices" on page 265.

Third edition (June 2005)

This edition applies to:
WebSphere Application Server, Version 6.0.2

and to all subsequent releases and modifications until otherwise indicated in new editions.

Order publications through your IBM representative or through the IBM branch office serving your locality.

Contents

Figures

About this book

This preface describes the audience and purpose of this book, its organization, accessibility features, conventions and terminology, and related documents.

Who should read this book

The *Caching Proxy Administration Guide* is written for experienced network and system administrators who are familiar with their operating systems and with providing Internet services. Prior exposure to the Caching Proxy is not required.

This book is not intended to support previous releases of Caching Proxy.

Conventions and terminology used in this book

This documentation uses the following typographical and keying conventions.

Table 1. Conventions used in this book

Convention	Meaning
Bold	When referring to graphical user interfaces (GUIs), bold face indicates menus, menu items, labels, buttons, icons, and folders. It also can be used to emphasize command names that otherwise might be confused with the surrounding text.
`Monospace`	Indicates text you must enter at a command prompt. Monospace also indicates screen text, code examples, and file excerpts.
Italics	Indicates variable values that you must provide (for example, you supply the name of a file for *fileName*). Italics also indicates emphasis and the titles of books.
Ctrl-*x*	Where *x* is the name of a key, indicates a control-character sequence. For example, Ctrl-c means hold down the Ctrl key while you press the c key.
Return	Refers to the key labeled with the word Return, the word Enter, or the left arrow.
`%`	Represents the Linux™ and UNIX® command-shell prompt for a command that does not require root privileges.
`#`	Represents the Linux and UNIX command-shell prompt for a command that requires root privileges.
`C:\`	Represents the Windows® command prompt.
Entering commands	When instructed to "enter" or "issue" a command, type the command and then press Return. For example, the instruction "Enter the **ls** command" means type **ls** at a command prompt and then press Return.
[ ]	Enclose optional items in syntax descriptions.
{ }	Enclose lists from which you must choose an item in syntax descriptions.
|	Separates items in a list of choices enclosed in { }(braces) in syntax descriptions.
...	Ellipses in syntax descriptions indicate that you can repeat the preceding item one or more times. Ellipses in examples indicate that information was omitted from the example for the sake of brevity.

Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. These are the major accessibility features in WebSphere® Application Server, Version 6.0.2:

- You can use screen-reader software and a digital speech synthesizer to hear what is displayed on the screen. You can also use voice recognition software, such as IBM ViaVoice, to enter data and to navigate the user interface.
- You can operate features by using the keyboard instead of the mouse.
- You can configure and administer Application Server features by using standard text editors or command-line interfaces instead of the graphical interfaces provided. For more information about the accessibility of particular features, refer to the documentation about those features.

How to send your comments

Your feedback is important in helping to provide the most accurate and high-quality information. If you have any comments about this book or any other documentation about the Edge components of WebSphere Application Server:

- Send your comments by e-mail to fsdoc@us.ibm.com. Be sure to include the name of the book, the part number of the book, the version of WebSphere Application Server, and, if applicable, the specific location of the text you are commenting on (for example, a page number or table number).

Related information

- *Concepts, Planning, and Installation for Edge Components*, GC31-6855-02
- *Programming Guide for Edge Components*, GC31-6856-02
- *Load Balancer Administration Guide*, GC31-6858-02
- *IBM WebSphere Edge Services Architecture*
- IBM home Web site: *www.ibm.com/*
- IBM® WebSphere Application Server product Web site: *www.ibm.com/software/webservers/appserv/*
- IBM WebSphere Application Server library Web site: *www.ibm.com/software/webservers/appserv/library.html*
- IBM WebSphere Application Server support Web site: *www.ibm.com/software/webservers/appserv/support.html*
- IBM WebSphere Application Server Information Center: *www.ibm.com/software/webservers/appserv/infocenter.html*
- IBM WebSphere Application Server Edge Components Information Center: *www.ibm.com/software/webservers/appserv/ecinfocenter.html*

Part 1. Getting started with the Caching Proxy

This part provides an overview of the Caching Proxy component, instructions for using the Configuration and Administration forms and Configuration Wizard, instructions for manually editing the ibmproxy.conf file, and procedures for starting and stopping the proxy server.

This part contains the following chapters:

Chapter 1. Overview

The Caching Proxy intercepts data requests from a client, retrieves the requested information from content-hosting machines, and delivers that content back to the client. Most commonly, the requests are for documents stored on Web server machines (also called origin servers or content hosts) and delivered via the Hypertext Transfer Protocol (HTTP). However, you can configure the Caching Proxy to handle other protocols, such as File Transfer Protocol (FTP), and Gopher.

The Caching Proxy stores cacheable content in a local cache before delivering it to the requester. Examples of cacheable content include static Web pages and JavaServer Pages (JSP) FILES with dynamically generated but infrequently changing fragments. Caching enables the Caching Proxy to satisfy subsequent requests for the same content by delivering it directly from the local cache, which is much quicker than retrieving it again from the content host.

IMPORTANT: Caching Proxy is available on all Edge component installations, with the following exceptions:

- Caching Proxy is not available for Edge component installations that run on Itanium 2 or AMD Opteron 64-bit processors.
- Caching Proxy is not available for Edge component installations of Load Balancer for IPv6.

New features

The *Caching Proxy Administration Guide* includes new features, new supported platforms, and corrective updates for Version 6.0 as well as for post-V5.0 releases (5.0.1, 5.0.2, 5.1, and 5.1.1).

The most significant new features for V6.0.2 are:

- **FIPS enable directive**

  This new directive enables FIPS approved ciphers for SSLV3 and TLS protocol in SSL connections. For more information, see "FIPSEnable — Enable Federal Information Processing Standard (FIPS) approved ciphers for SSLV3 and TLS" on page 201.
- **Directives to enable Caching Proxy to cache multiple variants of a resource (URI)**

  Two new directives enable Caching Proxy to cache and retrieve multiple variants of a URI based on the HTTP Vary header and the Cookie header.

  For more information, see "SupportVaryHeader — Cache more than one variant of a resource based on the HTTP Vary header" on page 257 and "RegisterCacheIdTransformer — Cache more than one variant of a resource based on the Cookie header" on page 246.

The most significant new features for V6.0, V6.0.1, and post-V5.0 are:

- **New platform support**

  For complete information on supported platforms for V6 Edge Components, access the following WebSphere Application Server Web page, `http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html`

  – **Support on Linux for S/390, zSeries, iSeries and pSeries**

In addition to support for running Caching Proxy on Linux for Intel, the proxy now runs on Linux for S/390, zSeries, iSeries and pSeries.

  - **Support on AIX 5.2 and AIX 5.3**

    In addition to AIX 5.1, Caching Proxy now supports AIX 5.2 and AIX 5.3.

  - **Support on Solaris 9**

    In addition to Solaris 8, Caching Proxy now supports Solaris 9.

  - **Support on Windows Server 2003**

    In addition to Windows 2000, Caching Proxy now supports Windows Server 2003.

  - **Support on HP-UX Version 11i**

    In addition to support for AIX, Linux, Solaris, and Windows systems, Caching Proxy supports HP-UX.

- **Support for JDK 1.4.2**

  A new version of the 32-bit JDK is now supported: JDK 1.4.2.

- **Support for GSKit 7**

  A new version of the GSKit is provided and installed by default with Caching Proxy: GSKit 7.

- **Enhancements and new directives for junction rewriting**

  There are enhancement and new directives related to junction rewriting. For more information, see:

  - "Enable junction rewrite (optional)" on page 41
  - "JunctionReplaceUrlPrefix — Replace URL instead of insert prefix when used with JunctionRewrite plugin" on page 210
  - "JunctionSkipUrlPrefix — Skip rewriting URLs that already contain the prefix, when used with JunctionRewrite plugin" on page 211
  - "JunctionRewriteSetCookiePath — Rewrite the path option in the Set-Cookie header, when used with JunctionRewrite plugin" on page 211
  - "RewriteSetCookieDomain — Specify the domain pattern that needs to be rewritten" on page 247
  - "Proxy — Specify proxy protocols or reverse proxy" on page 238

  Also provided is an alternative to using the JunctionRewrite plugin. For more information, see:

  - "UseCookie as an alternative to JunctionRewrite" on page 44
  - "JunctionRewrite — Enables URL rewriting" on page 210

- **Additional new directives**

  The following new directives are supported:

  - "CacheMatchLanguage — Specify the language preference for the returned cache content" on page 179
  - "EnableTcpNodelay — Enable TCP NODELAY socket option" on page 195
  - Limit request related directives:
    - "LimitRequestBody — Specify the maximum body size in PUT or POST requests" on page 212
    - "LimitRequestFields — Specify the maximum number of headers in client requests" on page 213
    - "LimitRequestFieldSize — Specify the maximum header length and request line" on page 213
  - Save or load cache memory to disk directives:
    - "ExportCacheImageTo — Export cache memory to disk" on page 200

- "ImportCacheImageFrom — Import cache memory from a file" on page 209

- **Enhancements to existing directives**

  There are enhancements to the following existing directives:

  – BindSpecific: The `OutgoingSrcIp` option allows the proxy to use a specific source IP address to make outgoing connections.

  – ReversePass: The `host:port` option allows the proxy to apply different ReversePass rules based on the backend server host name and port.

- **Sample transmogrifier plugin provided to extend JunctionRewrite functionality**

  For JunctionRewrite, sample code that can be customized is now provided which rewrites/parses JavaScript (SCRIPT) and applet (APPLET) tag blocks in HTML files. (On its own, the JunctionRewrite plugin cannot process the resource links in JavaScript or in parameter values of Java. For more information, see "Sample transmogrifier plugin to extend JunctionRewrite functionality" on page 45.

- **PACD daemon configuration related changes**

  To enable anonymous binding, see "Creating pac_ldap.cred" on page 133.

  For an SSL connection between a proxy and an LDAP server, put the key database password in the pac_keyring.pwd file. See "Creating a new key database, password, and stash file" on page 115.

- **Changes default configuration to be more secure**

  In the configuration file (ibmproxy.conf), changes to default settings were made in order to provide greater security. For example, changes were made to disable HTTP CONNECTION and disable SSL tunneling. There are no new directives for this enhancement.

Chapter 2. Using the Configuration and Administration forms

The Caching Proxy comes with HTML forms that can be served to requesting clients and used to configure the proxy server. These forms run CGI programs that edit the local proxy server configuration file, ibmproxy.conf. To use these forms, the proxy server must be running and must be configured to pass the forms from the local directory where they reside.

By default, Caching Proxy is installed with Pass directives included in the ibmproxy.conf file that enable access to the Configuration and Administration forms. When a client requests the default home page from this proxy server, Frntpage.html is served. This page contains a hypertext link to the Configuration and Administration forms start page, wte.html.

The Configuration and Administration forms are protected and require client authentication before they are served. For instructions on setting the administrator's ID and password, refer to "Setting the administrator password" on page 9.

Browser requirements

A Web browser used to access the Configuration and Administration forms must support the following:

- *HTML 4.0*: All forms are written to the HTML 4.0 specification. Your Web browser must support HTML 4.0 and framesets.
- *Java 1.1 and JavaScript*: Applets are written to the Java 1.1 specification. Your Web browser must support a Java Virtual Machine that is Java 1.1 compliant. The applets are incompatible with Java Virtual Machines compliant with the Java 2.0 specification. Both JavaScript and Java must be enabled.
- *256 colors*: The workstation on which the Web browser runs must support at least 256 colors.

Recommended browsers are Mozilla 1.4 or Mozilla 1.7 (for Linux and UNIX systems) or Internet Explorer (for Windows systems). Refer to *Concepts, Planning, and Installation for Edge Components* for additional browser related information for viewing Configuration and Administration forms.

Notes:

1. On 64-bit PowerPC Linux systems, it will not be possible to access the Configuration and Administration forms with the Mozilla browser since there is no SDK available for this architecture. Alternatively, you can access the Configuration and Administration forms from a different machine with a supported Web browser.
2. If you are prompted twice to login when starting the Administrative console, your Java setting in Internet Explorer may not be set correctly. To correct this in Internet Explorer, select **Tools>Internet Options>Advanced** and deselect the **Use Java 2 v1.4.X** checkbox.

Accessing the Configuration and Administration forms

To access the Configuration and Administration forms:

1. Ensure that the proxy server is running. For instructions on starting the proxy server, refer to Chapter 5, “Starting and stopping the Caching Proxy,” on page 15.
2. Direct an HTTP browser to request your proxy server’s home page (Frntpage.html) or the Configuration and Administration start page (wte.html).

   Note: This page depends on your proxy server’s actual mapping rules and can vary from the default pages shown in parentheses.

   `http://your.server.name[:port][/directory][/page.html]`

   where

   - *your.server.name* is the full path name of your host (for example, `http://www.ibm.com/`).
   - [:*port*] If your proxy server listens for administrative requests on a port other than 80, include the port number after the server name: `http://your.server.name:port`
   - [*/directory*] The addition of a directory within the URL depends on the mapping rules.
   - [*/page.html*] The HTML page needs to be specified only if it is not listed as a welcome page. For information on welcome pages, refer to “Define default welcome pages” on page 48.
3. Click **Configuration and Administration forms** to get to the server configuration forms. You are prompted for the administrator user name and password. Type an authorized user name and password. The Caching Proxy configuration client window opens.

   Notes:

   a. The contents of the navigation frame on the left can take several seconds to load after the main page is displayed.

   b. On Windows 2003 systems, connections that request administration forms (CGI scripts) might receive a reset before the connection is complete. As a result, browsers might report no data received or display page not available messages. To avoid this problem, increase MaxActiveThreads to a value greater than 200, or increase ConnThreads to a value greater than 50 to resolve the reset connections. See “MaxActiveThreads — Specify the maximum number of active threads” on page 218 and “ConnThreads — Specify the number of connection threads to be used for connection management” on page 187 for more information on these directives.
4. The navigation frame on the left shows the five major categories of configuration forms:

   - **Proxy Configuration**
   - **Cache Configuration**
   - **Server Configuration**
   - **Server Activity Monitor**
   - **Plug-in Configuration**

   Click the triangular pointer at the left of a heading to expand the list of configuration forms in that category. Click a form to open it. The form shows the current configuration values (if any) in the input fields; if you have not changed the configuration since installation, these are default values.

5. On any form, enter configuration information for that particular function. Each form provides instructions to assist you in deciding what changes to make. For further information, click the help icon, the question mark (**?**) at the top of each form. It provides the following links:
   - **Field help**—Descriptions of the fields in each screen panel
   - **How do I...**—Detailed steps in using the form to do specific tasks
   - **Index**—An index of the help information
6. After filling in a form, click **Submit** to update the server configuration with the changes you made. The **Submit** button is located below the input fields on each form. If you do not want to make the changes you indicated on the form, click **Reset**, and the fields on the form return to their original values.
7. If you click **Submit** and your input is accepted, the following message is displayed in the upper frame:

   ```
   The requested configuration changes have been completed successfully
   ```

   If the input is not accepted, an error message is displayed in the upper frame, indicating which settings are not acceptable.
8. To restart the proxy server, click the server restart icon (|) in the upper frame. When the proxy server receives the restart command, it stops accepting requests from clients but completes any requests that are already in process. After reloading the changed configuration file, the proxy starts accepting client requests again.

   Note: Changing certain directives either by using the Configuration and Administration forms or by editing the ibmproxy.conf file requires that instead of a restart, you stop the server completely and then start it again before the changes can take effect. Those directives are listed in Table 6 on page 161.

Setting the administrator password

After installing the Caching Proxy packages, you must create an administrator identification and password for accessing the Configuration and Administration forms. The default proxy server configuration authenticates users that request the Configuration and Administration forms by using the webadmin.passwd passwords file in either the /opt/ibm/edge/cp/server_root/protect/ directory on Linux and UNIX systems or the \Program Files\IBM\edge\cp\etc\ directory on Windows systems. Package installation does not overwrite an existing webadmin.passwd file.

Use the following commands to add an administrator entry to the webadmin.passwd file:

- On Linux and UNIX systems:

  ```
  # htadm -adduser /opt/ibm/edge/cp/server_root/protect/webadmin.passwd
  ```

  When prompted, provide the **htadm** program with a user name, password, and real name for the administrator.
- On Windows systems:

  ```
  cd "\Program Files\IBM\edge\cp\server_root\protect\"
  htadm -adduser webadmin.passwd"
  ```

  When prompted, provide the **htadm** program with a user name, password, and real name for the administrator.

Note: The administrator user name and password are case-sensitive even if the operating system is not case-sensitive. Be sure to input the exact user name and password entered using the `htadm` command when accessing the Configuration and Administration forms.

For a detailed description of the **htadm** command, refer to "htadm command" on page 153.

Chapter 3. Using the Configuration Wizard

The Caching Proxy Configuration Wizard enables you to quickly configure an installed Caching Proxy. This program sets only the essential directives that are required to modify the behavior of the Caching Proxy to function as a surrogate. The proxy server can require additional configuration.

To use the Caching Proxy Configuration Wizard:

1. Start the Configuration Wizard.

   On Windows systems: click **Start –> Programs –> IBM WebSphere –> Edge Components –> Caching Proxy –> Configuration Wizard**.

   On Linux and UNIX systems: enter the command
   `/opt/ibm/edge/cp/cpwizard/cpwizard.sh`
2. Select the network port on which the proxy server will listen for HTTP requests.
3. Type the name of the target content server.
4. Enter the user ID and password for the proxy server administrator.

Notes:

1. The Configuration Wizard sets the following directives:

   ```
   Port port
   Proxy /* http://content server :port
   ```
2. If the Configuration Wizard is used to configure the proxy server, then to enable SSL, a mapping rule must be created to proxy requests received through port 443. For more information, refer to “Define mapping rules” on page 39.

   Examples:

   ```
   Proxy /* http://content server :443
   ```

   or

   ```
   Proxy /* https://content server :443
   ```

Limitations: On Linux systems, keyboard shortcuts do not work for the Caching Proxy Configuration Wizard.

Chapter 4. Manually editing the ibmproxy.conf file

Caching Proxy can be configured manually, editing the ibmproxy configuration file, or via the Configuration and Administration forms.

- On Linux and UNIX systems, the ibmproxy.conf file is located in the /etc/ directory.
- On Windows systems, the ibmproxy.conf file is located in C:\Program Files\IBM\edge\cp\etc\en_US\.

The configuration file is made up of statements called directives. To change your configuration, edit the configuration file by modifying the directives, and save your changes. You can use almost any text editor, such as emacs and vi, to edit the configuration file.

Note: Do not use the text file editor that is included in the Solaris Common Desktop Environment (CDE). The Solaris editor sometimes modifies the file's owning group and changes properties of the file link so that the Configuration and Administration forms cannot write to the configuration file.

Your changes to the configuration file take effect when you restart the server, unless you changed one of the directives identified in Table 6 on page 161. If you changed one of the directives in that list, you must stop the server and start it again. For instructions, see Chapter 5, "Starting and stopping the Caching Proxy," on page 15.

Appendix B, "Configuration file directives," on page 161 describes each of the configuration file directives and gives details about syntax.

Chapter 5. Starting and stopping the Caching Proxy

The Caching Proxy is designed to run continuously as a background process with minimal operator intervention. Typically, the proxy server starts during the boot cycle of the machine and is stopped only when maintenance is required. The proxy server may be manually started when necessary. The proxy server can also be passed a restart instruction, which effectively stops then starts the proxy server without disrupting active client connections.

Automatic startup and shutdown on Linux and UNIX systems

On Linux and UNIX systems, an **ibmproxy** initialization script and associated symbolic links are placed in the appropriate /etc/ directories when the Caching Proxy is installed. These scripts are then integrated into the startup and shutdown routines of the operating system. You can change the configuration settings for automatic restart by editing the **ibmproxy** script and changing the **ibmproxy** command options.

Note: Solaris file descriptor limit

It is possible that the Caching Proxy initialization script can fail to set the desired maximum number of file descriptors due to the Solaris systemwide limit on file descriptors. If the systemwide maximum is less than the setting in the Caching Proxy initialization script, then the systemwide limit is used. You can change the file descriptor limit to avoid proxy performance problems that can result from too low a value (less than 1024). Issue the **ulimit** command to view the number of descriptors that are currently available. If the value is less than 1024, increase the file descriptor limit. To increase the file descriptor limit to 1024, add the following line to the /etc/system file:

```
set rlim_fd_cur=0x400
```

Disabling automatic startup and shutdown

To disable automatic startup and shutdown:

- On AIX systems, remove the **ibmproxy** command from the initialization file.
- On HP-UX systems, remove the following links to **ibmproxy**:
  - /sbin/rc1.d/K154ibmproxy
  - /sbin/rc2.d/S880ibmproxy
- On Linux systems, remove the symbolic links to **/etc/rc.d/init.d/ibmproxy** in the run level subdirectories.

  On SUSE Linux, remove the following links to **ibmproxy**:
  - /etc/rc.d/rc3.d/S20ibmproxy
  - /etc/rc.d/rc3.d/K20ibmproxy
  - /etc/rc.d/rc4.d/S20ibmproxy
  - /etc/rc.d/rc4.d/K20ibmproxy
  - /etc/rc.d/rc5.d/S20ibmproxy
  - /etc/rc.d/rc5.d/K20ibmproxy

  On Red Hat Linux, remove the following links to **ibmproxy**:

- /etc/rc.d/rc0.d/K54ibmproxy
- /etc/rc.d/rc1.d/K54ibmproxy
- /etc/rc.d/rc2.d/K54ibmproxy
- /etc/rc.d/rc6.d/K54ibmproxy
- /etc/rc.d/rc3.d/S88ibmproxy
- /etc/rc.d/rc5.d/S88ibmproxy

- On Solaris systems, remove the **ibmproxy start** command and its two kill scripts as follows:
  - Delete S88ibmproxy from the /etc/rc2.d directory.
  - Delete K54ibmproxy from the /etc/rc0.d directory.
  - Delete K54ibmproxy from the /etc/rc1.d directory.

Manual startup on Linux and UNIX systems

Regardless of the startup method, the **ibmproxy** command is eventually invoked, either directly from the command prompt or from within a script. For a detailed description of the **ibmproxy** command, refer to "ibmproxy command" on page 158. Examples of only the most commonly used arguments follow.

On AIX:

- To start the proxy server for the default locale, using the **startsrc** command, enter the following:

  ```
  startsrc -s ibmproxy
  ```

- To start the proxy server for any locale other than the default, using the **startsrc** command, enter the following:

  ```
  startsrc -s ibmproxy -e "LC_ALL=locale"
  ```

- To start the proxy server with the default run-time settings, without using the **startsrc** command, enter the following:

  ```
  ibmproxy
  ```

On HP-UX:

- To start the proxy server by running the initialization script, enter the following at a root prompt:

  ```
  /sbin/init.d/ibmproxy start
  ```

- To start the proxy server as a background process without running the initialization script, enter the following at a root prompt:

  ```
  /usr/sbin/ibmproxy
  ```

- To start the proxy server as a foreground process without running the initialization script, enter the following at a root prompt:

  ```
  /usr/sbin/ibmproxy -nobg
  ```

On Linux:

- To start the proxy server by running the initialization script, enter the following at a root prompt:

  ```
  /etc/rc.d/init.d/ibmproxy start
  ```

- To start the proxy server as a background process without running the initialization script, enter the following at a root prompt:

  ```
  /usr/sbin/ibmproxy
  ```

- To start the proxy server as a foreground process without running the initialization script, enter the following at a root prompt:

  ```
  /usr/sbin/ibmproxy -nobg
  ```

- To start the proxy server using a preexisting SQUID configuration file, squidConfig.file, enter the following at a root prompt:

  ```
  squidConfig.file -r /etc/errors_icons.conf
  ```

  where the *errors_icons.conf* file identifies which icons to use for designated file types when browsing directories.

On Solaris:

- To start the proxy server by running the initialization script, enter the following at a root prompt:

  ```
  /etc/init.d/ibmproxy start
  ```

- To start the proxy server as a background process without running the initialization script, enter the following at a root prompt:

  ```
  /usr/sbin/ibmproxy
  ```

- To start the proxy server as a foreground process without running the initialization script, enter the following at a root prompt:

  ```
  /usr/sbin/ibmproxy -nobg
  ```

Startup as a Windows service

If the Caching Proxy is installed as a Windows service, it is started like any other Windows service:

1. Click **Start –> Settings (for Windows 2000) –> Control Panel**.
2. In the **Control Panel** window, double-click **Administrative Tools –> Services**.
3. In the **Services** window, highlight **Caching Proxy**.
4. Click **Start** to initiate the Caching Proxy service.

If the Caching Proxy is installed as a service, it can be configured to start up automatically when Windows starts. In that case, you do not have to log on before the proxy can serve requests. To have your proxy start automatically:

1. Click **Start –> Settings (for Windows 2000) –> Control Panel**.
2. In the **Control Panel** window, double-click **Administrative Tools –> Services**.
3. In the **Services** window, highlight **Caching Proxy**.
4. Click the **Automatic** radio button, then click **Start** to initiate the Caching Proxy service automatically when Windows starts.

Refreshing the PATH environment variable

If the Caching Proxy is marked as `Started` in the **Services** window, but the proxy is not working, the machine might not have been restarted after the proxy was installed. If the Caching Proxy service is set to interact with the desktop, failure to restart can also cause the following error message to appear in a pop-up box:
`Message catalog error: the message catalog could not be loaded or is invalid`

The machine must be restarted so that the value of the PATH environment variable is refreshed in the Windows registry. If the registry is not refreshed, it is possible for the PATH variable to show the correct Caching Proxy and GSK7 paths but to function incorrectly.

Note: A potential conflict exists for Windows systems when both the Caching Proxy and another application, such as a network file system, run as services. The Caching Proxy sometimes cannot interpret a path containing a remote drive owned by a file system application that also is running as a service.

The problem can occur if the path for the file system service appears before the path for the Caching Proxy service in the Windows PATH environment variable. Altering the PATH variable to put file system services near the end of the setting can solve this problem.

This problem does not affect remote drives controlled by applications that do not run as Windows services. For example, the Caching proxy can access shared drives on other Windows machines that are visible through a local area network (LAN).

Startup as a Windows application

Using the Start menu

When the Caching Proxy is installed as a Windows application, the installation procedure creates a **Caching Proxy** entry as a submenu of the **Start** menu. To start the Caching Proxy as an application, click **Start –> Programs –> IBM WebSphere –> Edge Components –> Caching Proxy**.

This startup procedure runs the proxy server with the current configuration settings. If you want to specify other settings at startup time, use the command startup procedure (see the next section).

Using the command prompt

To start the server from any Windows or DOS command prompt, use the **ibmproxy** command. If you have not shut down and restarted Windows since you installed the server, enter the full path name for this command, as follows, (by default):

```
c:\Program Files\IBM\edge\cp\bin\ibmproxy.exe
```

The **ibmproxy** command starts the server with the current configuration settings. If you have not changed the server configuration since installation, the current configuration is based on the information you entered during installation and on the default options.

The **ibmproxy** command starts the server as an application, even if you have installed the Caching Proxy to run as a service. To force the server to run as an application, you can also specify the command option **-noservice**. Other command options change the configuration settings at run time.

Starting multiple proxy servers

Multiple instances of the proxy server can run concurrently, but each instance must listen on a separate port. On AIX systems, only one instance can be started with SRC. Unique configuration files must be specified for all instances of the server because the configuration file identifies a port number, and this number must be different for each server on a particular machine. To start an additional instance of the server (when at least one is already running), enter the following command:

- On Linux and UNIX:

```
ibmproxy -r other_config_file
```

- On Windows:

```
ibmproxy -noservice -r other_config_file
```

where *other_config_file* is a unique configuration file.

When starting multiple instances of the server, record the process ID that is displayed for each instance. These IDs are required to stop specific instances of the server.

Note: On Linux systems running multiple instances of the server, the command **/etc/rc.d/init.d/ibmproxy stop** stops only the last server that was started. Other instances must be stopped separately. Refer to "Manual shutdown on Linux and UNIX systems" for related information.

Manual shutdown on Linux and UNIX systems

To stop the server:

- You must be either the user who started the process or the superuser root.
- You must use the same method by which the server was started. The following table lists start methods and their associated stop methods.

Table 2. Start and stop methods for Linux and UNIX systems

Start method	Stop method
From /etc/inittab (On AIX)	Enter `stopsrc -s ibmproxy`
From /sbin/init.d (On HP-UX)	Enter `/sbin/init.d/ibmproxy stop`
From /etc/rc.d/init.d (On Linux)	Enter `/etc/rc.d/init.d/ibmproxy stop`
`ibmproxy`	1. Find the **ibmproxy** process ID: On AIX, enter `ps -aef | grep "ibmproxy"`. On Linux, enter `ps -aux | grep ibmproxy | grep server_ID`. On Solaris and HP-UX, enter `ps -ef | grep "ibmproxy"` 2. Stop the **ibmproxy** process: Enter `kill process_id` To stop all servers on this machine: Enter `killall ibmproxy`
`ibmproxy -nobg`	Enter `ctrl-c`
`ibmproxy -r -other_config_file`(On AIX)	Enter `stopsrc -s ibmproxy -p process_id`
`ibmproxy -r -other_config_file`(On Linux)	1. Find the **ibmproxy** process ID: Enter `ps aux | grep ibmproxy | grep process_id` 2. Stop the **ibmproxy** process: Enter `kill process_id`

To stop the server at a root prompt, enter:

- On AIX: `stopsrc -s ibmproxy`
- On HP-UX: `/sbin/init.d/ibmproxy stop`
- On Linux: `/etc/rc.d/init.d/ibmproxy stop`
- On Solaris: `/etc/init.d/ibmproxy stop`

Limitations of the shutdown commands

You can experience the following limitations when using the shutdown commands:

- **AIX, HP-UX, and Linux**

  On AIX, HP-UX, and Linux systems, the commands to stop the Caching Proxy system sometimes shut down only the Caching Proxy process. The AIX command that results in this behavior is the **stopsrc -s ibmproxy** command. The HP-UX and Linux command that results in this behavior is the **ibmproxy -stop** command.

  The PACD process, which is used by the LDAP server, might be left running after shutting down the proxy server. The PACD process can be safely shut down by using the **kill** command as follows:

  ```
  kill -15 PACD_process_ID
  ```

- **Solaris**

  Issuing the **ibmproxy -stop** command on a Solaris system does not have the same effect as the command does on other operating systems. Because of a limitation in Solaris code, the Server Termination plug-in step is not executed when **ibmproxy -stop** is used on Solaris platforms.

  This limitation has implications for the proxy server software as well as for customer-implemented plug-ins.

  It is possible for the PACD process, which is used by the LDAP server, to continue running after the proxy server is shut down. The PACD process can be safely shut down by using the **kill** command as follows:

  ```
  kill -15 PACD_process_ID
  ```

Manual shutdown on a Windows system

You can stop the Caching Proxy server in the same ways that you stop other Windows programs.

If the proxy is installed as a service:

1. Click **Start –> Settings (for Windows 2000) –> Control Panel**.
2. In the **Control Panel** window, double-click **Administrative Tools –> Services**.
3. In the **Services** window, highlight **Caching Proxy**.
4. Click **Stop** to stop the Caching Proxy service.

If the proxy is not installed as a service, do any of the following to stop the Caching Proxy:

- Click the **x** icon in the top right corner.
- From the **File** menu, click **Exit**.
- Press **Alt + F4**.

Restarting after configuration changes

After changing the server configuration (by using the Configuration and Administration forms or by editing the ibmproxy.conf file), you must restart the server before the changes take effect. In most cases, you can restart the server without stopping it first. But some settings are not refreshed by a simple restart. For more information, see Table 6 on page 161.

To restart the server without stopping it first, click the **Restart** button on any Configuration and Administration form, or type the following: `ibmproxy -restart`

Part 2. Configuring and tuning the Caching Proxy process

This part explains how the Caching Proxy component interacts with the operating system, computer hardware, and network. It also provides procedures for configuring this interaction. These elements of proxy server configuration are typically managed by the system administrator and must be carefully coordinated with network resources, such as IP addresses and host names, as well as with system resources, such as available memory and CPU cycles.

This part contains the following chapters:

Chapter 6. Define the server

The Caching Proxy typically runs as a background process on a host computer system that is configured to perform as a network server. This process is associated with (*bound to*) one or all active Internet Protocol (IP) addresses on the host computer system. It listens for various Internet protocols, such as FTP and HTTP, on specified ports and performs actions on these requests in accordance with its behavioral configuration. (For more information, refer to Part 3, "Configuring Caching Proxy behavior," on page 35.)

By default, the Caching Proxy assumes the name of its host computer system. You can override this default behavior by deliberately specifying a host name for the proxy server. In order to bind the Caching Proxy to a specific IP address, the host name of the proxy server must be changed to equal that IP address.

Note: In the event that the proxy server attempts to bind to an IP address, and that host name is not set to an available IP address, the bind will fail and the proxy server will listen on all available IP addresses.

The host name of the proxy server does not affect how client traffic is resolved. The proxy server does not compare its own host name with the value of the host name argument in the header of the HTTP request. The host name of the proxy server is occasionally incorporated into dynamically generated local content pages, such as error messages. It is also passed backed to the requested client as the value of the Via argument in the HTTP header.

The proxy server can be configured to replace the host name of the requesting client with the host name of the proxy server prior to passing the request on to the destination server. Doing so forces the destination server to maintain the communication channel through the proxy server, rather than establishing a direct connection with the client.

Define the proxy server process by specifying the physical location of the proxy server files on the host computer system, the name with which proxy server refers to itself, and the ports on which it listens as values for the ServerRoot, Hostname, and Port directives. If the host has multiple IP addresses, the proxy server can be bound to a specific address by setting the value of the BindSpecific directive to On and setting the value of the Hostname directive equal to the IP address.

An administration port provides a method of accessing the Configuration and Administration forms and maintaining the server. To provide access to the proxy server through an administration port, specify a value for the AdminPort directive. Requests received on the administration port are not queued with requests received on the standard port. Mapping rules can be written to allow access to the Configuration and Administration forms through this port.

When the BindSpecific directive is enabled, Caching Proxy is bound to the port specified by the Port directive along with the IP address derived from the value of the Hostname directive. The port specified by the AdminPort directive is bound to all IP addresses available on the system.

To override the default name of the server that is running, such as IBM-PROXY or IBM_HTTP_SERVER, specify a value for HeaderServerName directive. This value populates the HTTP response server field.

To improve proxy performance, the value of the PureProxy directive can be set to on. This completely disables all caching functionality.

Associated directives

The following directives define the proxy server process:

- "Hostname — Specify the fully qualified domain name or IP address for the server" on page 205
- "ServerRoot — Specify the directory where the server program is installed" on page 251
- "HeaderServerName — Specify the name of the proxy server returned in the HTTP header" on page 205
- "BindSpecific — Specify whether the server binds to one or all IP addresses" on page 174
- "Port — Specify the port on which the server listens for requests" on page 229
- "AdminPort — Specify the port for requesting administrative pages or forms" on page 171
- "PureProxy — Disable a dedicated proxy" on page 242

For more information, refer to Chapter 4, "Manually editing the ibmproxy.conf file," on page 13.

Configuration and Administration forms

The following Configuration and Administration forms edit the values of the associated directives:

- **Server Configuration –> Basic Settings –> Host name**
- **Server Configuration –> Basic Settings –> Server root**
- **Server Configuration –> Basic Settings –> Default port number(s)**
- **Server Configuration –> Basic Settings –> Administrator port number**
- **Server Configuration –> Basic Settings –> Bind options**
- **Proxy Configuration –> Proxy Performance –> Run as pure proxy**

Note: You cannot use the Configuration and Administration forms to edit the HeaderServerName directive.

For more information, refer to Chapter 2, "Using the Configuration and Administration forms," on page 7.

Chapter 7. Establish process ownership

When a user other than the superuser root starts the Caching Proxy, that user maintains ownership of all of the processes associated with the proxy server. However, if the Caching Proxy is started by the superuser root, a set user ID function within the proxy server reads the UserId and GroupId directives in the ibmproxy.conf file and changes process ownership to the specified user and group. This is done to limit file access and protect the computer system. If you change the UserId or GroupId directives, you must update the ownership and permissions for log directories and other files, such as an access control list (ACL), that are used by the proxy server.

Establish the ownership of the proxy server process by specifying the user identification, group identification, and location of the file in which the process ID is recorded as values for the UserID, GroupID, and PidFile directives.

To force the proxy server process to run as foreground process, set the value of the NoBG directive to on.

On Linux systems:

On Linux systems, only processes and threads responsible for listening for connections will have their ownership changed. Processes and threads responsible for other activities within the workflow will still be owned by root. All processes and threads receive process ID (PID) numbers. The **ps** command lists all process IDs, regardless of whether they are associated with a process or thread.

Note: On some Linux kernels, Caching Proxy may generate the following error message in its error log:

```
Cannot init groups for user nobody, errno: 1
```

You can disregard the error message because there is no affect on the normal operation of Caching Proxy. There is also a workaround to avoid the error message by exporting the following environment variables before starting Caching Proxy:

```
export RPM_FORCE_NPTL=1
export LD_ASSUME_KERNEL=2.4.19:
```

Associated directives

The following directives establish proxy server process ownership:

- "UserId — Specify the default user ID" on page 260
- "GroupId — Specify the group ID" on page 204
- "NoBG — Run the Caching Proxy process in foreground" on page 222
- "PidFile (Linux and UNIX only) — Specify the file in which to store the process ID of Caching Proxy" on page 228

For more information, refer to Chapter 4, "Manually editing the ibmproxy.conf file," on page 13.

Configuration and Administration forms

The following Configuration and Administration forms edit the values of the associated directives:

- **Server Configuration –> Basic Settings –> UserID**
- **Server Configuration –> Basic Settings –> GroupID**
- **Server Configuration –> Basic Settings –> Process ID file location**

Note: You cannot use the Configuration and Administration forms to edit the NoBG directive.

For more information, refer to Chapter 2, "Using the Configuration and Administration forms," on page 7.

Chapter 8. Manage connections

The Caching Proxy spawns a new thread to handle each client request. If no threads are available, the proxy server holds requests until more threads become available. As the number of active threads increases, the proxy server consumes more memory. Specify the maximum number of active threads as the value for the MaxActiveThreads directive.

The listen backlog is the number of pending requests for client connections that the server logs before it refuses connections with new clients. Base this setting on the number of requests that the server can process in a few seconds. A server must respond to a client connection before it times out. Specify the maximum number of connections that can be held in the backlog as the value for the ListenBacklog directive.

The proxy server can maintain persistent client/server connections. With a persistent connection, the server accepts multiple requests from the client and sends responses over the same TCP/IP connection. Using persistent connections reduces latency for clients and lowers the CPU load on the proxy server, at the low cost of a small increase in server memory. Overall throughput is increased when the server does not establish a separate TCP/IP connection for each request and response, and the TCP/IP connection can be used with greatest efficiency when the connection is persistent.

Server-side connection pooling applies the benefits of persistent connections on the server side by allowing the reuse of existing connections between a proxy server and the origin servers. Each reused connection saves three TCP packets (two three-way handshake packets to set up the connection, and one to close it). The benefits of server-side connection pooling include:

- Less network congestion (by minimizing the opening and closing of connections)
- Less CPU time used in routers, clients, and servers
- Less memory used on clients and servers
- On cache misses, quicker proxy response (by avoiding opening and closing connections)

Note: Connection pooling is recommended only in a controlled environment. It can degrade performance where the origin servers are not HTTP 1.1 compliant. Note also that it is critical that the origin servers are set up properly. The following is a simple example from the Apache 1.3.19 configuration file:

- `#KeepAlive: Whether or not to allow persistent connections (more than one request per #connection). Set to Off to deactivate#`
- `KeepAlive On`
- `#MaxKeepAliveRequests: The maximum number of requests to allow during a persistent connection. Set to 0 to allow an unlimited amount. Leave this number high for maximum performance#`
- `Max KeepAliveRequests 0`
- `#KeepAliveTimeout: Number of seconds to wait for the next request from the same client on the same connection#`
- `KeepAliveTimeout 240`

These settings keep connections to the Web servers open as long as they are being used and allow the proxy, rather than the origin server, to manage the connections. Therefore, the connections are pooled only to the extent needed.

When server-side connection pooling is enabled, HTTP connections to the origin servers are pooled. SSL connections are also pooled in configurations where the SSLEnable directive for the proxy is set to on.

Configure how connection pooling is maintained by specifying the maximum number of idle sockets to hold per server at any one time, how long the server waits before terminating an idle persistent connection, and the time interval at which the garbage collection thread checks for timed-out connections (the default is two minutes).

Define the amount of time that various connections remain open as values for the InputTimeout, OutputTimeout, PersistTimeout, ReadTimeout, and ScriptTimeout directives.

Associated directives

The following directives manage connections with the proxy server process:

- "MaxActiveThreads — Specify the maximum number of active threads" on page 218
- "ConnThreads — Specify the number of connection threads to be used for connection management" on page 187
- "ListenBacklog — Specify the number of listen backlog client connections that the server can carry" on page 213
- "ProxyPersistence — Allow persistent connections" on page 241
- "MaxPersistRequest — Specify the maximum number of requests to receive on a persistent connection" on page 219
- "ServerConnPool — Specify the pooling of connections to origin servers" on page 250
- "MaxSocketPerServer — Specify the maximum open sockets for server" on page 220
- "ServerConnTimeout — Specify maximum inactive period" on page 250
- "ServerConnGCRun — Specify the interval at which to run garbage collection thread" on page 249
- "PersistTimeout — Specify the time to wait for the client to send another request" on page 228
- "InputTimeout — Specify the input timeout" on page 210
- "ReadTimeout — Specify the time limit for a connection" on page 244
- "OutputTimeout — Specify the output timeout" on page 225
- "ScriptTimeout – Specify the timeout setting for scripts" on page 248

For more information, refer to Chapter 4, "Manually editing the ibmproxy.conf file," on page 13.

Configuration and Administration forms

The following Configuration and Administration forms edit the values of the associated directives:

- **Server Configuration –> System Management –> Performance –> Maximum number of active threads**
- **Server Configuration –> System Management –> Performance –> Size of listen backlog**
- **Proxy Configuration –> Proxy Performance –> Allow persistent connections**
- **Server Configuration –> System Management –> Performance –> Maximum Requests**
- **Server Configuration –> System Management –> Performance –> Persist timeout**
- **Server Configuration –> System Management –> Timeouts –> Input timeout**
- **Server Configuration –> System Management –> Timeouts –> Read timeout**
- **Server Configuration –> System Management –> Timeouts –> Output timeout**
- **Server Configuration –> System Management –> Timeouts –> Script timeout**
- **Server Configuration –> System Management –> Timeouts –> Persist timeout**

Notes:

1. You cannot use the Configuration and Administration forms to edit the ServerConnPool, MaxsocketPerServer, ServerConnTimeout, or ServerConnGCRun directives.
2. The PersistTimeout can be edited from either the **Server Configuration –> System Management –> Performance** form or the **Server Configuration –> System Management –> Timeouts** form.

For more information, refer to Chapter 2, “Using the Configuration and Administration forms,” on page 7.

Chapter 9. Tune the proxy server process

You can noticeably improve the performance of the Caching Proxy by properly setting up and tuning the system. The following are suggestions for improving setup and tuning.

Set performance-related directives

The following directives significantly affect the performance of the proxy server process:

- "PureProxy — Disable a dedicated proxy" on page 242. This feature improves system performance by completely disabling caching.
- "ProxyPersistence — Allow persistent connections" on page 241. This feature enables clients and servers to maintain open connections. Persistent connections decrease the lag time for requests for documents from the proxy server, but require increased network bandwidth as well as a dedicated server thread for each connection. Do not allow persistent connections if your setup limits the number of available threads.

The following Configuration and Administration form fields edit the values of the associated directives:

- **Proxy Configuration –> Proxy Performance: Run as a pure proxy**
- **Proxy Configuration –> Proxy Performance: Allow persistent connections**

Examine other applications

Examine the services or daemons that are running on your system and remove those that are not required to increase available memory and CPU cycles. For example, if your system is running a Web server that serves only a few Web pages, consider using the Caching Proxy as your only Web server. Disable other Web servers as follows:

- On AIX: Examine /etc/inittab
- On Linux: Examine /etc/rc.d/rc*x*.d for your system's default run level (typically 2)
- On HP-UX and Solaris: Examine /etc/rc*x*.d for your system's default run level (typically 2).
- On Windows systems:
  1. Click **Start –> Settings (for Windows 2000) –> Control Panel –> Administrative Tools –> Services**.
  2. Review services that are not required but are set to `Automatic`.
  3. Change the startup type for those services from `Automatic` to `Manual`.

Verify paging space

Ensure that your system has sufficient paging space for proper operation. The system needs twice as much paging space as physical memory. If possible, spread the paging space across multiple physical drives. For example, a Netfinity 5000 server with 512 MB of memory and five SCSI drives needs 1 GB of total paging space with approximately 200 MB on each drive.

Tune the file system

The Caching Proxy creates and destroys many files during its operation. If your proxy server records accesses (using the access log, proxy access log, or cache access log), direct the logs to their own file system so that if the logs grow unexpectedly, they do not use space intended for another function (such as the cache).

Tune TCP/IP configuration

The Caching Proxy is sensitive to changes to TCP/IP configurations. Lowering the TCP/IP values on any operating system might cause the proxy server to perform in an unexpected manner. More specifically, if the TCP/IP values are set too low, connections might be reset by clients that connect to the proxy server or by origin servers to which the proxy connect. This is especially true for clients that connect to the proxy server through a low bandwidth connection (56700 bps or lower). If TCP/IP parameters must be lowered, proceed with caution.

Tune TCP time wait interval for high-load environments (HP-UX, Linux, Solaris, Windows)

The TCP time wait interval specifies the length of time that a socket waits for a FIN packet from the sender before forcibly closing. In high-load environments, the proxy server may appear to stall if a large number of sockets remain suspended in the TIME_WAIT state after their connections are closed. Reducing the TCP time wait interval will reduce the number of suspended sockets and, in high-load environments, may prevent the proxy server from appearing to stall. It is recommended that this interval be set to 5 seconds.

To set the TCP time wait interval to 5 seconds:

- On HP-UX:

  Issue the following command:

  ```
  ndd /dev/tcp -set tcp_time_wait_interval 5000
  ```

  Use the ″sam″ utility to set the kernel parameter max_thread_proc to at least 2048.

  Note: Also consider adjusting the following kernel parameters: maxfiles, maxfiles_lim, maxproc, shmem, tcp_conn_request_max, tcp_ip_abort_interval, tcp_keepalive_interval, tcp_rexmit_interval_initial, tcp_rexmit_interval_max, tcp_rexmit_interval_min, tcp_xmit_hiwater_def, tcp_recv_hiwater_def.

- On Linux:

  Issue the following commands:

  ```
  echo "1024 61000" > /proc/sys/net/ipv4/ip_local_port_range
  echo "5" > /proc/sys/net/ipv4/tcp_fin_timeout
  ```

- On Solaris:

  Issue the following command:

  ```
  ndd /dev/tcp –set tcp_time_wait_interval 5000
  ```

  Edit the /etc/system file to read as follows::

  ```
  set tcp:tcp_conn_hash_size=8129
  ```

- On Windows:

A registry entry must be created set a TCP time wait interval. Refer to your Windows documentation for more information.

Adjust the Linux kernel

Several limits in the Linux kernel are low and can be modified. Some can be changed through the /proc file system, and others require recompiling the kernel.

Note: The /proc file system is virtual; that is, it does not exist physically on the disk. Instead, it serves as an interface into the Linux kernel. Because it does not exist, your input values are lost on restart. Therefore, place changes that you want to make to the /proc file system in the /etc/rc.d/rc.local file on RedHat or in the /etc/rc.config file on SUSE. Changes are then always activated at restart.

Some recommendations follow:

- The file descriptor maximum is 4096 by default. It can be changed by adding the following to the rc.local file:

  ```
  echo 32768 > /proc/sys/fs/file-max
  ```

- The inode maximum is 16384 by default. It can be changed by adding the following to the rc.local file:

  ```
  echo 65536 > /proc/sys/fs/inode-max
  ```

- The TCP and UDP port range is by default 1024 – 4999. This can be changed to 32768 – 61000 by adding the following to the rc.local file:

  ```
  echo 32768 61000 > /proc/sys/net/ipv4/ip_local_port_range
  ```

- By default, the number of allowed tasks is 512. If too many tasks are running, this affects the maximum number of threads for a process. This limit can be increased to 2048 by modifying the value for NR_TASKS in the *YourKernelSource*/include/linux/tasks.h file.
- In addition, change the value of MIN_TASKS_LEFT_FOR_ROOT to 24. You must recompile your kernel for this change to take effect.

If you decide to rebuild your kernel, enable only those options that you definitely need. If you do not need a specific daemon, do not run it.

Adjust the AIX thread tuning variables

On AIX systems, Caching Proxy performance can be improved by using system scope threads and allowing multiple heaps to be used by threads. Performance is related to the operating system's multiprocessing ability and the thread scheduling of the underlying operating system. Performance improvement can be achieved by setting the following AIX thread tuning variables as follows:

```
export AIXTHREAD_SCOPE=S
export SPINLOOPTIME=500
export YIELDLOOPTIME=100
export MALLOCMULTIHEAP=1
```

You can set these environment variables either before starting `/usr/sbin/ibmproxy`, or you can add them to `/etc/rc.ibmproxy` if using **startsrc -s ibmproxy** to start the proxy server. After adjusting these thread tuning variables, performance improvement will be more evident on SMP systems. However, in some cases, improvement may also be evident on uniprocessor systems.

Note: For more information, look to your AIX operating system documentation for details on thread tuning variables.

Part 3. Configuring Caching Proxy behavior

This part explains how the Caching Proxy component responds to client requests and provides procedures for configuring this behavior. These elements of proxy server configuration are typically managed by a Web administrator and do not affect other processes on the host computer system or other computer systems within the network.

This part contains the following chapters:

Chapter 10. Manage request processing

When the Caching Proxy receives a client request, it performs the action specified in the method field on the object specified in the URL field, if the requested method has been enabled. The proxy server resolves the URL according to a set of mapping rules defined by the administrator. These mapping rules might instruct the Caching Proxy to act as a Web server and retrieve the object from the local file system or to act as a proxy server and retrieve the object from an origin server.

This chapter describes how to enable methods, define mapping rules, and configure a surrogate proxy server.

Enable HTTP/FTP methods

Client requests to the server include a method field that indicates the action that the server is to perform on the specified object.

Following is a list of methods that the proxy server supports and a description of how it responds to a client request containing the method if the method is enabled.

Note: Some methods are the same for HTTP and for FTP requests. Enabling these methods for HTTP also enables them for FTP.

DELETE
The proxy server deletes the object identified by the URL. DELETE allows clients to erase files from your Caching Proxy. Use server protection setups to define who can use DELETE and on which files. For details, see Chapter 25, “Server protection setups,” on page 107.

GET The proxy server returns whatever data is identified by the URL. If the URL refers to an executable program, the proxy returns the output of the program. This method can be handled over persistent connections.

HEAD
The proxy server returns only the HTTP document header identified by the URL without the document body.

OPTIONS
The proxy server returns information about the communications options on the request-response chain identified by the URL. This method allows a client to determine the options and requirements associated with an object, or the capabilities of a server, without having to act on or retrieve the object.

POST The request contains data and a URL. The proxy server accepts the data enclosed in the request as a new subordinate of the resource identified in the URL, which processes the data. The resource can be a data-accepting program, a gateway to some other protocol, or a separate program that accepts annotations.

The POST method is designed to handle the annotation of existing resources. Examples include posting a message to a bulletin board, newsgroup, mailing list, or similar group of resources; providing a block of data, for example, from a form to a data-handling program; or extending a

database through an **append** operation. In the Caching Proxy itself, the POST method is used to process the Configuration and Administration forms.

This method can be handled over persistent connections.

PUT The request contains data and a URL. The proxy server stores the data in the resource identified in the URL. If the resource already exists, PUT replaces it with the data contained in the request. If the resource does not exist, PUT creates it and populates it with the data contained in the request. This method can be handled over persistent connections.

Enabling the PUT method allows files to be written to the Caching Proxy by using HTTP and FTP. Because PUT allows clients to write to the Caching Proxy, you need to use server-protection setups to define who can use PUT and the files on which PUT can be used. (See Chapter 25, “Server protection setups,” on page 107.)

TRACE
The proxy server echoes the request message sent by the client. This method allows the client to see what is being received at the other end of the request chain and use that data for testing or diagnostics. The content type of the proxy response is message/http.

Associated directives

The following directives enable HTTP/FTP methods:

- “Enable — Enable HTTP methods” on page 195
- “Disable — Disable HTTP methods” on page 193

For more information, refer to Chapter 4, “Manually editing the ibmproxy.conf file,” on page 13.

Configuration and Administration forms

The following Configuration and Administration forms edit the values of the associated directives:

- **Server Configuration –> Request Processing –> HTTP Methods –> GET**
- **Server Configuration –> Request Processing –> HTTP Methods –> HEAD**
- **Server Configuration –> Request Processing –> HTTP Methods –> POST**
- **Server Configuration –> Request Processing –> HTTP Methods –> PUT**
- **Server Configuration –> Request Processing –> HTTP Methods –> DELETE**
- **Server Configuration –> Request Processing –> HTTP Methods –> OPTIONS**
- **Server Configuration –> Request Processing –> HTTP Methods –> TRACE**

Note: If you disable the POST method, you cannot use the Configuration and Administration forms to configure the Caching Proxy.

For more information, refer to Chapter 2, "Using the Configuration and Administration forms," on page 7.

Define mapping rules

Mapping rules are configuration directives that cause client requests to the Caching Proxy to be processed in some way, for example, passed to an origin server (proxied), redirected, or rejected. Setting mapping rules correctly is important to the proper functioning of your Caching Proxy. Mapping rules affect the following:

- Basic proxy function
- Access to the browser-based Configuration and Administration forms
- Ability to cache servlet results and other dynamically generated content

Mapping rule directives use the following form:

rule template target [*IP_address* | *host_name*]:[*port*]

Only requests that match the given template and IP-port combination are subject to that rule. A template can contain wildcards, for example, https://**/*.asp.

The order in which the rules appear in the configuration file is significant. Except for Map directives, as soon as the request is matched to a template, it is processed and subsequent rules are not evaluated. The Map directive replaces the URL in the request. This new request continues to be compared to the remaining mapping rules.

Mapping rules

The following mapping rules apply to client requests that match the given template:

- **Map** — rewrite the request. The Map rule replaces a request URL (template) with another URL string (target). After this substitution, the request, containing the new string, continues to be compared to the remaining mapping rules.
- **Pass, Exec** — serve the request locally. The Pass and Exec rules process the request at the proxy server. The Pass rule maps a request URL (template) to a file that is served from the proxy server (target); the Exec rule maps a request URL to a CGI program that runs on the proxy server.

- **Fail** — reject the request. The Fail rule rejects a request (template) at the proxy server. Any request that matches the template of a Fail rule is not processed further. Fail rules do not have target arguments.
- **Redirect** — forward the request. The Redirect rule forwards a request (template) to another Web server (target). Because a full URL, including the communication protocol, is the target of this rule, it is possible to change the protocol during this redirection, for example, to add SSL encryption to an HTTP request. A redirection does not check the cache prior to satisfying the request.
- **Proxy, ProxyWAS** — proxy the request. The Proxy and ProxyWAS rules pass requests (templates) to another server (target). Unlike a simple Redirect rule, the Proxy rules allow the proxy server to check the cache to satisfy a request, to cache content from origin servers, and to write HTTP headers that enable advanced functions. Use the ProxyWAS rule instead of the Proxy rule when the origin server is a WebSphere Application Server.

The following mapping rule applies to the origin server response:

- **ReversePass** — intercept automatically redirected requests. A ReversePass rule matches the response from the origin server to the template as it passes through the proxy server on its way to the client. The ReversePass directive is designed to detect a redirection status code that would cause a client to directly contact the origin server. The client is instructed to contact the server defined in the target argument.

The following mapping rules apply to API applications:

- **nameTrans** — accepts the request and runs an API application, defined by the replacement file path, during the Name Translation step of request processing.
- **service** — accepts the request and runs an API application, defined by the replacement file path, during the Service step of request processing.

Configure a surrogate server

To configure a standard surrogate:

- Set the proxy server port to 80.

  ```
  Port 80
  ```

- Add a Proxy rule, prior to all other rules, that proxies all requests received on port 80 to the origin server.

  ```
  Proxy /* http://our.content.server.com/* :80
  ```

- Enable an administration port on a port other than 80.

  ```
  AdminPort 8080
  ```

This allows all HTTP traffic on port 80 to be proxied to the origin server. Traffic entering on the administration port does not match the initial wildcard proxy rule, and so it is unaffected. The remaining mapping rules are used to process the request.

Associated directives

The following directives define mapping rules:

- "Map — Change matching requests to a new request string" on page 217
- "Pass — Specify the template for accepting requests" on page 226
- "Exec — Run a CGI program for matching requests" on page 198
- "Redirect — Specify a template for requests sent to another server" on page 244
- "Proxy — Specify proxy protocols or reverse proxy" on page 238
- "ProxyWAS — Specify that requests are sent to WebSphere Application Server" on page 242
- "ReversePass — Intercept automatically redirected requests" on page 246

For more information, refer to Chapter 4, "Manually editing the ibmproxy.conf file," on page 13.

Configuration and Administration forms

The following Configuration and Administration form edits the values of the associated directives:

- **Server Configuration –> Request Processing –> Request Routing**

Note: The Configuration and Administration forms do not support the port number argument.

For more information, refer to Chapter 2, "Using the Configuration and Administration forms," on page 7.

Enable junction rewrite (optional)

The JunctionRewrite directive enables the junction rewriting routine within the Caching Proxy to rewrite responses from origin servers to ensure that server-relative URLs get mapped to the appropriate origin server when junctions are used. The junction rewriting plug-in must also be enabled. Junctions are defined by the proxy mapping rules.

When using the proxy mapping rules to define the junction, you can use the Proxy directive with or without the JunctionPrefix option.

Define the junction without the JunctionPrefix option

The following are examples of valid junctions that can be acted upon by the junction rewriting routine:

- `Proxy /shop/* http://shopserver.acme.com/*`
- `Proxy /auth/* http://authserver.acme.com/*`

The following is an example of a valid junction that will *not* be acted upon by the junction rewriting routine:

- `Proxy /* http://defaultserver.acme.com/*`

The following are examples of invalid junctions:

- `Proxy /images/*.gif http://imageserver.acme.com/images/*.gif`
- `Proxy /cgi-bin/* http://cgiserver.acme.com/cgi/perl/*`

These mapping rules have created junctions for shopserver, authserver, and b2bserver. Consider that shopserver returns an HTML document with the following URLs contained within appropriate HTML tags:

- /index.html (server relative reference)
- /images/shop.gif (server relative reference)
- buy/buy.jsp (directory relative reference)
- http://ebay.com (absolute reference)

The junction rewriting routine will rewrite the server relative references using the proxy mapping rules as follows:

- /shop/index.html (changed)
- /shop/images/shop.gif (changed)
- buy/buy.jsp (unchanged)
- http://ebay.com (unchanged)

Define the junction with the JunctionPrefix option (recommended method)

When using the JunctionPrefix option with the Proxy directive, instead of inferring the JunctionPrefix from the first URL pattern in the Proxy rule, you can declare the junction prefix in the Proxy rule using the following format:

`Proxy` *`url_patern1 url_pattern2`* `JunctionPrefix:`*`url_prefix`*

When using JunctionPrefix there is no limitation on the format of the first URL pattern. In order to support junction rewriting when *not* using the JunctionPrefix option, the proxy URL must have the following format: `Proxy /market/* http://b2bserver/*`. However, when using JunctionPrefix, the following Proxy rule is valid for junction rewriting:

```
Proxy  /market/partners/*.html http://b2bserver.acme.com/*.html
       junctionprefix:/market/partners
```

The junction rewriting routine affects the following tags:

Table 3. Tags affected by the junction rewriting routine

Tag	Attributes
!—	URL
a	href

Table 3. Tags affected by the junction rewriting routine (continued)

Tag	Attributes
applet	archive, codebase
area	href
base	href
body	background
del	cite
embed	pluginspage
form	action
input	src
frame	src, longdesc
iframe	src, longdesc
ilayer	src, background
img	src, usemap, lowsrc, longdesc, dynsrc
layer	src, background
link	href
meta	url
object	data, classid, codebase, codepage
script	src
table	background
td	background
th	background
tr	background

Note: The junction rewriting routine will not affect tags generated by JavaScript or by plug-ins within the browser.

Associated directives

The following directives are used to enable the junction rewriting routine and plug-in.

- "ServerInit — Customize the Server Initialization step" on page 250
- "Transmogrifier — Customize the Data Manipulation step" on page 259
- "JunctionRewrite — Enables URL rewriting" on page 210
- "JunctionRewriteSetCookiePath — Rewrite the path option in the Set-Cookie header, when used with JunctionRewrite plugin" on page 211
- "JunctionReplaceUrlPrefix — Replace URL instead of insert prefix when used with JunctionRewrite plugin" on page 210
- "JunctionSkipUrlPrefix — Skip rewriting URLs that already contain the prefix, when used with JunctionRewrite plugin" on page 211

For more information, refer to Chapter 4, "Manually editing the ibmproxy.conf file," on page 13.

Configuration and Administration forms

The following Configuration and Administration form can be used to enable the junction rewriting plug-in:

- **Server Configuration –> Request Processing –> API Request Processing**

Note: The Configuration and Administration forms do not support the JunctionRewrite directive.

For more information, refer to Chapter 2, "Using the Configuration and Administration forms," on page 7.

UseCookie as an alternative to JunctionRewrite

You can use cookies to store back-end server information as follows: A cookie is sent to the client browser. When the browser sends requests for the resources in the HTML page, it attaches a cookie so that Caching Proxy forwards the requests to the correct back-end server.

To use cookies as an alternative to JunctionRewrite, make the following modifications to the ibmproxy.conf file:

1. Change **JunctionRewrite on** with **JunctionRewrite on UseCookie**.
2. Comment out the JunctionRewrite plugin.

The following is a comparison of the JunctionRewrite plugin and the cookie implementation.

- JunctionRewrite plugin
  - HTML page is rewritten.
  - Does not support rewrite of script languages and applets, unless using transmogrifier plugin. See "Sample transmogrifier plugin to extend JunctionRewrite functionality" on page 45.
  - Decreased performance.
  - No limitations on back-end server configurations. Client can cross-access back-end servers in a session.
- Cookie implementation
  - HTML page is *not* rewritten. A cookie is sent to the client.
  - Client browser must enable cookie support.
  - Increased performance.
  - Some limitation on back-end server configurations. Can be used only when a client accesses on back-end server in a session.

Note: There is a known limitation when using JunctionRewrite with the UseCookie option. It will incorrectly translate URLs for all requests even though the cookie applies only to one subdirectory of the host. The following are two ways to correctly handle the URLs that are under ROOT and do not need junction:

- Place the proxy rules before the JunctionRewrite directive in the ibmproxy.conf file. (All proxy rules that are before the JunctionRewrite directive will not be rewritten.)
- Explicityly map each URL, instead of using a wildcard (*). For example:

```
Proxy /no-juntion.jpg  http://login-server/no-junction.jpg
```

Sample transmogrifier plugin to extend JunctionRewrite functionality

Customizable sample code is provided that rewrites and parses JavaScript™ (SCRIPT) and applet (APPLET) tag blocks in HTML files. Alone, the JunctionRewrite plugin cannot process the resource links in JavaScript or in parameter values of Java™.

After installing Caching Proxy, you can compile the same code and configure it to run with JunctionRewrite.

The following sample files are located in the ...samples/cp/ subdirectory, under the directory in which you downloaded the fix pack.

- Makefile (Makefile for this sample plugin)
- junctionRewrite2.h (interface for customized parser handler
- junctionRewrite2.c (implementation for above interface)
- scriptHandler.c (sample JavaScript rewrite handler)
- appletHandler.c (sample Applet block handler)
- junctionRewrite2.def (Windows plugin def file)
- junctionRewrite2.exp (Linux and UNIX plugin export file)

Chapter 11. Manage delivery of local content

The Pass and Exec mapping rules are used to deliver local content to a requesting client. By default, a Pass rule with a wildcard template is placed as the last mapping rule. This rule directs all requests that do not match previous templates to retrieve files from a target directory, which is commonly referred to as the document root directory.

When a URL is received that does not contain a file name, the Caching Proxy satisfies the request by searching the specified directory, if one is given, or the document root directory, if no directory is specified, for a file that matches the list of welcome pages specified in the configuration file. If more than welcome page is defined, the proxy server searches for the pages in the order in which they are defined. The first welcome page found is served.

The server home page is the Web page that the server delivers by default when it receives a request that contains only the server's URL without a directory or file name. As previously explained, the default wildcard mapping rule requires that the server home page is stored in the document root directory and that the filename of the home page matches a defined welcome page.

Note: Some Web browsers use the term *home page* to refer to the first page the browser loads when it is started. This document uses the term only for the server home page.

This chapter describes how to define the document root directory and welcome pages.

Define document root directory

The default document root directories are:

- On Linux and UNIX: /opt/ibm/edge/cp/server_root/pub/*lang*/
- On Windows: *drive*:\Program Files\IBM\edge\cp\server_root\pub*lang*\, or the directory you specified as your HTML directory during installation

Associated directives

The following directive defines the document root directory:

- "Pass — Specify the template for accepting requests" on page 226

For more information, refer to Chapter 4, "Manually editing the ibmproxy.conf file," on page 13.

Configuration and Administration forms

To change the document root directory in the Configuration and Administration forms, use the following procedure:

1. Select **Server Configuration –> Request Processing –> Request Routing**.
2. In the request routing table, find the row that contains the string /* (slash asterisk) in the **Request Template** column. This represents the document root directory. In the **Index** box below the table, click the number that corresponds to the number in the **Index** column for that row.

3. Click **Replace**.
4. In the **Action** drop-down list, click **Pass**.
5. Type /* in the URL Request Template field.
6. Type your new document root directory in the **Replacement File Path** field.
7. Click **Submit**.
8. After your changes are accepted, click the **Restart Server** icon (|) in the upper frame.

After restarting, the server begins to use the new document root directory.

For more information, refer to Chapter 2, "Using the Configuration and Administration forms," on page 7.

Define default welcome pages

The server looks for the home page in the document root directory, but the specific file it returns is defined by the list of welcome pages.

About welcome pages

When the server receives a URL request that does not specify a file name, it tries to satisfy the request according to a list of welcome pages set in the server's configuration file. This list defines files to use as default home pages. The server determines your home page by matching the list of welcome pages to the files in the document root directory. The first match it finds is the file returned as your home page. If no match is found, the server displays a listing of the document root directory.

To have a particular file used as your server's home page and returned when a request does not specify either a directory or file name, you must put the file in the document root directory and also make sure that its name matches one of the file names listed in the list of welcome pages.

The default configuration file defines these file names, in this order, to be used as welcome pages:

1. welcome.html or welcome.htm
2. index.html or index.htm
3. Frntpage.html

The server returns the first file it finds that matches a file name in the list. Until you create a welcome.html or index.html file and place that file in the document root directory, the server uses Frntpage.html as your home page.

For example, if you are using the default configuration and your document root directory does not contain a file named welcome.html, but does contain files named index.html and FrntPage.html, the index.html file is used as your home page.

If no home page is found, the contents of the document root directory are displayed as a directory.

Associated directives

The following directive defines the welcome pages:

- "Welcome — Specify the names of welcome files" on page 263

For more information, refer to Chapter 4, “Manually editing the ibmproxy.conf file,” on page 13.

Configuration and Administration forms

The following Configuration and Administration form defines the welcome pages:

- **Server Configuration –> Directories and Welcome Page –> Welcome Page**

For more information, refer to Chapter 2, “Using the Configuration and Administration forms,” on page 7.

Chapter 12. Manage FTP connections

The Caching Proxy proxies requests for FTP URLs to the appropriate FTP server, but it cannot be used to proxy requests from an FTP client. It can support only those FTP requests received from an HTTP client (using the ftp:// protocol scheme).

Only the GET, PUT, and DELETE methods are supported for requests for FTP files. Only the GET method is supported for requests for FTP directory listings. By default, PUT and DELETE are disabled in the Caching Proxy. For more information, refer to "Enable HTTP/FTP methods" on page 37.

This chapter describes how to protect FTP files and manage FTP server login, directory paths, and chaining.

Protect FTP files

If you have enabled the PUT method for FTP file uploading or the DELETE method for FTP file deletion, you need to define FTP proxy protection for at least PUT and DELETE requests, to prevent unauthorized file updating at your FTP server.

To protect the proxying of FTP requests, in the Configuration and Administration forms, select **Server Configuration –> Document Protection**. To create a protection setup for FTP file requests, include `ftp://` at the beginning of the request template. For example, to protect files in a directory named exams, use the template `ftp://exams/*`.

For more information on creating protection setups, see Chapter 25, "Server protection setups," on page 107.

Manage FTP server login

If no user ID and password is specified in the request URL, the Caching Proxy attempts to log into the requested FTP server anonymously (using the user ID ANONYMOUS). Many FTP servers require an e-mail address as the password for anonymous FTP. If the FTP server asks for a password for the anonymous login, the Caching Proxy sends the e-mail address specified by the WebmasterEmail directive in the configuration file.

To set the Webmaster e-mail address in the Configuration and Administration forms, select **Server Configuration –> System Management–> SNMP MIB**. The e-mail address can also be set by using the WebmasterEmail directive; for details, see the reference section: "WebMasterEMail — Set an e-mail address to receive select server reports" on page 262.

If the FTP server in the request URL requires a specific user ID and password to log in, users can enter the user ID and password in the request URL, for example:
`ftp://`*`userid`*`:`*`password`*`@`*`ftpserverhost`*`/`

If you do not want to specify the password for the FTP user ID in the request URL, users can enter only the user ID in the URL: `ftp://`*`userid`*`@`*`ftpserverhost`*.

The Caching Proxy first attempts to log into the FTP server with the specified user ID and no password. If the login is unsuccessful without a password, then the browser prompts for the password associated with the specified user ID.

For logins other than anonymous, at least the user ID must be specified in the URL. If the user ID is not specified, then anonymous login is attempted and the client is not prompted for the user ID.

Manage FTP directory paths

You must specify to the Caching Proxy whether you want the path names in FTP URLs to be interpreted as relative to the user's working directory or relative to the root directory. For example, if a user who is logged into an FTP server has a default working directory called /export/home/user1 and wants to retrieve a file called test1.exe from a subdirectory called test, the proxy uses the following URLs to retrieve the file from the FTP server, depending on how FTP URLs are interpreted:

- If *absolute* path names are set:
  `ftp://user1:user1pw@FTPhost/export/home/user1/test/test1.exe`
- If *relative* path names are set: `ftp://user1:user1pw@FTPhost/test/test1.exe`

If relative FTP URL paths are set, users can still specify an absolute path name by using the convention of escaping the initial slash character (/) with `%2F` to indicate the root directory. For example, if user1, whose working directory is /export/home/user1, wants to access a file in user2's working directory, /export/home/user2, the request `ftp://user1:user1pw@FTPhost/%2Fexport/home/user2/`*`file`* is interpreted correctly as a URL relative to the root directory / (that is, as an absolute path name), even if relative FTP URL path names have been chosen.

To specify how FTP URLs are to be interpreted, in the Configuration and Administration forms, select **Proxy Configuration –> Proxy Performance**. In the lower portion of the form, under **FTP URL paths should be:**, select **absolute paths** to specify the server's root directory or **relative paths** to specify the user's working directory as the start of the path.

This setting can also be changed in the proxy configuration file; for more information, see "FTPUrlPath — Specify how FTP URLs are interpreted" on page 203.

Manage FTP chaining

If you are chaining multiple Web proxy servers together, you can specify that requests containing FTP URLs are to be sent to a chained Web proxy server rather than directly to the FTP server. To specify a chained proxy server for FTP requests, in the Configuration and Administration forms, select **Proxy Configuration –> Proxy Chaining and Non-Proxy Domains**. The http:// protocol scheme is used to specify the URL of the chained proxy, even when chaining requests for an ftp:// protocol scheme.

To configure FTP chaining by using the proxy configuration file, see the reference section on "ftp_proxy — Specify another proxy server for FTP requests" on page 202.

Chapter 13. Customize server processing

This chapter describes how to use server-side includes to insert information into CGI programs and HTML documents that are delivered to a client. Customizing the server's error messages and resource mapping are also discussed.

Server-side includes

Server-side includes allow you to add information to CGI programs and HTML documents that the server sends to the client when acting as an origin server (that is, not to proxied or cached objects). The current date, the size of a file, the date of last change to a file are examples of the kind of information that can be sent to the client. This section describes the command format for server-side includes and explains how to make server-side include commands work in your CGI programs and HTML documents. You can also use server-side includes to customize error pages.

Considerations for server-side includes

Before using server-side includes on your server, consider performance, security, and risk issues:

- Performance can be significantly impaired when the server is processing files while sending them.
- Security can be compromised if you let ordinary users execute commands on your server. Be careful when deciding in which directories you place server-side includes and in which directories you place the **exec** command. You can minimize the security risk if you do not enable the **exec** command.
- Using server-side includes can cause some problems. For example, files cannot be referenced recursively: if you are executing the file sleepy.html and the program finds `<-- !#include file="sleepy.html" -->` , the server does not detect the error and can fail. (Referencing files nonrecursively within other files is not a problem.)

Configuration for server-side includes

To enable server-side includes, in the Configuration and Administration forms, select **Server Configuration –> Basic Settings**. Use this form to specify which of the following types of server-side includes are acceptable:

- CGI scripts
- Files
- All except CGI scripts that use the **exec** command
- None

Use this form also to specify whether to do server-side include processing for text or HTML documents in addition to other file types.

In addition, ensure that the file extension that you use for the include is recognized. In the Configuration and Administration forms, select **Server Configuration –> MIME Types and Encoding**, and use the **MIME Types** form. Note that the shtml and htmls extensions are recognized by default.

To configure your server for server-side includes by editing the directives in the proxy configuration file, see the reference sections for the following directives:

- "AddType — Specify the data type of files with particular suffixes" on page 169
- "imbeds — Specify whether server-side include processing is used" on page 208

Format for server-side includes

Include commands must be included in the HTML document or CGI program as comments. The commands have the following format:

```
<!--#directive tag=value ... -->
or
<!--#directive tag="value" ... -->
```

The quotation marks around values are optional, but they are required in order to embed spaces.

Directives for server-side includes

This section explains the directives that are accepted by the server for server-side includes. (Do not confuse these directives with the directives for the proxy configuration file, which are documented in Appendix B, "Configuration file directives," on page 161.)

config—control file processing

Use this directive to control certain aspects of file processing. Valid tags are `cmntmsg`, `errmsg`, `sizefmt`, and `timefmt`.

cmntmsg

Use this tag to specify a message that precedes the beginning of comments added by other directives. For any directive that contains text between a directive specification and "`-->`", that text is treated as a comment and is added to the file that the server sends to the client.

Example:

```
<!--#config cmntmsg="[This is a comment]" -->
<!-- #echo var="  " extra text -->
```

Result: `<!--[This is a comment] extra text -->`

Default: `[the following was extra in the directive]`

errmsg

Use this tag to specify the message that is sent to the client if an error occurs while a file is being processed. The message is logged in the server's error log.

Example:

```
<!-- #config errmsg="[An error occurred]" -->
```

Default: `"[An error occurred while processing this directive]"`

sizefmt

Use this tag to specify the format in which file size is displayed. In the following examples, `bytes` is the value used to display the number of bytes, and `abbrev` is the value used to display the number of kilobytes or megabytes.

Example 1:

```
<!--#config sizefmt=bytes -->
<!--#fsize file=foo.html -->
```

Result: `1024`

Example 2:

```
<!--#config sizefmt=abbrev -->
<!--#fsize file=foo.html -->
```

Result: `1K`

Default: `"abbrev"`

timefmt

Use this tag to specify the format used to provide dates.

Example:

```
<!--#config timefmt="%D %T" -->
<!--#flastmod file=foo.html -->
```

Result: `"10/18/95 12:05:33"`

Default: `"%a, %d %b %Y %T %Z"`

The following `strftime()` formats are valid with the `timefmt` tag:

Specifier	Meaning
%%	Replace with %
%a	Replace with the abbreviated weekday name
%A	Replace with the full weekday name
%b	Replace with the abbreviated month name
%B	Replace with the full month name
%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)
%D	Insert the date as *%m/%d/%y*
%e	Insert the month of the year as a decimal number (01-12) (Under C POSIX only, it is a 2-character, right-justified, blank-filled field)
%E[cCxyY]	If the alternative date/time format is not available, the %E descriptors are mapped to their unextended counterparts (For example, %EC is mapped to %C)
%Ec	Replace with the alternative date and time representation
%EC	Replace with the name of the base year (period) in the alternative representation
%Ex	Replace with the alternative date 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 (the same as %b)

Specifier	Meaning
%H	Replace with hour (23-hour clock) as a decimal number (00-23)
%I	Replace with hour (12-hour clock) as a decimal number (00-12)
%j	Replace with the day of the year (001-366)
%m	Replace with the month (01-12)
%M	Replace with minute (00-59)
%n	Replace with a new line
%O[deHlmMSUwWy]	If the alternative date/time format is not available, the %E descriptors are mapped to their unextended counterparts (For example, %Od is mapped to %d)
%Od	Replace with the day of the month, using the alternative numeric symbols, filled as needed with leading zeroes 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 the 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
%Oy	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 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
%u	Replace with the weekday as a decimal number (1-7), with 1 representing Monday
%U	Replace with the week number of the year (00-53), where Sunday is the first day of the week
%V	Replace with the week number of the year (01-53), where Monday is the first day of the week
%w	Replace with the weekday (0-6), where Sunday is 0

Specifier	Meaning
%W	Replace with the week number of the year (00-53), where Monday is the first day of the week
%x	Replace with the appropriate date representation
%X	Replace with the appropriate time representation
%y	Replace with the 2-digit year number within the century
%Y	Replace with the full 4-digit year number
%Z	Replace with the name of the time zone or no characters if the time zone is unknown

The operating system configuration determines the full and abbreviated month names and years.

echo—display variable values

Use this directive to display the value of environment variables specified with the `var` tag. If a variable is not found, a (`None`) is displayed. Also, **echo** can display a value set by the **set** or **global** directives. The following environment variables can be displayed:

DATE_GMT
The current date and time in Greenwich Mean Time. The formatting of this variable is defined using the **config timefmt** directive.

DATE_LOCAL
The current date and local time. The formatting of this variable is defined using the **config timefmt** directive.

DOCUMENT_NAME
The name of the topmost document. If the HTML was generated by a CGI, this variable contains the name of the CGI.

DOCUMENT_URI
The full URL requested by the client, without the query string.

LAST_MODIFIED
The date and time that the current document was last modified. The formatting of this variable is defined using the **config timefmt** directive.

QUERY_STRING_UNESCAPED
The search query sent by the client. This is undefined unless HTML was generated by a CGI.

SSI_DIR
The path of the current file, relative to SSI_ROOT. If the current file is in SSI_ROOT, this value is `"/"`.

SSI_FILE
The file name of the current file.

SSI_INCLUDE
The value used in the include command that retrieved the current file. This is not defined for the topmost file.

SSI_PARENT
The path and file name of the file containing the include command that retrieved the current file, relative to SSI_ROOT.

SSI_ROOT
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 -->
```

exec—specify CGI programs

Use this directive to include the output of a CGI program. The exec directive discards any HTTP headers that the CGI outputs *except* for the following:

Content-type
Determines whether to parse the body of the output for other includes

Content-encoding
Determines whether translation from EBCDIC to ASCII needs to be done

Last-modified
Replaces the current last-modified header value unless the current value is later than the specified value

cgi—specify CGI program URL

Use this directive to specify the URL of a CGI program.

In this example, **program** is the CGI program to be executed, and **path_info** and **query_string** represent one or more parameters passed to the program as environment variables:

```
<!--#exec cgi="/cgi-bin/program/path_info?query_string" -->
```

This example shows the use of variables:

```
<!--#exec cgi="&path;&cgiprog;&pathinfo;&querystring;" -->
```

flastmod—display date and time document was last changed

Use this directive to display the last date and time the document was changed. The formatting of this variable is defined by the **config timefmt** directive. The **file** and **virtual** tags are valid with this directive, and their meanings are defined as follows.

Directive formats:

```
<!--#flastmod file="/path/file" -->
<!--#flastmod virtual="/path/file" -->
```

file Use this tag to specify the name of a file. For **flastmod**, **fsize**, and **include**, **file** is assumed to be relative to **SSI_ROOT** if preceded by a **'/'**. Otherwise, it is relative to **SSI_DIR**. The file specified must exist either in **SSI_ROOT** or in one of its descendants. For example:

```
<!--#flastmod file="/path/file" -->
```

virtual
Use this tag to specify the URL of a virtual path to a document. For **flastmod**, **fsize**, and **include**, **virtual** is always passed through the server's mapping directives. For example:

```
<!--#flastmod virtual="/path/file" -->
```

Example:

```
<!--#flastmod file="foo.html" -->
```

Result: 12May96

fsize—display file size

Use this directive to display the size of the specified file. The formatting of this variable is defined by the **config sizefmt** directive. The **file** and **virtual** tags are valid with this directive, and their meanings are the same as defined previously for the **flastmod** directive.

Example:

```
<!--#fsize file="/path/file" -->
<!--#fsize virtual="/path/file" -->
```

Result: 1K

global—defines global variables

Use this directive to define global variables that can be echoed later by this file or by any included files.

Example:

```
<!--#global var=VariableName value="SomeValue" -->
```

For example, to reference a parent document across the virtual boundary, you need to set a global variable DOCUMENT_URI. You also need to reference the global variable in the child document. This example shows the HTML coding you need to insert in the parent document:

```
<!--#global var="PARENT_URI" value=&DOCUMENT_URI; -->
```

This example shows the HTML coding you need to insert in the child document:

```
<!--#flastmod virtual=&PARENT_URI; -->
```

include—includes a document in output

Use this directive to include the text from a document in the output. The **file** and **virtual** tags are valid with this directive, and their meanings are the same as defined above for the **flastmod** directive.

set—set variables to be echoed

Use this directive to set a variable that can be echoed later, but only by this file.

Example:

```
 <!--#set var="Variable 2" value="AnotherValue" -->
```

While defining a directive, you can echo a string in the middle of value. For example:

```
<!--#include file="&filename;" -->
```

Variables: A server-side set directive is generally followed by an echo directive, so that it looks for the set variable, echoes where the variable is found, and proceeds

with the function. It can contain multiple references to variables. Server-side sets also allow you to echo a variable already set. If a set variable is not found, nothing is displayed.

When a server-side set encounters a variable reference inside a server-side include directive, it attempts to resolve it on the *server* side. In the second line of the following example, the server-side variable `&index;` is used with the string `var` to construct the variable name `var1`. The variable `&var1;` is then assigned a value by escaping the & in `&ecirc;` so that it is not recognized as a variable. Instead, it is used as a string to create the value `fr&ecirc;d`, or *fred* with a circumflex over the e. The variable `&ecirc;` is a client-side variable.

```
<!--#set var="index" value="1" -->
<!--#set var="var&index;" value="fr\&ecirc;d" -->
<!--#echo var="var1" -->
```

Characters that can be escaped (called escaped variables) are preceded with a backslash (\) and include the following:

Character	Meaning
\a	Alert (bell)
\b	Backspace
\f	Form feed (new page)
\n	New line
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\′	Single quotation mark
\″	Double quotation mark
\?	Question mark
\\	Backslash
\-	Hyphen
\.	Period
\&	Ampersand

Customizing error messages

You can customize the error messages that the Caching Proxy returns, and you can define specific messages for particular error conditions. In the Configuration and Administration forms, select **Server Configuration –> Error Message Customization**. Use this form to select an error condition and specify a particular HTML file to use for that condition.

To customize error messages by editing directives in the proxy configuration file, see the reference section for the directive “ErrorPage — Specify a customized message for a particular error condition” on page 196.

Real Time Streaming Protocol (RTSP) redirection

WebSphere Application Server, Version 6.0.2 introduces streaming media support in the form of the RTSP Redirector. RTSP enables the Caching Proxy to act as a first point of contact with media players and to redirect their requests to an appropriate proxy server or to a content server that provides the requested media content.

RTSP, the Real Time Streaming Protocol, is defined in RFC 2326. It is an Internet standard protocol for controlling data streams. Although it does not include technology for *delivering* streams, it is flexible enough that it can be used to control data streams that are unrelated to video or audio playback.

About RTSP redirection

The RTSP redirection feature allows Caching Proxy to redirect requests for any streaming media sessions controlled by RTSP. These include the following types of media:

- RealNetworks recorded audio
- RealNetworks recorded video
- RealNetworks live streams (audio and video)
- Microsoft Media Player files
- Apple Quicktime media files

Any player that can be configured to contact a proxy server on its RTSP port (typically 554) can use this framework in the Caching Proxy to have its requests handled by the RTSP redirector.

The RTSP redirector does not cache or directly proxy media presentations. The RTSP redirector must be used in conjunction with a third-party streaming media server to provide either or both of those functions. The Caching Proxy with RTSP redirector must have network access to one or more RTSP proxy servers.

RTSP limitation

This feature is subject to the following limitation:

Currently, only RealNetworks technologies are supported. These include the RealProxy proxy server, the RealServer origin server, and the RealPlayer media player.

RTSP enhancement

Previously, the RTSP Redirector was subject to the limitation that all requests to the same origin server for any URL were redirected the same way. Redirection based on file names or other portions of the requested URL was not possible. This limitation no longer applies. The RTSP Redirector now uses the complete URL from received requests along with the threshold value (rtsp_proxy_threshold) set in the Caching Proxy configuration file to determine whether to redirect the client request to the origin server or to a proxy server. Requests to the same origin server are now handled individually.

Configuring RTSP redirection

The following configuration file directives are used to control RTSP redirection. The settings for these directives are not refreshed by a server restart. The server must be completely stopped and then started again before changes to these directives take effect.

Chapter 14. Configure header options

When requesting documents, Web clients send headers that provide additional information about the browser or the request. Headers are automatically generated when a request is sent.

The Caching Proxy allows several options for customizing header information to keep it hidden from the destination server. Although substituting a more generic header for the actual header has the advantage of increasing client anonymity, it has the disadvantage of disabling header-based page customization that is written into some Web pages.

Headers typically use this form:

```
User-Agent: Mozilla 2.02/OS2
Client-IP: 45.37.192.3
Referer: http://www.bigcompany.com/WebTrafficExpress/main.html
```

This header includes the following fields:

- **User-Agent:** provides information about the browser and operating system.
- **Client-IP:** provides the IP address of the client requesting the URL.
- **Referer:** provides the destination server with the URL of the referring link to this page.

Most headers can be blocked by the appropriate proxy configuration settings. However, some header fields are required by origin servers, so blocking them can cause Web pages to be displayed incorrectly (For example, in certain cases blocking the "host" header field can cause users to see the wrong Web page). For further information on header fields, refer to the HTTP Version 1.1 specification.

Associated directives

To change header options by editing the proxy configuration file, see the reference sections for the following directives:

- "NoProxyHeader — Specify the client headers to block" on page 224
- "ProxyFrom — Specify a client with a From: header" on page 240
- "ProxyIgnoreNoCache — Ignore a reload request" on page 240
- "ProxySendClientAddress — Generate the Client IP Address: header" on page 241
- "ProxyUserAgent — Modify User Agent string" on page 241
- "ProxyVia — Specify format of HTTP header" on page 242

For more information, refer to Chapter 4, "Manually editing the ibmproxy.conf file," on page 13.

Configuration and Administration forms

You can use two Configuration and Administration forms to specify header options:

- Select **Proxy Configuration –> Privacy Settings**. In the **Privacy Settings** form, set the following:

- **Forward client's IP address to destination server**

  Check this box if you want the requesting client's IP address to be forwarded to the destination (content) server. If you do not check this box, the destination server receives the IP address of the proxy server. Leaving this box unchecked increases the clients' anonymity while surfing the Web.
- **User-agent string**

  Type the string to send in the header to the destination server to replace the type of browser and operating system that a client is using. For example: specifying `Caching Proxy 4.0` replaces `Mozilla 2.02/OS2` in the following header:

  ```
  Content-Type:MIME
  User-Agent: Mozilla 2.02/OS2
  Referer: http://www.ics.raleigh.ibm.com/WebTrafficExpress/main.html
  Pragma:no-cache
  ```
- **From:**

  Type the e-mail address that the destination server reads when it parses the "From:" header. You might want to specify the e-mail address of the proxy administrator because the administrator is the person who needs to receive reports of any problems.
- Click **Submit** to make the changes to the configuration file.

- Select **Proxy Configuration –> Proxy Header Filtering**. Use this form to list HTTP headers to block:
  1. Click **Add** or **Remove**, and indicate an index position for the blocked header.
  2. Type the client HTTP header to block. (Refer to the HTTP 1.1 specification for a complete list and explanation of headers.)
  3. Click **Submit** to make the changes to the configuration file.

For more information, refer to Chapter 2, "Using the Configuration and Administration forms," on page 7.

Chapter 15. About the application programming interface

The application programming interface (API) is fully detailed in the *Programming Guide for Edge Components*. API directives within the configuration file enable plug-in routines that are called during specific steps within the request processing workflow. These plug-in routines can replace or run in addition to built-in routines.

Associated directives

Following are the API directives:

- "Authentication — Customize the Authentication step" on page 173
- "Authorization — Customize the Authorization step" on page 173
- "Error — Customize the Error step" on page 195
- "Log — Customize the Log step" on page 214
- "Midnight — Specify the API plugin used to archive logs" on page 221
- "NameTrans — Customize the Name Translation step" on page 222
- "ObjectType — Customize the Object Type step" on page 225
- "PostAuth — Customize the PostAuth step" on page 230
- "PostExit — Customize the PostExit step" on page 230
- "PreExit — Customize the PreExit step" on page 231
- "ServerInit — Customize the Server Initialization step" on page 250
- "ServerTerm — Customize the Server Termination step" on page 251
- "Service — Customize the Service step" on page 252
- "Transmogrifier — Customize the Data Manipulation step" on page 259
- "TransmogrifiedWarning — Send warning message to client" on page 259

For more information, refer to Chapter 4, "Manually editing the ibmproxy.conf file," on page 13.

Configuration and Administration forms

The following Configuration and Administration form edits the values of the associated directives:

- **Server Configuration –> Request Processing –> API Request Processing**

For more information, refer to Chapter 2, "Using the Configuration and Administration forms," on page 7.

Part 4. Configuring proxy server caching

This section discusses the proxy cache and how to configure it. The cache can be set up to store files in memory (memory cache) or on one or more storage devices (disk cache). A cache refresh agent can be configured to preload frequently requested files into the cache, and various URL filters can be applied to caching. This section also discusses cache sharing by using remote cache access or the Internet caching protocol (ICP) plug-in, removing obsolete files with cache garbage collection, and caching dynamically generated files.

This part contains the following chapters:

Chapter 16. Overview of proxy server caching

Caching is a feature in which the proxy server saves local copies of the files that clients request so that it can serve them quickly from the cache when they are requested again by the same or other clients.

The Caching Proxy is HTTP 1.1 compliant and generally follows the HTTP 1.1 protocol for caching and determining the freshness of documents.

This chapter discusses some features of the proxy server cache. For those features that can be configured, details about how to set the appropriate values are included in subsequent chapters.

Cache storage

The proxy server can store the cache on a physical storage device or in system memory. The type of cache storage that is better for your system depends on the capabilities of your hardware and on whether it is more important to have fast cache response or to have a larger number of items stored in the cache. Response time for a memory cache typically is faster than for a disk cache, but the size of a memory cache is limited by the amount of RAM in the proxy server machine. The size of a disk cache is limited by the size of the storage device, which typically is much larger than the amount of RAM.

For disk caches, the Caching Proxy uses raw disk caching, which means that the proxy server writes directly to the cache device without using the operating system's read and write protocols. The storage device for a disk cache must be prepared by using the **htcformat** command. Details about **htcformat** are included in the section Chapter 17, "Configuring basic caching," on page 73.

The cache index

In either a memory or disk cache, the Caching Proxy also uses system memory space to hold an index of the cache, which reduces the processing time to find cached files.

The Caching Proxy's cache directory structure and its lookup methods are different from those of other proxy servers. The Caching Proxy maintains an index in memory with information about the files in the cache. Using RAM for lookup instead of a disk or other medium results in faster file lookup and retrieval.

The index includes URLs, cache locations, and expiry information for cached objects. For this reason, the amount of memory necessary to hold the index is proportional to the number of objects in the cache.

When a request is received from a client, the proxy checks the cache index in memory for that URL.

- If the file is not in the index, the request is made to the destination server.
  - The URL is then checked to determine whether the retrieved file can be cached. If allowed, the proxy server caches the retrieved file.
  - The cache index is then updated with URL, location, and expiry information for the newly cached object.

- If the file is in the index:
  - The expiry information is checked to determine whether the cached file is fresh.
    - If the object has expired, the destination server is contacted, and the expired object is replaced by the newly retrieved document. Expiry information is updated in the cache index.
    - If the object has not expired, the document is served from the proxy cache.

FTP caching

When the proxy is configured to cache requests, it can cache FTP file requests as well as HTTP file requests. However, because FTP files do not contain the same type of header information as HTTP files, expiration dates for cached FTP files are calculated differently than for other cached files.

When a request is made to the FTP server to retrieve a file, the proxy first sends to the FTP server a LIST request for the file to obtain FTP directory information about the file. If the FTP server responds to the LIST request with a positive completion reply and the directory information for the file, the proxy creates an HTTP Last-Modified header with the date parsed from the FTP directory information. The caching function of the proxy then uses this Last-Modified header, together with the value set in the CacheLastModifiedFactor directive in the configuration file, to determine the length of time that the FTP file remains in the cache before expiring.

For more information on how the Last-Modified header and the CacheLastModifiedFactor directive are used to determine the length of time a file remains in the cache, see Chapter 19, “Maintaining cache content,” on page 81.

FTP files that are retrieved for a specific user ID rather than by anonymous login are considered to be private files and are not cached.

DNS caching

In addition to caching Web content, the proxy server performs domain name server (DNS) caching. For example, when a client requests a URL from www.myWebsite.com, the proxy asks its DNS server to resolve the host name www.myWebsite.com to an IP address. The IP address is then cached to improve response time for subsequent requests to that host name. DNS caching is automatic and cannot be reconfigured.

Cache exclusions

Some files and documents are never cached. These include the following:

- Files returned from requests using HTTP methods other than GET, such as POST and PUT.
- Any documents requiring authentication, unless caching such documents is specifically allowed by the origin server.
- The dynamic output of any CGI script (because this is unique each time it is requested). Dynamically generated results from servlets and JavaServer Pages (JSP) executed by IBM WebSphere Application Server can be cached if dynamic caching is enabled. Refer to Chapter 22, “Caching dynamically generated content,” on page 97 for details.

- Any file returned from a URL containing a question mark (?), unless query caching is specifically allowed. (Refer to Chapter 18, “Controlling what is cached,” on page 77 for information about configuring query results caching.)

It is possible to further restrict the items cached by setting caching filters. For example, you might not want the proxy server to cache files that are served locally from the proxy. Refer to Chapter 18, “Controlling what is cached,” on page 77 for details.

Cache management

Managing a cache involves many factors. As a server administrator, you can specify the following:

- What documents are cached (refer to Chapter 18, “Controlling what is cached,” on page 77 for details).
- How many documents can be cached (refer to Chapter 17, “Configuring basic caching,” on page 73 for details).
- How long cached documents are considered current (refer to Chapter 19, “Maintaining cache content,” on page 81 for details).
- How frequently the cache is purged (garbage collection) and what type of files tend to be kept (refer to Chapter 19, “Maintaining cache content,” on page 81 for details).
- How cached documents are indexed (refer to Chapter 17, “Configuring basic caching,” on page 73 for details).
- When the cache is refreshed (refer to Chapter 20, “Configuring the cache agent for automatic refreshing and preloading,” on page 87 for details).
- Remote cache access (refer to Chapter 21, “Using a shared cache,” on page 93 for details).
- How logs are kept and archived (refer to Chapter 17, “Configuring basic caching,” on page 73 for details).

In addition, adjustments can be made to cache configuration in order to improve the overall performance of the Caching Proxy. For details about performance tuning, see Chapter 23, “Tuning the proxy server cache,” on page 101.

Chapter 17. Configuring basic caching

If you used the default settings in the Edge components Product Setup Program to install the Caching Proxy, caching is enabled and the cache is stored in memory. You might want to adjust the following basic cache settings to customize the cache for the needs of your system.

If you did not use the setup program, configure these settings to enable caching.

The basic steps necessary to configure caching are the following:

1. Enable caching.
2. Configure cache storage.

After configuring basic cache settings, you might want to add or change settings for the following features.

- Customize cache memory.
- Save or load cache memory to disk.
- Restrict what will be cached by using URL filters.
- Expand what will be cached by enabling caching for query results or dynamically generated files.
- Configure cached file expiration and garbage collection.
- Configure automatic cache refresh and preload.
- Configure cache sharing with remote cache access (RCA) or Internet caching protocol (ICP).
- Configure logging.

Instructions about changing each of these settings are given or referred to in this chapter.

1. Enable caching

To enable caching, set the Caching directive to on, or check the **Enable proxy caching** box on the **Cache Configuration –> Cache Settings** configuration form. If you do not specify a cache device, the cache will be stored in memory. To create a disk cache, follow the steps in "2. Configure cache storage."

2. Configure cache storage

The tasks for configuring cache storage depend on whether you use a memory cache or a disk cache.

To use a memory cache, customize the Cache Memory setting so that it includes enough memory to hold the contents of a cache. See "Set cache memory" on page 74 for recommended cache memory sizes.

To use a disk cache, you must do the following:

1. Prepare a storage device to hold the cache.

   The cache requires a specially formatted device. Devoting an entire device or disk partition to the cache is recommended. The minimum size for a cache is 16392 KB.

To format the cache device:

a. Choose a device to hold the cache. Ensure that no other program is using that storage space and that the device can be accessed as a raw (or character-formatted) device.

b. Format the device by using the **htcformat** command. The syntax is as follows:

```
htcformat raw_device_path [-blocksize block_size]
     [-blocks number_of_blocks]
```

The -blocksize and -blocks arguments are optional. The default block size is 8192 bytes. If the number of blocks is not specified, the disk partition will be filled with as many blocks as it can contain.

When specifying the device path, be sure to specify the raw device path.

- On AIX platforms, the raw device path for a logical volume defined as /dev/lv02 is /dev/rlv02
- On Linux platforms, you must first run the **raw** command prior to running **htcformat** to associate the raw device path with the real SCSI drive sdb1.

  ```
  raw /dev/raw/raw1 dev/sdb1
  ```

- On HP-UX and Solaris platforms, the raw device path for a partition defined as /dev/dsk/c0t0d0s0 is /dev/rdsk/c0t0d0s0
- On Windows platforms, the raw device path for a device defined as e: is \\.\e:

See the reference material for your file system for additional information about accessing raw devices.

2. Specify the cache device by using the CacheDev directive or the **Cache Settings** configuration form. You can specify more than one device.

CAUTION:
On Windows systems, the htcformat command does not automatically mark the cache device as unwritable.

If the operating system attempts to write to the cache device, cached data can be lost. To avoid this, you can use the Windows Disk Manager utility to prepare the disk before using the htcformat command. To prepare the disk, use the disk utility to delete the device or partition you want to use, then recreate it without formatting it. This causes the system to consider the device unavailable for system storage.

Optional customizations

Set cache memory

Set the value in the CacheMemory directive (or in the **Cache Memory** field of the **Cache Settings** configuration form), according to the following principles. The amount of memory set in this value is used for cache infrastructure support, including the cache index, and, if memory caching is configured, to store the contents of the cache.

Minimum values

For optimum performance of disk caches, a minimum cache memory value of 64 MB is recommended for cache infrastructure support, including the cache index. As the cache size increases, the cache index increases, and more cache memory is

required to store the index. A cache memory value of 64 MB is large enough to provide cache infracture support and store a cache index for a disk cache of up to approximately 6.4 GB. For larger disk caches, the cache memory should be 1% of the cache size.

For memory caches, the cache memory value is the amount of memory set aside for the cache infrastructure support and the cache itself. A minimum cache memory value of 64 MB is recommended.

Maximum value

If too much physical memory is allocated for a memory cache, undesirable operations such as "out of memory" errors or proxy server failures can possibly occur. The value limitations for cache memory are due to the limitations of a 32-bit application. Because the Caching Proxy is a 32-bit application, it can use a maximum of 2 GB of memory.

The Caching Proxy allocates the memory defined by the CacheMemory directive and uses it as the cache to store objects. Additional memory must be allocated, whether it is a memory cache or a raw disk cache, for data structures for the cache, network I/O and connection buffers, session buffers, and memory for the main process and for all the threads. Furthermore, it is possible that requests from some clients might need to allocate a larger memory pool block than the default. Therefore, if the CacheMemory directive is set close to the 2-GB mark, it is possible that the Caching Proxy might not have enough memory to operate, especially under high request loads.

It is recommended that the value of the CacheMemory directive be less than or equal to 1600 MB. Setting the value higher than 1600 MB interferes with the memory that the Caching Proxy needs for its normal operation, and causes adverse side effects. These side effects typically include but are not limited to increased CPU usage (possibly up to 100% usage), out-of-memory errors, and sluggish performance. If an overall larger cache size is required, use cache devices or implement a shared cache configuration with RCA or ICP.

Save or load cache memory to disk

You can import and export cache contents to or from a dump file. This is useful when cache memory gets lost during restart, or when deploying the same cache for multiple proxies.

Set caching filters

Filters can restrict what is cached by matching the form of the URL request. See Chapter 18, "Controlling what is cached," on page 77 for details about setting filters.

Configure caching for query results and dynamically generated files

Optionally, you can configure the proxy server to cache results of query requests. By default, URLs that contain a question mark (?) are not cached. Refer to "Caching query responses" on page 77 for details.

Another option is to cache results of servlet or JSP execution from an IBM WebSphere Application Server. Refer to Chapter 22, "Caching dynamically generated content," on page 97 for details.

Configure file expiration and garbage collection

Refer to Chapter 19, “Maintaining cache content,” on page 81 for information about configuring when files in the cache expire and how obsolete files are removed.

Configure automatic preloading

The cache can be configured to automatically refresh the most popular files on a daily basis, before they are requested. Refer to Chapter 20, “Configuring the cache agent for automatic refreshing and preloading,” on page 87 for information.

Configure cache sharing

Under certain circumstances, using a shared cache increases the likelihood that a requested file is found in the cache. Refer to Chapter 21, “Using a shared cache,” on page 93 for information.

Configure logging

Maintaining concise and accurate logs is important for managing the Caching Proxy. Part 6, “Monitoring the Caching Proxy,” on page 135 contains information about configuring and using proxy server logs.

Chapter 18. Controlling what is cached

The caching proxy offers several filtering methods to control which files, documents, and other objects are cached. These include the following features:

- URL-based caching filters
- Query response caching
- Caching locally served files
- Partial URL-based caching
- Caching files based on part of the request URL
- Caching dynamically generated files — refer to Chapter 22, "Caching dynamically generated content," on page 97

Note: The **Cache Configuration** –> **Cache Behavior** Configuration and Administration form contains an option labeled **Cache based on incoming URL**. (The corresponding configuration file directive is named CacheByIncomingURL.) This directive refers to the file name of the cached file. Check this box to base the file name of the cached file on the incoming URL; if this box is not checked, the file name is based on the outgoing URL.

Configuring URL-based caching filters

The proxy server can be configured to compare requests to a URL template in order to determine whether a file is cached. This feature is configured by setting templates for requests whose files are *always* cached, and separate templates for requests whose files must *never* be cached. Multiple templates can be used.

A similar system is used to enable query response caching. Refer to "Caching query responses" for information.

To set URL caching filters by editing the ibmproxy.conf file, refer to "CacheOnly — Cache only the files with URLs that match a template" on page 181 and"NoCaching — Specify that files with URLs that match a template are not cached" on page 223.

To set URL caching filters in the Configuration and Administration forms, use the **Cache Configuration** –> **Cache Behavior**: **Cache filtering by URL** field. Use this section to specify URLs whose files are always cached, or to specify URLs whose files are never cached. To specify two lists, one of files to always cache and one of files to never cache, create one list and then click **Submit** before creating the other list.

Caching query responses

The responses returned from queries (URL requests that contain a question mark) can be cached by using caching filters. This feature can be useful in reverse proxy (surrogate) scenarios if many clients make the same query request.

Query caching can be configured by editing the CacheQueries directive in the ibmproxy.conf configuration file. The CacheQueries directive has the following options:

- Always — all query responses from hosts matching the template will be cached, if they are cacheable by HTTP 1.1 standards.
- Public — query responses from hosts matching the template will be cached if they contain the ″Cache-control: public″ header or a forced revalidation header, and they are cacheable by HTTP 1.1 standards.

Additional information about these options is available in “CacheQueries — Specify cache responses to URLs containing a question mark (?)” on page 182.

To configure query response caching in the Configuration and Administration forms, use the **Cache Configuration** –> **Cache Behavior**: **Cache Query Response filtering by URL** field. To specify two lists, create one list and then click **Submit** before creating the other list.

Additional requirements for query response caching

In addition to configuring the query caching setting, ensure that the following settings are configured properly to enable query responses to be cached. Refer to “Configuring cache freshness” on page 84 for information about setting these options by using the Configuration and Administration forms.

- CacheTimeMargin — This directive specifies a minimum expiration time; files with expiration times below this minimum are not cached. Because query responses sometimes have very short expiration times, setting this directive to a lower setting allows more query responses to be cached. Refer to “CacheTimeMargin — Specify the minimum lifetime for caching a file” on page 183 or use the **Cache Expiration Settings** form, which is described in “Configuring cache freshness” on page 84.
- CacheDefaultExpiry — This directive specifies the expiration time for files that do not have an explicit expiration date or a last-modified date from which to compute expiration time. Increasing this setting for HTTP requests from the default of 0 allows more query responses to be cached. However, changing the setting in this manner also increases the risk that stale content might be served from the cache. Refer to “CacheDefaultExpiry — Specify the default expiration time for files” on page 176 or use the **Cache Expiration Settings** form, which is described in “Configuring cache freshness” on page 84.
- CacheLastModifiedFactor — This directive is used to calculate an expiration date for files that have a last-modified date but no explicit expiration date. Setting the factor for HTTP files to a higher value increases the amount of time an HTTP file resides in the cache without being revalidated. Changing the setting in this manner also increases the risk that stale content might be served from the cache. Refer to “CacheLastModifiedFactor — Specify the value for determining expiration dates” on page 178 or use the **Last Modified Factor** form, which is described in “Configuring cache freshness” on page 84.
- Optionally, set the SignificantUrlTerminator directive and the AggressiveCaching directive. Refer to “SignificantURLTerminator — Specify a terminating code for URL requests” on page 252 and “AggressiveCaching — Specify caching for noncacheable files” on page 171.

Caching locally served files

Because it is typically inefficient to cache files that are served from the proxy server, files originating in the server’s local domain are not cached by default. To cache objects that originate in the server’s local domain, check the **Cache local domain files** box on the **Cache Configuration** –> **Cache Behavior** Configuration and Administration form. Alternatively, set the CacheLocalDomain directive in the proxy configuration file to `on`.

Caching files by partial URL

Items can be cached based only on a specified (significant) part of the incoming URL, instead of the full URL. This is useful in transaction-model Web serving or for dynamic caching, because the same response is often returned for varying incoming requests when significant parts of the incoming requests' URLs are identical.

You cannot use the Configuration and Administration forms to specify caching based on partial URLs. Instead, use the SignificantUrlTerminator directive in the proxy configuration file to specify a terminating code for URL requests. This specification causes the Caching Proxy to evaluate only characters before the terminating code when it processes the request and determines whether the requested file is cached. When more than one terminator code is defined, the Caching Proxy compares incoming URLs against the terminator codes in the order in which they are defined in the ibmproxy.conf file. See "SignificantURLTerminator — Specify a terminating code for URL requests" on page 252 for more information.

Related configuration file directives

To set caching filters by directly editing the proxy configuration file, see the reference sections for the following directives:

- "NoCaching — Specify that files with URLs that match a template are not cached" on page 223
- "CacheOnly — Cache only the files with URLs that match a template" on page 181
- "CacheQueries — Specify cache responses to URLs containing a question mark (?)" on page 182
- "CacheLocalDomain — Specify whether to cache the local domain" on page 178
- "SignificantURLTerminator — Specify a terminating code for URL requests" on page 252

See Chapter 16, "Overview of proxy server caching," on page 69 for information on documents that cannot be cached.

Chapter 19. Maintaining cache content

Because caching involves making and saving a copy of the served file, some routine maintenance is required for the cache to function properly. Cached files must be checked for *freshness* and invalidated when they are no longer consistent with the files on the origin server. This file expiration process is explained in "File expiration." Also, invalidated or unused files must be removed from the cache to make room for new files. This cache-purging process is described in "Garbage collection" on page 85.

File expiration

Keeping cached objects consistent with the original object on the content server is known as maintaining cache freshness. For each document or other object that it caches, the Caching Proxy computes a time at which the object expires.

For HTTP pages, the header of the document, generated by the content server, contains the expiration information.

Because the FTP protocol does not include equivalent expiration information, the Caching Proxy generates its own `Last-Modified:` header for FTP files, based on the FTP directory information for each file, and uses this information to compute expiry times. If the proxy server cannot obtain directory information for the file from the FTP server, the default value that matches the FTP URL is used. In addition, because there is no standard date format for FTP servers, the Caching Proxy might be unable to understand the date and time sent by some FTP servers. In that case, the proxy server's default expiry time value is used. This procedure allows the proxy to manage the caching of HTTP pages and FTP files in a similar manner.

Expiration can be specified by a content server in one of several ways (in order of preference):

1. The content server specifies a header saying `Cache-control: s-maxage=`*n*. This tells the proxy that the object is fresh for *n* seconds after it is received.
2. The content server specifies a header saying `Cache-control: max-age=`*n*. This tells the proxy that the object is fresh for *n* seconds after it is received.
3. The content server specifies a header saying: `Expires:` *n*. This tells the proxy that the object is fresh until the time specified by *n*.
4. The content server indicates when the document was last modified, using a `Last-Modified:` *n* header. The proxy server computes the length of time since the document was last modified, multiplies this by the Cache Last Modified factor set in the proxy configuration file, and assumes that the document is valid for that length of time. For example, if the content server indicates that the document was last modified one week (seven days) ago, and the Cache Last Modified factor is 0.14, then the proxy server assumes that the document is valid for about one day. See "Configuring cache freshness" on page 84 for instructions on setting the Cache Last Modified factor.
5. If none of the above information is specified by the content server, the Caching Proxy looks for the Cache Default Expiry setting that matches the current URL and uses that for the expiry time. See "Configuring cache freshness" on page 84 for instructions on setting the Cache Default Expiry values.

After the expiry time is computed as just described, the Caching Proxy checks to see whether there is a Minimum Hold value that applies for this URL. If there is, and the time it specifies is longer than the computed expiry time, then the time specified by the Minimum Hold value is used as the object's expiry time. This is true even if the Caching Proxy computes an expiry time of 0 minutes for a document. Therefore, to avoid serving stale content, be cautious about using the Minimum Hold setting. (To set the Minimum Hold value, use the CacheMinHold directive or the **Cache Configuration –> Cache Expiry Settings**: **URL Expiration** setting. Refer to "Configuring cache freshness" on page 84 for additional information.)

The final expiry time value is checked against the time specified in the Time Margin setting. If the expiry time is greater than the Time Margin value, the document is cached; otherwise, it is not added to the cache. (To set the Time Margin value use the CacheTimeMargin directive or see the instructions in "Configuring cache freshness" on page 84.)

If the document is found in the cache, but it has expired, the Caching Proxy issues a special request known as an *if-modified-since* request to the content server. This request causes the content server to send the document only if it has been modified since it was last received by the proxy. If the document has not been modified, the content server sends a message indicating that, and does not resend the page. In that case, the proxy serves the cached document. For FTP files, the proxy server simulates this if-modified-since process. If it determines that the file has not been changed at the FTP server, it serves the file from the cache. Otherwise, it gets the newer version from the FTP server.

Additional information about cache freshness

- Almost all static Web documents (as opposed to dynamically generated documents) include a Last-Modified header. This is the most common way that proxies compute expiry times for documents and the first method that the Caching Proxy tries for FTP files. If this fails, the proxy refers to the Default Expiry values.
- Very few documents use a Cache-control: s-maxage, Cache-control: max-age, or Expires: header.
- Dynamically generated pages, which frequently are not cacheable, can include a header saying `Expires: 0` or `Cache-control: no-cache`, which mean that the document expires immediately. For information about caching dynamically generated files from IBM WebSphere Application Servers, see Chapter 22, "Caching dynamically generated content," on page 97.
- Be cautious when setting the Default Expiry value to anything other than 0 minutes for URLs using the HTTP: syntax. Many dynamically generated pages include none of the expiration headers and are therefore subject to the Default Expiry value. Setting Default Expiry to more than 0 minutes allows the proxy to cache those objects, but this might mean that users get out-of-date content (or unexpected results from CGI programs or servlets).
- In the following circumstances, the proxy server revalidates documents with the server for every request, regardless of whether the cached document is expired:
  - The document includes one of the following headers:
    - Cache-control: s-maxage
    - Cache-control: must-revalidate
    - Cache-control: proxy-revalidate

- The document requires user credentials but is allowed to be cached by the server.
- The document contains a Cache-Control: no-cache header but is cached anyway (due to aggressive caching).

About dates in FTP

Because the FTP protocol does not define dates and times as strictly as the HTTP protocol does, several factors can cause the Last-Modified header generated by the proxy for FTP files to be slightly different from the actual file date. These factors include the following:

- Unlike the HTTP protocol, the FTP protocol does not specify that returned dates must be in Greenwich Mean Time (GMT). The date returned by the FTP server is likely to be in the FTP server's local time. Because the proxy has no way of determining what time zone the FTP server is running in, it interprets the time as in its own time zone. An exception to this is the Windows FTP server, which returns dates in GMT. If the proxy detects that the FTP server is running on Windows systems, it assumes that the directory date is in GMT.
- Some FTP servers specify the date in the returned directory information in the format of *Month Day Year* only, and do not include the actual hours or minutes information for the date specified. If the FTP server does not return hour and minute information for the file, the proxy assumes that the file was last modified on the latest possible hour and minute of the date returned by the FTP server. For example, if the FTP server returns directory information for a file indicating that the file was last modified on October 13, 1998, but does not include information on the hours or minutes, the proxy assumes that the file was modified at 11:59:59 p.m. on October 13, 1998. Then, if the FTP server is not a Windows FTP server, the proxy converts this date from its own local time zone to the corresponding GMT.

When an FTP file expires from the cache, the proxy simulates the HTTP if-modified-since revalidation process for the FTP file. It does this by reissuing the FTP LIST command for the requested file, parsing the file date from the response returned by the FTP server, and comparing this date with the date that the proxy server generated for the Last-Modified header when the file was initially retrieved. If the file date has not changed, then the proxy server marks the cached FTP file as revalidated, sets a new expiration time for the file, and serves the file from the cache rather than retrieving it again from the FTP server. If the two file dates do not match, then the proxy retrieves the file from the FTP server again and caches the new copy with the new file date.

It is not always possible to obtain the directory information for the file from the FTP server. If the proxy is unable to determine the file date for the FTP file, it does not generate a Last-Modified header for the file. Instead, it uses the value specified for the CacheDefaultExpiry directive that matches the URL to determine the length of time to keep the file in the cache. When this time period expires, the proxy always retrieves the file from the FTP server again. If specific FTP files in your cache seem to be using the CacheDefaultExpiry directive very often and are frequently being retrieved (generating a high volume of network traffic), consider specifying a more granular CacheDefaultExpiry value for those specific files. Doing this holds them in the cache for a longer period of time.

To specify cache expiration settings in the Configuration and Administration forms, use the **Cache Configuration** –> **Cache Expiry Settings** –> **Time Limit for Cached Files** form. For more details on setting cached file expiration dates, see "File expiration" on page 81.

Configuring cache freshness

To specify the expiration times for cached files, in the Configuration and Administration forms, select **Cache Configuration –> Cache Expiry Settings**. The following forms are useful.

URL-based expiration

Use this form to set the minimum length of time that files are held in the cache, based on their URLs. You can specify different caching behavior for different URL request templates.

To set URL-based file expiration by editing the proxy configuration file, see the reference sections in Appendix B, "Configuration file directives," on page 161 for the following directives:

- "CacheMinHold — Specify how long to keep files available" on page 181

Default expiration settings

Use the **Cache Expiration Settings** form to specify the default expiration settings for used or unused files. You can set different values for HTTP, FTP, and Gopher files, and you can set different values for used or unused files.

This form also contains additional file-expiration options:

- **Enable cached file expiration checking**. This check box is selected by default. Generally, it is desirable to select this option so that the server does not send stale content.
- **Disable retrieval of files from remote servers**. Select this option if you do not want the server to retrieve files from remote servers.
- **Do not cache files that will expire within**. To prevent caching files that expire in a short time, specify the time period with this option. By default, files that expire within 10 minutes are not cached.

To set default expiration settings by editing the proxy configuration file, see the reference pages for the following directives:

- "CacheDefaultExpiry — Specify the default expiration time for files" on page 176
- "CacheExpiryCheck — Specify whether the server returns expired files" on page 177
- "CacheTimeMargin — Specify the minimum lifetime for caching a file" on page 183
- "CacheUnused — Specify how long to keep unused cached files" on page 183
- "CacheNoConnect — Specify the stand-alone cache mode" on page 181

Last Modified Factor settings

Use the **Last Modified Factor** form to set the value that the proxy uses to calculate an expiration date for cached files with no expiration dates in their headers. You can set different values for files matching different request templates. The first matching template is used to calculate the expiration date.

To set the Last Modified factor by directly editing the proxy configuration file, see "CacheLastModifiedFactor — Specify the value for determining expiration dates" on page 178.

Cache time limit

Use the **Time Limit for Cached Files** configuration form to set the maximum time that a file can remain in the cache. Time limits are set based on request templates,

and you can specify that files are discarded or revalidated when the time limit expires. These settings can be used to maintain files whose expiration dates are invalid or files with extremely long expiration times.

To set the maximum expiration time limit for cached files by editing the proxy configuration file, see the following:

- "CacheMaxExpiry — Specify the maximum lifetime for cached files" on page 180
- "CacheClean — Specify how long to keep cached files" on page 176

Garbage collection

As part of the effort to keep popular URLs cached and minimize usage of system resources, the Caching Proxy performs the cleanup process known as *garbage collection*, in which old or unused files are removed from the cache to make room for more-current files.

The garbage collection process examines the files in the cache directory and attempts to eliminate expired files to reduce the size of the cache and make room for new files. Garbage collection is done automatically, but some settings can be configured to tailor the process to your needs.

Configuring garbage collection

To configure garbage collection, in the Configuration and Administration forms, select **Cache Configuration –> Garbage Collection Settings**. Use this form to set the *high water mark* and *low water mark*, which determine when garbage collection is started and stopped. When the amount of space used in the cache reaches or exceeds the percentage set for the high water mark, garbage collection begins. Garbage collection continues until the percentage of used space in the cache is equal to or less than the value set for the low water mark.

You can choose between two garbage-collection algorithms. The **responsetime** algorithm optimizes the time required to respond to users by preferentially removing large files from the cache. The **bandwidth** algorithm optimizes the use of network bandwidth by preferentially removing smaller files from the cache. Choose either, or a blend of the two.

To configure garbage collection by editing the proxy configuration file, see the reference sections for the following directives:

- "Gc — Specify garbage collection" on page 203
- "GcHighWater — Specify when garbage collection begins" on page 203
- "GcLowWater — Specify when garbage collection ends" on page 204
- "CacheAlgorithm — Specify the cache algorithm" on page 175

Chapter 20. Configuring the cache agent for automatic refreshing and preloading

Most caching proxy servers cache a file only after a user requests it. The Caching Proxy has a cache agent that provides automatic cache preloading. You can specify that the cache agent automatically retrieves specified URLs, the most popular URLs, or both, and places them in the cache before they are requested.

In some cases, you need to set the host name of the proxy server and identify the cache access log before the cache is preloaded. To configure the cache agent, in the Configuration and Administration forms, select **Cache Configuration** and use the **Cache Preload** and **Cache Refresh** forms. Note that files representing query results (that is, files whose URLs include the question mark character (?) are cached only if query caching is enabled).

Automatic cache refreshing and preloading provides the following advantages:

- Caching is applied to specified URLs before a user requests the pages.
- The cache is populated before the server becomes busy with user activity.
- Current files are supplied to users more quickly from the cache than if they were fetched on the first request.

Disadvantages include the following:

- The proxy server is busy caching pages even during hours of low user activity.
- You must exercise some control over what is automatically loaded. Loading linked files from high-level pages, such as Web indexes and search sites, can generate requests for a large number of pages.

For optimal efficiency, set the cache agent to run when server activity is low and before the server becomes busy with client requests. Then the files are ready in the cache to provide fast service the first time a user requests them. By default, the cache agent is started every night at 3 a.m. local time.

Note: When the cache agent is run to refresh the cache, you must uncomment the ″Proxy http:*″ line in the ibmproxy.conf file. Otherwise, a ″403 Forbidden By Rule Error″ in the error log results and refreshing the cache does not complete.

Setting the server host name

On Linux and UNIX platforms, specify the host name of the proxy server whose cache is being preloaded or refreshed. On Windows platforms, specify the host name only if the proxy server being refreshed is not on the local machine (Note that refreshing a remote server's cache based on its most frequently accessed files is not possible because the local cache agent does not have access to a remote server's cache access log.)

To set the host name of the proxy server, in the Configuration and Administration forms, select **Cache Configuration –> Cache Refresh: Identify cache destination server**.

Preloading the cache with specific files

To preload the cache with the content stored at specific URLs, in the Configuration and Administration forms, use **Cache Configuration –> Cache Preload**. In this form, you can specify URLs for the cache agent to load. The proxy retrieves those pages when the cache agent starts, regardless of whether they were in the cache previously (These URLs are specified in the proxy configuration file by the LoadURL directive). This form can also be used to define URLs whose content is never cached. Access to a cache access log is not required for this type of cache preloading.

Use the **Cache Preload** form to configure the following options:

- **Refresh the cache daily**—Check this box if you want the cache agent to refresh the cache every night. If you do not want to start the cache agent, make sure this box is not checked.
- **Cache refresh time**—If you want the cache agent to run at a time other than 3:00 a.m. local time, specify when you want it to start.
- **Cache Contents**—In the **URL or IP Address** field, specify the URLs to load. To exclude URLs from being preloaded, specify the URLs and click **Ignore** in the **Cache status** box.

Preloading the cache with frequently cached files

To preload the most frequently accessed pages automatically, use the **Cache Configuration –> Cache Refresh** form. This function requires a Cache Access Log for the proxy server. (The log's location and name can be changed; refer to Part 6, "Monitoring the Caching Proxy," on page 135 for information.) The most popular URLs are determined automatically from the Cache Access Log. The administrator can also specify the number of popular pages to preload in the cache. (This number is specified in the proxy configuration file by the LoadTopCached directive.)

Use the **Cache Refresh** form to configure the following options:

- **Refresh the cache daily**—Check this box if you want the cache agent to refresh the cache every night. If you do not want to start the cache agent, make sure this box is cleared.
- **Cache refresh time**—If you want the cache agent to run at a time other than 3:00 a.m., specify the hour and minute when you want it to start.
- **Identify cache destination server**—Use this option if you want to refresh a server other than the local machine. (Note that you cannot refresh a remote server based on the frequency of access to specific files.)
- **Cache the most popular URLs**—Specify the number of URLs to cache from the previous night's cache access log.
- **Load linked pages**—Use this setting to configure delving (see the following section for details on delving). Set the number of levels to delve, and whether to delve for all pages (**always**), no pages (**never**), administrator-specified pages only (**admin**), or popular pages only (**topn**). Also specify whether to delve across hosts, whether to delay between requests, and whether to cache inline images.
- **Number of threads**—Set the maximum number of threads to use for cache refreshing.
- **Maximum work queue depth**—Set the maximum queue for URLs to request.
- **Maximum URLs to request**—Set the maximum number of pages to load. This number is checked before delving page retrieval begins.

- **Maximum time**—Set the maximum time to run the cache agent. If this time is set to 0 hours 0 minutes, the cache agent runs to completion.

Delving

Delving is an optional part of the automatic cache refresh feature. Most Web pages have links to other pages with related information, and users often follow the path linking from one page to another and from one site to another. Delving is a way to cache these logical information paths. In delving, the cache agent follows a specified level of hypertext (HTML) links on the pages it is loading, and also caches all of those linked pages. The linked pages can reside on the same host as the source page or on other hosts. An illustration is shown in Figure 1.

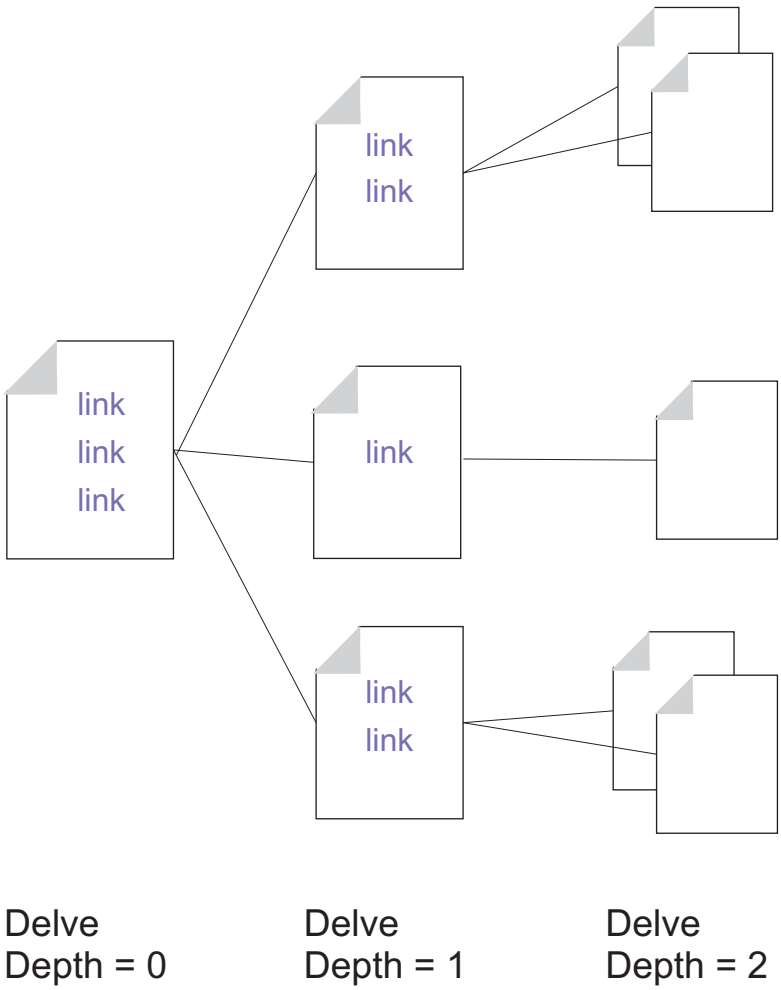

Figure 1. Delving

To control the delving process, the administrator specifies to the cache agent a maximum number of URLs that it can load (the default setting is 2000), a maximum length of time it can run (the default setting is two hours), and a maximum number of threads it can use (the default setting is four). The administrator can also configure additional controls. By default, delving is enabled for two levels of hierarchy and is not allowed across hosts. Additionally, a delay is inserted between requests. To change these settings, see "Related proxy configuration file directives" on page 90.

The cache agent loads and then refreshes the cache in this order:

1. It loads specific pages that the administrator has specified.
2. It loads popular (frequently accessed) pages from the cache access log.
3. If the maximum number of pages is not reached at this point, additional pages are loaded by delving.

Note that the cache agent does not check whether the maximum number of pages has been reached until it starts delving across links. If the value for the maximum number of pages (called MaxURLs in the proxy configuration file) is lower than the number of pages retrieved in steps 1 and 2, no linked pages are retrieved.

The following examples show how the cache agent handles cache refresh priorities and delving, relative to the maximum number of URLs that are specified (assume that delving is configured for all of these examples).

Configuration file setting	Result

`LoadURL`
` http://www.getthis.com/main.html`
`LoadURL`
` http://www.getmetoo.com/welcome.htm`
`LoadTopCached 30`
`MaxURLs 50`
	If the Cache Access Log has more than 30 unique URLs, the cache agent retrieves main.html, welcome.htm, and the top 30 requested URLs based on the cache access log. Because it has not reached the MaxURLs value, it retrieves and loads up to 18 linked URLs from pages already cached.

`LoadURL`
` http://ww.joesmith.edu/favorites.html`
`LoadURL`
` http://www.janesmith.edu/dislikes.html`
`LoadTopCached 30`
`MaxURLs 25`
	If the cache access log has more than 30 unique URLs, the cache agent retrieves favorites.html, dislikes.html, and the top 30 requested URLs from the cache access log. No other files are retrieved because the value in MaxURLs has been exceeded.

`LoadURL http://www.hello.com/hi.htm`
`LoadURL`
` http://www.ballyhoo.com/index.html`
`LoadTopCached 20`
`MaxURLs 25`
	If the cache access log has more than 20 unique URLs, the cache agent retrieves hi.htm, index.html, the top 20 requested URLs from the cache access log, and up to 3 linked URLs from the earlier pages. No other files are retrieved because the value in MaxURLs has been reached.

Related proxy configuration file directives

The cache agent can also be configured by directly editing the appropriate directives in the proxy configuration file. For proxy configuration file directives relating to the cache agent, see the following reference pages in Appendix B, "Configuration file directives," on page 161:

- "AutoCacheRefresh — Specify whether cache refreshing is to be used" on page 174
- "CacheAccessLog — Specify the path for the cache access log files" on page 175
- "CacheRefreshTime — Specify when to start the cache agent" on page 183
- "DelayPeriod — Specify pausing between requests" on page 190
- "DelveAcrossHosts — Specify caching across domains" on page 190
- "DelveDepth — Specify how far to follow links while caching" on page 190
- "DelveInto — Specify whether the cache agent follows links" on page 190
- "IgnoreURL — Specify URLs that are not refreshed" on page 208

- "LoadInlineImages — Control the refreshing of imbedded images" on page 214
- "LoadTopCached — Specify the number of popular pages to refresh" on page 214
- "LoadURL — Specify the URLs to refresh" on page 214
- "MaxUrls — Specify the maximum number of URLs to refresh" on page 220

Starting the cache agent manually

If automatic cache refreshing is enabled, the cache agent automatically runs a refresh operation at the specified time. However, you also can run the cache agent at any time from a command line.

The executable file is as follows:

- On Linux and UNIX platforms: usr/sbin/cacheagt
- On Windows platforms: *server_root*\bin\cacheagt.exe

  Where *server_root* is the drive and directory where you installed the Caching Proxy (for example, C:\Program Files\IBM\edge\cp).

On Linux and UNIX platforms, you can automatically run the cache agent at various times by using the **cron** daemon. Jobs controlled by **cron** are specified by adding a line to the system crontab file. An example entry of the command file on Linux and UNIX is:

```
45 16 * * * /usr/sbin/cacheagt
```

This command example starts the cache agent every day at 4:45 p.m. local time. You can use multiple entries to run the cache agent more than once, if desired. For more information, see your operating system's documentation about the **cron** daemon.

When using a **cron** daemon to run the cache agent, remember to turn off the automatic refresh option, either by using the **Cache Configuration –> Cache Refresh** configuration form or by editing the proxy configuration file. Otherwise, the cache agent runs more than once each day.

Chapter 21. Using a shared cache

It is common for a point of presence on the Web to have more traffic than a single server can handle. One solution is to simply add more servers. However, when multiple caching proxy servers are used, the contents of one cache often overlap with the contents of the other caches. In addition to unnecessary redundancy in storage, maximum bandwidth savings are not achieved because a cached file is re-fetched from the origin server when a request for it comes to a proxy server that does not have that file in its own cache. Although duplicate caching can be minimized by using a hierarchical chain of proxy servers, this scenario still results in additional traffic through a given server, and each additional link in the chain adds latency.

Cache sharing solves these problems by allowing each cache to share its contents with the other caches. Bandwidth savings result because of the following facts:

- Objects are not fetched multiple times.
- The larger, combined logical cache yields a higher hit rate.

Two methods are provided for using multiple caches as if they are one logical cache:

- Remote Cache Access (RCA) is a feature of the Caching Proxy that defines an array of member caches. A file is stored in exactly one of these caches based on internal logic.
- A Caching Proxy plug-in is provided to allow the proxy server to use the Internet Caching Protocol (ICP). You might use the ICP plug-in instead of RCA if you want to share data between Caching Proxy machines and non-Caching Proxy caches.

RCA and ICP can be used together.

Remote cache access

In planning for RCA, consider the following recommendations:

- The participating proxy servers need to be close to one another and connected with high-bandwidth links (for example, FDDI, SP2 bus).
- Membership in the RCA array needs to be long-term so that the configuration is as stable as possible.
- Proxy servers need to have similar capabilities (for example, CPU, memory size, cache size).
- Network outages must be infrequent.
- There need to be fewer than 100 members in any array.
- All members of the array must use the same version of the Caching Proxy software.

Note: If the proxies in the RCA array use different Linux operating systems (for example, SUSE and Red Hat), ensure that your ″nobody″ user has the same UID on all its peers. Check the password and group file entries in the /etc/ directory on each computer and assign the same UID to ″nobody.″

Remote cache access is not appropriate if any of these conditions is violated or if different organizations manage different servers that are members of the array.

Configuring remote cache access

To configure remote cache access, in the Configuration and Administration forms, select **Cache Configuration –> Remote Cache Access**. The fields in this form define a named array that shares one logical cache. Enter the required information for each member of the array.

To configure remote cache access by editing the proxy configuration file, see the reference sections in Appendix B, "Configuration file directives," on page 161 for the following directives:

- "ArrayName — Name the remote cache array" on page 172
- "Member — Specify a member of an array" on page 220

Configuring the Internet Caching Protocol plug-in

The Internet Caching Protocol plug-in enables a Caching Proxy to query ICP-compliant caches in search of HTML pages and other cacheable resources. When the proxy server receives an HTTP request, it searches its own cache for the resource. If the resource is not found in the local cache and the ICP plug-in is enabled, the proxy server encapsulates the URL request in an ICP query packet and then delivers this packet to all identified ICP peer caches. If a peer cache responds that it has the resource, the proxy server retrieves the resource from that peer's cache. If two or more peers respond positively, the first response is processed. If no peers respond with hits, the original server continues to process the request according to its workflow. For example, the proxy server can invoke another plug-in, continue to the Remote Caching Access routine (if RCA is enabled), or retrieve the requested resource itself.

Configuring the ICP plug-in

The ICP plug-in is activated and configured by editing the proxy configuration file, ibmproxy.conf. A ServerInit directive, a PreExit directive, or both must be added to the API directives section of the configuration file in order to use the ICP plug-in. Which directives are used depends on the role that the Caching Proxy has in the ICP system:

- For the Caching Proxy to function as an ICP server, use the ServerInit directive to call the icpServer module.
- For the Caching Proxy to function as an ICP client, use the PreExit directive to call the icpClient module.
- For the Caching Proxy to function as both an ICP client and and ICP server, use both directives.
- Use the directives icpAddress, icpMaxThreads, icpPeer, icpPort, and icpTimeout to configure the settings that the plug-in uses.

To create these directives, either manually edit the ibmproxy.conf file, or if the proxy server is already running, connect to the Configuration and Administration form **Server Configuration–> Request Processing –> API Request Processing**.

Note that prototype directives (in the form of comments) have been added to the API section of the ibmproxy.conf file. These API directives are in a purposeful order. When adding API directives to enable new features and plug-in modules,

order the directives as shown in the prototype section of the configuration file. Alternatively, uncomment and edit, if necessary, API directives to include support for each desired function or plug-in.

Both the ServerInit and the PreExit directives have two arguments: (1) the fully qualified path of the shared library and (2) the function call. These arguments are delimited by a colon (:). The first argument is system specific and depends on where the plug-in components are installed. The second argument is hard-coded into the shared library and must be typed exactly as shown.

Each directive must appear on a single line in the proxy configuration file.

```
ServerInit path_of_shared_library:icpServer
```

Linux and UNIX example:

```
ServerInit /opt/ibm/edge/cp/internet/lib/plugins/icp/libicp_plugin.so:icpServer
```

Windows example:

```
ServerInit C:\Program Files\IBM\edge\cp\Bin\plugins\icp\icpplugin.dll:icpServer
```

```
PreExit path_of_shared_library:icpClient
```

Linux and UNIX example:

```
PreExit /opt/ibm/edge/cp/internet/lib/plugins/icp/libicp_plugin.so:icpClient
```

Windows example:

```
PreExit C:\Program Files\IBM\edge\cp\Bin\plugins\icp\icpplugin.dll:icpClient
```

To configure settings of the plug-in, add or modify the ICP* directives that are provided in the proxy configuration file. For additional information, refer to the descriptions of the following directives.

- "ICP_Address — Specify IP address for ICP queries" on page 206
- "ICP_MaxThreads — Specify maximum threads for ICP queries" on page 206
- "Occupier — Specify a member of an ICP cluster" on page 207
- "ICP_Port — Specify port number for ICP queries" on page 207
- "ICP_Timeout — Specify maximum wait time for ICP queries" on page 207
- "PreExit — Customize the PreExit step" on page 231
- "ServerInit — Customize the Server Initialization step" on page 250

Chapter 22. Caching dynamically generated content

The dynamic caching function enables the Caching Proxy to cache dynamically generated content in the form of responses from JavaServer Pages (JSP) and servlets generated by a IBM WebSphere Application Server. A Caching Proxy adapter module is used at the application server to modify the responses so that they can be cached at the proxy server in addition to being cached in the application server's dynamic cache. With this feature, dynamically generated content can be cached on the edge of the network, freeing the content host from making repeated requests to the application server when more than one client requests the same content.

Note: The dynamic caching feature does not enable the proxy server to cache results from URL queries. To cache query results, configure caching filters, which are described in Chapter 18, "Controlling what is cached," on page 77 and in the directives reference documentation on "CacheQueries — Specify cache responses to URLs containing a question mark (?)" on page 182. Query results from origin servers that are not IBM WebSphere Application Servers can be cached.

It is sometimes necessary to enable query caching in order for the dynamic caching feature to work, for example, if your servlets use URLs in the form of queries. The proxy server considers any URL that contains a question mark (?) to be a query.

The caching of dynamically generated content offers the following benefits:

- It reduces the load on the content hosts.
- It reduces the load on the application servers.
- It speeds the delivery of requested resources to end users.
- It reduces bandwidth usage between servers.
- It improves scalability for Web sites that create or serve dynamically generated content.

The application server exports only fully composed public pages for proxy caching. Private pages are not cached by the proxy. For example, a dynamically generated page from a public site that lists the current weather forecast can be exported by the IBM WebSphere Application Server and cached by the Caching Proxy. However, a dynamically generated page that lists the contents of a user's shopping cart cannot be cached by the proxy server. Also, in order for a dynamically generated page to be cached, all subcomponents of that page also must be cacheable.

Cached dynamic files do not expire in the same way that regular files do; they can be invalidated by the application server that generated them.

Entries in the dynamic cache are invalidated in the following circumstances:

- The dynamic cache garbage collector removes an entry because of cache congestion.
- The time-out set in the servlet entry (servletcache.xml) or in the proxy's ExernalCacheManager directive expires.

- An external agent or application invokes the dynamic cache APIs to invalidate cache entries.

Invalidation of dynamic cache entries is done by generating an Invalidate message for the specific instance of the Caching Proxy dynamic caching plug-in. The Caching Proxy receives Invalidate messages as posts to the /WES_External_Adapter resource locator. The Caching Proxy then clears the invalid entries from its cache.

Dynamic caching requires the following configuration steps.

- IBM WebSphere Application Server configuration:
  - Configure each application server to do local dynamic caching.
  - Configure each application server to use the External Cache Adapter.
  - Specify which external caches can be used for each cacheable servlet and JSP file.
- Caching Proxy configuration:
  - Enable the Caching Proxy to use the dynamic caching plug-in.
  - Specify the sources from which dynamic content will be cached.

Configuring IBM WebSphere Application Server for proxy caching

Configure dynamic caching at the application server

Follow the instructions in the IBM WebSphere Application Server documentation for configuring your application server to use its local dynamic cache (also called the dynamic fragment cache). The dynamic fragment cache interacts with the external cache at the Application Server Caching Proxy.

Configure the application server adapter

The IBM WebSphere Application Server communicates with the Caching Proxy by using a software module called the External Cache Adapter, which is installed with the Application Server.

Note: Refer to the IBM WebSphere Application Server support Web site for a TechNote about configuring dynamic caching.

Configuring the Caching Proxy for dynamic caching

To enable the proxy server to cache dynamically generated content (results from servlets and JSPs), you must make two changes in the proxy configuration file, ibmproxy.conf. The first change enables the dynamic caching plug-in module, and the second change configures it to recognize the sources of cacheable dynamic content.

Set the Service directive to enable the dynamic caching plug-in

An API directive for the Service step is used to enable the dynamic caching plug-in. To create this directive, either manually edit the ibmproxy.conf file, or if the proxy server is already running, use the Configuration and Administration forms to select **Server Configuration –> Request Processing –> API Request Processing**. The content of the directive is shown in examples that appear later in this section.

A prototype Service directive for enabling dynamic caching exists as a comment in the API section of the ibmproxy.conf file. It has the heading `JSP Plug-in`. Note that the prototype API directives are in a purposeful order. When adding API directives to enable new features and plug-in modules, order the directives as shown in the prototype section of the configuration file. Optionally, you can remove the comment characters from the prototype API directives and edit them as necessary to include support for each desired function or plug-in.

Set the Service directive as shown in the following examples. (Note that each directive must appear on a single line in the proxy configuration file; these examples sometimes contain line breaks for readability.)

- For AIX:

  ```
  Service  /WES_External_Adapter  /opt/ibm/edge/cp/lib/plugins/
  dynacache/libdyna_plugin.o:exec_dynacmd
  ```

- For Solaris:

  ```
  Service  /WES_External_Adapter  /opt/ibm/edge/cp/lib/plugins/
  dynacache/libdyna_plugin.so:exec_dynacmd
  ```

- For Linux:

  ```
  Service  /WES_External_Adapter  /usr/lib/libdyna_plugin.so:exec_dynacmd
  ```

- For Windows:

  ```
  Service  /WES_External_Adapter  C:\Program Files\IBM\edge\cp\bin\plugins\
  dynacache\dyna_plugin.dll:exec_dynacmd
  ```

If the Caching Proxy software is installed in a directory other than the default, substitute your installation path for the path in these examples.

Set the ExternalCacheManager directive to specify file sources

Each Caching Proxy must also be configured to recognize the source of the dynamically generated files. Add an ExternalCacheManager directive to the ibmproxy.conf file for each application server that caches dynamically generated content at this proxy server. This directive specifies a WebSphere Application Server that caches results at the proxy, and sets a maximum expiration time for content from that server. More details appear in “ExternalCacheManager — Configure the Caching Proxy for dynamic caching from IBM WebSphere Application Server” on page 200.

The server ID used in the ExternalCacheManager directive must match the group ID used in the external cache group stanza of the application server’s dynacache.xml file.

For the previous example, add the following entry to each proxy’s ibmproxy.conf file.

```
ExternalCacheManager   IBM-edge-cp-XYZ-1  20 seconds
```

The Caching Proxy caches only contents from a IBM WebSphere Application Server whose group ID matches an ExternalCacheManager entry in the ibmproxy.conf file.

Chapter 23. Tuning the proxy server cache

When caching is enabled, the speed of cache storage devices is critical to Caching Proxy performance. This section gives suggestions on choosing a type of cache storage and configuring your cache storage devices for best performance.

Choosing the cache storage media

The Caching Proxy can use two different types of cache storage media:

- Memory
- Raw disk partitions

A memory cache provides the fastest file retrieval, but the size of a memory cache is limited by the amount of available memory on the proxy server machine. A disk cache, made up of one or more raw disk partitions, is slower than a memory cache but allows larger cache sizes in most cases.

Optimizing disk cache performance

The device partitions used for disk caching need to be dedicated to the cache; that is, do not use these physical disks to contain any other file systems, and do not use them for any purpose other than storing the proxy cache. Additionally, do not use data compression on any disk used for the proxy cache because it reduces performance.

Each cache storage device (whether a disk or a file) incurs memory overhead on the proxy server. In general, using an entire physical disk as a single cache device yields the best performance. Using RAID or other mechanisms to combine multiple physical disks into a single logical disk can be counterproductive. If you want to use multiple disks, specify them as multiple cache devices by using the **Cache Settings** configuration form or by editing the CacheDev directive in the proxy configuration file. This method allows the proxy server to control the parallelism of reading and writing to multiple disks, and does not rely on the performance of the operating system or a disk subsystem.

Cache garbage collection

Cache garbage collection for the proxy server discards expired files from the cache, freeing space to cache files for new requests. Garbage collection is triggered automatically when the amount of used space in the cache reaches an administrator-specified limit called the *high water mark* and continues until the amount of used space reaches a *low water mark*.

Because the garbage collection routine uses minimal CPU resources and does not affect the availability of unexpired cached material, configuring garbage collection to run at specific times is not necessary.

To improve the performance of garbage collection, you can set the high water mark and low water mark. You can also configure the type of algorithm used for garbage collection. See “Garbage collection” on page 85 for more information on modifying garbage collection.

Platform-specific optimizations

The following are additional suggestions for optimizing cache performance on each platform.

AIX

Create a single logical volume on a disk, preferably using all the physical partitions (PPs) available. For example, given a 9-GB disk, create a 9-GB logical volume called cpcache1. Format it and specify it as a proxy cache device using its raw logical volume, /dev/rcpcache1.

HP-UX and Solaris

On your cache device, create a single partition (or slice) that uses the entire size of the disk. For example, on a 9-GB disk, create a 9-GB partition called c1t3d0s0. Format it and specify it as a proxy cache device using its raw device, /dev/rdsk/c1t3d0s0.

Windows

Create a single partition, using the entire size of the disk. For example, on a 9-GB disk, create a 9-GB partition called i:. Format it and specify it as a proxy cache device using its raw device, \\.\i:.

Information on configuring the proxy server's cache and on formatting and specifying cache devices is included in Part 4, "Configuring proxy server caching," on page 67.

Part 5. Configuring Caching Proxy security

This part provides information about basic security, using SSL with the Caching Proxy, enabling cryptographic hardware, and using the IBM Tivoli® Access Manager (formerly Tivoli Policy Director) plug-in and the PAC-LDAP authorization module.

This part contains the following chapters:

Chapter 24. About proxy server security

Any server accessible from the Internet is at risk for attracting unwanted attention to the system on which it runs. Unauthorized people might try to guess passwords, update files, execute files, or read confidential data. Part of the attraction of the World Wide Web is its openness. However, the Web is open to both positive use and abuse.

The following sections describe how to control who has access to the files on your Caching Proxy server.

The Caching Proxy supports Secure Sockets Layer (SSL) connections, in which secure transmissions involving encryption and decryption are established between the client browser and the destination server (either a content server or a surrogate server).

When the Caching Proxy is configured as a surrogate, it can establish secure connections with clients, with content servers, or both. To enable SSL connections, in the Configuration and Administration forms, select **Proxy Configuration –> SSL Settings**. On this form select the **Enable SSL** check box and specify a key ring database and a key ring database password file.

You can take several basic precautions to protect your system:

- Place a server meant for public access in a network that is separate from your local or internal network.
- Disable utilities that allow remote users to access the server's internal processes. In particular, consider disabling **telnet**, **TN3270**, **rlogin**, and **finger** clients on the system that is running the server.
- Use packet filtering and firewalls.

  Packet filtering allows you to define where data can come from and where it can go. You can configure your system to reject certain source-destination combinations.

  A firewall separates an internal network from a publicly accessible network, such as the Internet. The firewall can be a group of computers or a single computer that acts as a gateway in both directions, regulating and tracking the traffic passing through it. IBM Firewall is an example of firewall software.
- Control CGI scripts. Using CGI scripts on a Web server can create a security risk because it is possible for these scripts to display environment variables that include such sensitive data as user IDs and passwords. Make sure that you know exactly what a CGI program does before you execute it on your server, and control who has access to CGI scripts on your server.

Note: If the Configuration Wizard is used to configure the proxy server, then to enable SSL, a mapping rule must be created to proxy requests received through port 443. For more information, refer to "Define mapping rules" on page 39.

Examples:

```
Proxy /* http://content server :443
```

or

```
Proxy /* https://content server :443
```

Chapter 25. Server protection setups

This chapter describes how to protect the data and files on your server by using protection setups. Protection setups are triggered based on the request that the server receives, specifically on the particular directory, file, or type of file that the request addresses. Within a protection setup, subdirectives control how access is granted or denied based on the characteristics of the directories or files being protected.

Using the Configuration and Administration forms to set protection

To define a protection setup and how it is applied, in the Configuration and Administration forms, select **Server Configuration –> Document Protection**. Use this form for the following steps:

1. Set the order for this protection rule.

   Protection rules are applied in the order in which they are listed in the table on the configuration form. In general, rules are listed from specific to generic.

   Use the drop-down menu and buttons to specify the placement of a protection rule.

2. Define a request template.

   Protection is activated based on request templates, which are compared to the content of requests that clients send to your proxy server.

   A *request* is the part of a full URL that follows your server host name. For example, if your server is named fine.feathers.com and a browser user enters the URL `http://fine.feathers.com/waterfowl/schedule.html`, your server receives the request /waterfowl/schedule.html. Request templates specify directory or file names, or both, that are subject to protection. For example, some requests that activate protection based on the request template just described (/waterfowl/schedule.html) include /waterfowl/* and /*schedule.html.

   Type the request template in the **URL request template** field.

3. Define a protection setup.

   A protection setup tells the Caching Proxy what to do with a request that matches a request template. You can use a named protection setup or define a new setup in the **Document Protection** form.

   To use a named setup, click the **Named protection** radio button and type the name in the field provided. To define a new setup, click the **In-line** radio button and follow the instructions provided (see Step 6).

4. Choose a requester address (optional).

   Different rules can be applied to requests from different server addresses. For example, you might want to apply a different protection setup to requests for log files when those requests are received from IP addresses assigned to your company.

   Note: For requester addresses to be screened, DNS lookup must be enabled. See “DNS-Lookup — Specify whether the server looks up client host names” on page 194.

   If you want to include the address of the requester in the rule, type it in the **Server IP address or host name** field.

5. Click **Submit**.

   If you have used a named protection setup, no further input is required. If you have selected an in-line protection setup or specified a named setup that does not exist, the system opens additional forms.

6. Set protection details.

   If you did not specify an existing named protection setup, an additional form opens, on which you can specify which users can access the documents or directories matching the request template, and which actions those users are allowed.

   - **Password Authentication Settings**—Specify the password file, group file, or both, to use for user authentication. Also specify the name that is used to identify the server when it prompts for a requester's name and password.

     Note: Some browsers cache user IDs and passwords and associate them with a server ID. Your users might find it more convenient if you always use the same server ID with the same password file.

   - **Permissions**—Specify which users or groups are authorized to read, write, or delete the protected files.

7. Click **Submit**.
8. Restart the server.

Using configuration file directives to set protection

To set protection by directly editing the Caching Proxy's configuration file, you must first understand the following issues:

- The differences between Protect, defProt, and Protection directives
  - The Protect directive sets protection by linking a request template to a protection setup. See "Protect — Activate a protection setup for requests that match a template" on page 231 for more information.
  - The defProt directive sets a default protection setup for a particular request template. See "DefProt — Specify default protection setup for requests that match a template" on page 187 for more information.
  - The Protection directive is used to define a named protection setup. See "Protection — Define a named protection setup within the configuration file" on page 235 for more information.
- How protection interacts with request routing

  Request-routing directives, like Map, Exec, Pass, and Proxy, are used to control which requests your server accepts and how it redirects requests to actual file locations. Request-routing directives use the same type of request templates as protection directives. Because the directions associated with the first matching template for each request are executed, protection directives must be listed before routing directives in the configuration file, in order for protection to work correctly. For more information, see "Protect — Activate a protection setup for requests that match a template" on page 231.
- The difference between in-line and named protection setups

  The Protect directive can be used to specify an in-line protection setup or can refer to an existing, named setup. The syntax for the two types of statements is slightly different. For information, see "Protect — Activate a protection setup for requests that match a template" on page 231.
- How to write a protection setup

A protection setup is a series of statements that use the protection subdirectives. Syntax and reference information about writing protection setups is contained in Appendix B, "Configuration file directives," on page 161; see the following reference sections:

- "Protect — Activate a protection setup for requests that match a template" on page 231
- "Protection — Define a named protection setup within the configuration file" on page 235
- "Protection subdirectives — Specify how a set of resources is protected" on page 236

Default protection settings

The default proxy configuration file includes a protection setup that requires an administrator ID and password in order to access files in the /admin-bin/ directory. This setting restricts access to the Configuration and Administration forms.

Chapter 26. Secure Sockets Layer (SSL)

Secure Sockets Layer (SSL) is a system that automatically encrypts information before sending it over the Internet and decrypts it at the other end before it is used. This protects sensitive information like credit card numbers while it is transmitted over the Internet.

The Caching Proxy uses SSL to secure surrogate servers and to provide secure remote administration, as described in the following sections. SSL can also be used to protect connections to back-end servers (for example, content or application servers) as well as to protect communications between the proxy server and its clients.

The SSL handshake

SSL protection is initiated when a secure connection request is sent from one machine to another—for example, when a browser sends a request to a surrogate proxy server. The request syntax https:// instead of http:// tells the browser to send the request on port 443, which is where the server listens for secure connection requests (instead of port 80 for routine requests). To establish a secure session between the browser and the server, the two machines perform an exchange called an *SSL handshake* to agree on a cipher specification and to select a key that is used to encrypt and decrypt information. Keys are automatically generated, and they expire when the session expires. A typical scenario (assuming SSL Version 3) is the following:

1. Client hello

   The client initiates an SSL session with the Caching Proxy by sending a Client Hello message that describes the client's encryption capabilities.

2. Server hello

   The server sends its certificate to the client and chooses the ciphersuite to use for data encryption.

3. Client finish

   The client sends cipher key information that is used to create symmetric encryption keys for the encrypted data. This key material is known as the *premaster secret* and it is encrypted with the server's public key (obtained from the server's certificate; see "Key and certificate management" on page 112). Both the server and the client can derive the read and write symmetric encryption keys from the pre-master secret.

4. Server finish

   The server sends a final confirmation and a Message Authentication Code (MAC) for the entire handshake protocol.

5. Client validation

   The client sends a message to validate the server finish message.

6. Secure data flow

   If the client validates the server finish message, then encrypted data flow begins.

Using a Caching Proxy as an end point for secure connections can reduce the load on your content or application servers. When a Caching Proxy maintains a secure connection, it performs encryption, decryption, and key creation, which are all

CPU-intensive operations. The Caching Proxy also allows you to configure SSL session timeouts to maximize the use of each key.

Limitations of SSL

The following limitations apply to SSL in WebSphere Application Server's Caching Proxy:

- The Caching Proxy itself cannot be used as a certificate authority (see "Key and certificate management").
- Some browsers might not support all of the encryption technology used in the Caching Proxy.

Configuring secure remote administration

Remote administration of your Caching Proxy can be achieved by using the security features provided by Secure Sockets Layer (SSL) and password authentication. This significantly reduces the probability of access to the proxy server by unauthorized persons.

To apply SSL during remote administration of your server, use an https:// request instead of an http:// request to open the Configuration and Administration forms. For example:

`https://your.server.name/yourFrontPage.html`

Key and certificate management

As noted previously, before configuring SSL you must set up a key database and obtain or create a certificate. Certificates are used to authenticate server identities. Use the IBM Key Management utility (sometimes called iKeyman) to set up your certification files. This utility is part of the GSKit software, which is included with Application Server. GSKit also includes a Java-based graphical interface for opening certificate files.

The following are the basic steps to set up your SSL keys and certificates.

1. Ensure that GSKit is installed. On most platforms, it is installed automatically with the Caching Proxy component. The name of the package is gsk7ikm (gsk7ikm_gcc295 on Linux systems for i386). The GSKit is usually installed in the ibm/gsk7/ directory (ibm/gskit/ on AIX systems). On Windows platforms, it can also be accessed from the **Start** menu.

   Note: On Windows, if GSKit does not install when using InstallShield, check to make sure the path to the install media directory does not contain a blank space.
2. Use the key manager to create a key for secure network communications and receive a certificate from a certificate authority. You might decide to create a self-signed certificate while waiting to receive the certificate from the authority.
3. Create a key database and specify a key database password.

Note: The key and keystash files are uninstalled whenever Caching Proxy is uninstalled. To avoid having to request a new certificate from a certificate authority, save backup copies of these two files in another directory before uninstalling the proxy software.

On all operating systems except for Linux, if the certificate has expired, Caching Proxy will not start properly, and an error message will display indicating the key database has expired. On Linux, the proxy appears to start but the process quickly disappears and no error message gets generated.

Certificate authorities

Your public key must be associated with a digitally signed certificate from a certificate authority (CA) that is designated as a trusted root CA on your server. You can buy a signed certificate by submitting a certificate request to a certificate authority (CA) provider. The Caching Proxy supports the following external CAs:

- VeriSign
- Thawte

By default, the following are designated as trusted CAs:

- Verisign Class 1 Individual Subscriber CA - Persona Not Validated
- Verisign Class 2 Individual Subscriber CA - Persona Not Validated
- Verisign Class 3 Individual Subscriber CA - Persona Not Validated
- VeriSign Class 3 International Server CA
- VeriSign Class 2 OnSite Individual CA
- VeriSign Class 1 Public Primary CA
- VeriSign Class 2 Public Primary CA
- VeriSign Class 3 Public Primary CA
- VeriSign Class 1 Public Primary CA - G2
- VeriSign Class 2 Public Primary CA - G2
- RSA Secure Server CA (from VeriSign)
- Thawte Personal Basic CA
- Thawte Personal Freemail CA
- Thawte Personal Premium CA
- Thawte Premium Server CA
- Thawte Server CA

Using the IBM Key Manager utility

This section provides a quick reference for using the IBM Key Manager utility (iKeyman). Use the key manager to create your SSL key database file, public-private key pair, and certificate request. After you receive the CA-signed certificate, use the key manager to place the certificate in the key database where you created the original certificate request.

More detailed documentation on the IBM Key Manager and GSKit is packaged with the GSKit software.

Set up your system to run the key manager

Before starting the IKeyman GUI, do the following:

1. Install IBM or an IBM-equivalent 32-bit Java 2 Technology, version 1.4.2
2. Set JAVA_HOME to the Java directory location. For example:
   - Windows: `set JAVA_HOME=C:\Program Files\IBM\Java142`
   - Linux and UNIX: `export JAVA_HOME=/usr/opt/IBMJava2-142`

3. Remove the ibmjsse.jar and the gskikm.jar (if present) and ibmjcaprovider.jar files from your JAVA_HOME/jre/lib/ext directory.

   Note: For Sun, substitute JAVA_HOME/lib/ext/ directory for JAVA_HOME/jre/lib/ext directory.

4. All the following jar files are currently in the *GSKit_Installation_path*/classes/jre/lib/ext/.
   - Copy the specified jar files into JAVA_HOME/jre/lib/

     ```
     ibmjcefw.jar
     ibmpkcs11.jar
     ```

   - Copy the specified jar files into JAVA_HOME/jre/lib/ext

     ```
     ibmjceprovider.jar
     ibmpkcs.jar
     ```

   - Copy the specified jar files to JAVA_HOME/jre/lib/security

     ```
     local_policy.jar
     US_export_policy.jar
     ```

5. Register IBM JCE, IBM CMS, and/or IBMJCEFIPS service providers:

   Update the JAVA_HOME/jre/lib/security/java.security file to add both IBM CMS and IBM JCE providers after the Sun provider. For example:

   ```
   security.provider.1=sun.security.provider.Sun
   security.provider.2=com.ibm.spi.IBMCMSProvider
   security.provider.3=com.ibm.crypto.provider.IBMJCE
   ```

   A sample java.security file can be found in *GSKit_Installation_path*/classes/gsk_java.security.

   - To enable FIPS operation, update the JAVA_HOME/jre/lib/security/java.security file to also add IBMJCEFIPS after the Sun provider. Make sure the IBMJCEFIPS provider was registered at a higher priority than IBMJCE. For example:

     ```
     security.provider.1=sun.security.provider.Sun
     security.provider.2=com.ibm.spi.IBMCMSProvider
     security.provider.3=com.ibm.crypto.fips.provider.IBMJCEFIPS
     security.provider.4=com.ibm.crypto.provider.IBMJCE
     ```

6. (Optional) If you are a JSSE user and use JSSE to access crypto hardware, install the ibmpkcs11.jar in the JAVA_HOME/jre/lib directory and follow the instructions in *GSKit_Installation_path*/classes/native/native-support.zip to setup the crypto hardware shared libraries.

   Note: You could also find ibmpkcs11.jar in the JSSE package released after August 5, 2002. To register an IBMPKCS11 service provider, an example that updates the JAVA_HOME/jre/lib/security/java.security file is the following:

   ```
   security.provider.1=sun.security.provider.Sun
   security.provider.2=com.ibm.crypto.provider.IBMJCE
   security.provider.3=com.ibm.crypto.pkcs11.provider.IBMPKCS11
   ```

Starting the key manager

Start the key manager graphical user interface as follows:

- On Linux and UNIX platforms, enter gsk7ikm at a command prompt.
- On Windows platforms, click **Start –> Programs –> IBM WebSphere –> Edge Components –> Caching Proxy –> Start Key Management Utility.**

Note that if you create a new key database file during this session, the file is stored in the directory from which you started the key manager.

Creating a new key database, password, and stash file

A key database is a file that the server uses to store one or more key pairs and certificates. You can use one key database for all your key pairs and certificates, or create multiple databases. The key management utility is used to create new key databases and specify their passwords and stash files.

To create a key database and stash file:

1. Start the key management utility.
2. From the main menu, select **Key Database File –> New**.
3. In the **New** dialog box, make sure that the file type **CMS Key Database** is selected. Type your key database name and file location or accept the default, **key.kdb**. Click **OK**.
4. In the **Password Prompt** dialog box, type and confirm the password for this database. Click **OK**.
5. Select the check box to stash the password file. When prompted, type and confirm a password for verification. The following message is displayed:
   `DB-Type: CMS key database file` *`keyfile_database_name`*

   Note: If you do not stash the password file, the server starts but does not listen on port 443.

The password that you specify when you create a new key database protects the private key. The private key is the only key that can sign documents or decrypt messages encrypted with the public key.

Use the following guidelines when specifying the password:

- The password must be composed from the U.S. English character set.
- The password must be at least six characters in length and contain at least two nonconsecutive numbers. Make sure that the password does not consist of publicly obtainable information about you, such as your name or your immediate family's names, initials, or birth dates.
- Stash the password.

It is a good practice to change the key database password frequently. However, if you specify an expiration date for the password, keep a record of when to change it. If the password expires before you change it, a message is written to the error log and the server starts, but it cannot make secure network connections.

Follow these steps to change the key database password:

1. From the main menu, click **Key Database File –> Open**.
2. In the **Open** dialog box, type your key database name or accept the default, **key.kdb**. Click **OK**.
3. In the **Password Prompt** dialog box, type your established password and click **OK**.
4. From the main menu, click **Key Database File –> Change Password**.
5. In the **Change Password** dialog box, type and confirm a new password. Click **OK**.

For an SSL connection between a proxy and an LDAP server, put the key database password in the pac_keyring.pwd file. (Note that the pac_keyring.pwd file is not the stash file generated by IKeyMan.)

Creating a new key pair and certificate request

The key database stores key pairs and certificate requests. To create a public-private key pair and certificate request, follow these steps:

1. If you have not created the key database, follow the instructions in "Creating a new key database, password, and stash file" on page 115.
2. In the key management utility, from the main menu, click **Key Database –> File –> Open**.
3. In the **Open** dialog box, type your key database name (or click **key.kdb** if you are using the default). Click **OK**.
4. In the **Password Prompt** dialog box, type your password and click **OK**.
5. From the main menu, click **Create –> New Certificate Request**.
6. In the **New Key and Certificate Request** dialog box, specify the following:
   - **Key Label:** Type a name (label) that is used to identify the key and certificate in the database: for example, `my self-signed certificate` or `www.companyA.com`.
   - **Keysize:** Size of the key, for example, `1024`. (In order to take advantage of 128 bit encryption, a Keysize of 1024 is recommended.)
   - **Organization Name:** Name of the organization to associate with the key, for example, `Company A`.
   - **Organization Unit** (Optional)
   - **Locality** (Optional)
   - **State/Province** (Optional)
   - **Zipcode** (Optional)
   - **Country:** Your country code. You must specify at least two characters, for example, `US`.
   - **Certificate request file name:** A name for the request file. Optionally, a default name can be used.
7. Click **OK**. A confirmation message is displayed:

   ```
   A new certificate request has been successfully created
   in the file keyfile_database_name.
   ```

8. Click **OK**. Expect the label name that you entered to be displayed under the **Personal Certificate Requests** heading.
9. In the **Information** dialog box, click **OK**. You are reminded to send the file to a certificate authority.
10. Unless you have created a self-signed certificate (see the following section, "Creating a self-signed certificate," for details), send the certificate request to a CA:
    - Leave the key manager running.
    - Start a Web browser and enter the URL of the CA from which you want to obtain the certificate.
    - Follow the instructions provided by the CA to send your certificate request.

    Certificate requests can take two to three weeks to be fulfilled. While you are waiting for the CA to process your certificate request, you can act as your own CA and use iKeyman to create a self-signed server certificate to enable SSL sessions between clients and your Caching Proxy server.

Creating a self-signed certificate

Use the key management utility to create a self-signed server certificate to enable SSL sessions between clients and your proxy server while waiting for a certificate to be issued. You also can use self-signed certificates for testing purposes.

Follow this procedure to create a self-signed certificate:

1. If you have not created the key database, follow the instructions in "Creating a new key database, password, and stash file" on page 115.
2. In the key management utility, from the main menu, click **Key Database –> File –> Open**.
3. In the **Open** dialog box, type your key database name (or accept the default, **key.kdb**). Click **OK**.
4. In the **Password Prompt** dialog box, type your password and click **OK**.
5. In the **Key Database** content frame, select **Personal Certificates** and click **Create New Self-Signed Certificate**.
6. In the **Create New Self-Signed Certificate** window, specify the following:
   - **Key Label:** A name (label) that is used to identify the key and certificate in the database: for example, `my self-signed certificate`
   - **Key Size:** Size of the key, for example, `512`.
   - **Common Name:** The full host name of the server, for example, `www.myserver.com`
   - **Organization Name:** Name of the organization to associate with the key, for example `Company A`
   - **Organization Unit** (Optional)
   - **Locality** (Optional)
   - **State/Province** (Optional)
   - **Zipcode** (Optional)
   - **Country:** Your country code. You must specify at least two characters, for example, US.
   - **Validity Period:** The period of time that the certificate is valid.
7. Click **OK**.
8. Register the key database with the server by adding the key file and stash file to the configuration settings (see "Creating a new key database, password, and stash file" on page 115).

Exporting keys

Use this procedure to export keys to another key database:

1. Start the key management utility.
2. From the main menu, click **Key Database File –> Open**.
3. In the **Open** dialog box, type your key database name (or accept the default, **key.kdb**). Click **OK**.
4. In the **Password Prompt** dialog box, type your password and click **OK**.
5. In the **Key Database** content frame, select **Personal Certificates**, then click the **Export/Import** button on the label.
6. In the **Export/Import Key** window:
   - Select **Export Key**.
   - Select the target database type (for example, **PKCS12**).
   - Type the file name or click **Browse** to select it.
   - Type the correct location.

7. Click **OK**.
8. In the **Password Prompt** dialog box, type the correct password, type the password again to confirm, then click **OK** to export the selected key to another key database.

Importing keys

To import keys from another key database:

1. Start the key management utility.
2. From the main menu, select **Key Database File –> Open**.
3. In the **Open** dialog box, type your key database name (or accept the default, **key.kdb**). Click **OK**.
4. In the **Password Prompt** dialog box, type your correct password and click **OK**.
5. In the **Key Database** content frame, select **Personal Certificates**, then click the **Export/Import** button on the label.
6. In the **Export/Import Key** window:
   - Select **Import Key**.
   - Select the key database file type (for example, **PKCS12**).
   - Type the file name or click **Browse** to select it.
   - Select the correct location.
7. Click **OK**.
8. In the **Password Prompt** dialog box, type the correct password and click **OK**.
9. In the **Select from Key Label** list, select the correct label name and click **OK**.

Listing certificate authorities

To display a list of trusted certificate authorities (CAs) in a key database:

1. Start the key management utility.
2. From the main menu, click **Key Database File –> Open**.
3. In the **Open** dialog box, type your key database name (or accept the default, **key.kdb**). Click **OK**.
4. In the **Password Prompt** dialog box, type your correct password and click **OK**.
5. In the **Key Database** content frame, select **Signer Certificates**.
6. Click **Signer Certificates**, **Personal Certificates**, or **Certificate Requests** to view the list of CAs in the **Key Information** window.

Receiving a CA certificate

Use this procedure to receive a certificate that is electronically mailed to you from a certificate authority (CA) that is designated as a trusted CA by default (see the list in "Certificate authorities" on page 113). If the CA that issues your CA-signed certificate is not a trusted CA in the key database, you must first store the CA's certificate and designate the CA as a trusted CA. Then you can receive your CA-signed certificate into the database. You cannot receive a CA-signed certificate from a CA that is not a trusted CA (see "Storing a CA certificate" on page 119).

To receive a CA-signed certificate into a key database:

1. Start the key management utility.
2. From the main menu, select **Key Database File –> Open**.
3. In the **Open** dialog box, type your key database name (or accept the default, **key.kdb**). Click **OK**.

4. In the **Password Prompt** dialog box, type your password and click **OK**.
5. Ensure that the file name in the **DB-Type** listing is correct.
6. In the **Key Database** window, select **Personal Certificates**, then click **Receive**.
7. In the **Receive Certificate from a File** dialog box, type the name of a valid base 64-encoded file in the **Certificate filename** text field. Click **OK**.
8. To close the key manager utility, from the main menu, click **Key Database File –> Exit**.

Storing a CA certificate

Only certificates signed by trusted CAs are accepted for establishing secure connections. To add a CA to the list of trusted authorities, you must obtain and store its certificate as trusted. Follow this procedure to store a certificate from a new CA, prior to receiving it into the database:

1. Start the key management utility.
2. From the main menu, click **Key Database File –> Open**.
3. In the **Open** dialog box, type your key database name (or accept the default, **key.kdb**). Click **OK**.
4. In the **Password Prompt** dialog box, type your password and click **OK**.
5. In the **Key Database** content frame, select **Signer Certificates**, then click **Add**.
6. In the **Add CA's Certificate from a File** dialog box, select the base 64-encoded ASCII data certificate file name, or use the **Browse** option. Click **OK**.
7. In the **Label** dialog box, type a label name and click **OK**.
8. Use the check box to designate the certificate as trusted (default).

   Note: View the check box *after* the certificate is created by using the "View/Edit" button. The check box is listed on the panel but it is not displayed during the adding of the certificate.

Displaying the default key in a key database

Display the default key entry as follows:

1. Start the key management utility.
2. From the main menu, click **Key Database File –> Open**.
3. In the **Open** dialog box, type your key database name (or accept the default, **key.kdb**). Click **OK**.
4. In the **Password Prompt** dialog box, type your password and click **OK**.
5. In the **Key Database** content frame, select **Personal Certificates** and select the CA certificate label name.
6. In the **Key Information** window, click **View/Edit** to display the certificate default key information.

Supported cipher specifications

The encryption algorithms and hashes used for SSL versions 2 and 3 are listed in the following tables.

Key Pair Generation: RSA 512–1024 private key sizes

SSL Version 2

US version	Export version

RC4 US	RC4 Export
RC2 US	RC2 Export
DES 56-bit	*not applicable*
Triple DES US	*not applicable*
RC4 Export	*not applicable*
RC2 Export	*not applicable*

SSL Version 3

US version	**Export version**
Triple DES SHA US	DES SHA Export
DES SHA Export	RC2 MD5 Export
RC2 MD5 Export	RC4 MD5 Export
RC4 SHA US	NULL SHA
RC4 MD5 US	NULL MD5
RC4 MD5 Export	NULL NULL
RC4 SHA 56-bit	*not applicable*
DES CBC SHA	*not applicable*
NULL SHA	*not applicable*
NULL MD5	*not applicable*
NULL NULL	*not applicable*

These SSL specifications can also be configured by directly editing the proxy configuration file. For details, see the reference sections in Appendix B, "Configuration file directives," on page 161 for the following directives:

- "V2CipherSpecs — List the supported cipher specifications for SSL Version 2" on page 261
- "V3CipherSpecs — List the supported cipher specifications for SSL Version 3" on page 261
- "FIPSEnable — Enable Federal Information Processing Standard (FIPS) approved ciphers for SSLV3 and TLS" on page 201

128-bit encryption for the Caching Proxy

Only a 128-bit encryption version of the Caching Proxy is being delivered. The 56-bit version is no longer available. If you are updating a previous version, you can install the Caching Proxy directly to your currently installed 128-bit or 56-bit version. If you were previously using a 56-bit (export) browser, you must upgrade to a 128-bit browser in order to take advantage of the 128-bit encryption in the proxy.

After an upgrade from a 56-bit version of the Caching Proxy to the 128-bit version, if the key size used to encrypt certificates is set to 1024, then no configuration change is necessary. However, if the key size is set to 512, in order to take advantage of the proxy's 128-bit encryption, you must create new certificates with a key size of 1024. Create new keys by using the IBM Key Manager utility (iKeyman).

1. Start the key manager.

- On Linux and UNIX platforms, enter `gsk7ikm` at a command prompt.
- On Windows systems, click **Start –> Programs –> IBM WebSphere –> Edge Components –> Start Key Management Utility**.

2. From the main menu, click **Key Database File –> Open**.
3. In the **Open** dialog box, type your key database name (or click **key.kdb** if you are using the default) and click **OK**.
4. If the **Password Prompt** dialog box opens, type your password and click **OK**.
5. From the main menu, click **Create –> New Certificate Request**.
6. In the **New Key and Certificate Request** window, specify the following:
   - **Key Label**: Type a name that is used to identify the key and certificate in the database.
   - **Keysize**: Select **1024**.
   - **Organization Name**: type the name of the organization to be associated with the key.
   - **Country**: Type your country code. You must specify at least two characters, for example, `US`.
   - **Certificate request file name**: Type a name for the request file, or optionally, use a default name.
7. Click **OK**.

See "Key and certificate management" on page 112 for a detailed discussion of the IBM Key Manager utility.

Note that this version of the product does not support encryption on SUSE Linux.

Chapter 27. Enabling the support of cryptographic hardware

Follow this procedure to enable the SSL handshake routine to be offloaded to a cryptographic hardware card:

1. Install the cryptographic hardware card according to the manufacturer's instructions.
2. Enable SSL for the caching proxy. For more information, refer to Chapter 26, "Secure Sockets Layer (SSL)," on page 111.
3. Manually edit the SSLCryptoCard directive in the ibmproxy.conf configuration file. No entry for this directive appears in the Configuration and Administration forms. For more information, refer to the SSLCryptoCard directive reference, "SSLCryptoCard — Specify the installed cryptographic card" on page 254.

Chapter 28. Using the Tivoli Access Manager plug-in

A Caching Proxy plug-in is provided with Tivoli Access Manager (formerly Tivoli Policy Director) that enables the Caching Proxy to use Access Manager for authentication and authorization. This plug-in makes it possible for an enterprise that uses Access Manager for Web access control to add Edge technology without having to duplicate work by setting up separate authorization schemes for the proxy server.

For additional information about Tivoli Access Manager, view the product Web site at http://www.ibm.com/software/tivoli/products/. For information about software and hardware requirements and about installing the Access Manager plug-in, refer to the documentation provided with Tivoli Access Manager.

Note: The Tivoli Access Manager plug-in may not be supported on Red Hat Linux. Contact Tivoli for current support information on Linux platforms.

Configuration

A setup script for the Caching Proxy is provided with the Access Manager plug-in.

Steps to take before using the configuration script

Before running the script, do the following:

- Install all necessary software.
- Ensure that the proxy server is set to use port 80 (This is the default value.)
- Configure your LDAP and Access Manager components, and make sure they are running while you configure the Access Manager plug-in.
- Make sure that you have the Access Manager administrator ID and the LDAP administrator name available. These values are required to set up the proxy server.

Using the configuration script

The set up script is named **wslconfig.sh** and it is provided in the /opt/pdweb-lite/bin/ directory. Enter the Access Manager administrator ID and the LDAP administrator name when prompted.

The configuration script automatically performs the following steps:

- Sets the Caching Proxy user ID to `root` and group ID to `other`
- Sets the noLog directive to `*`, which causes no items to be written to the Caching Proxy's Access Log
- Creates a ServerInit directive with the following information:

```
ServerInit /opt/pdweb-lite/lib/wesauth.so:WTESeal_Init
  /opt/pdweb-lite/etc/ibmwesas.conf
```

- Creates a PreExit directive with the following information:

```
PreExit /opt/pdweb-lite/lib/wesauth.so:WTESeal_PreExit
```

- Creates an Authorization directive with the following information:

```
Authorization * /opt/pdweb-lite/lib/wesauth.so:WTESeal_Authorize
```

- Creates a ServerTerm directive with the following information:

```
ServerTerm /opt/pdweb-lite/lib/wesauth.so:WTESeal_Term
```

Creates a Protect statement and Protection setup that forwards all requests to the Access Manager authentication process, as follows:

```
Protection PROXY-PROT {
        ServerId WebSEAL-Lite
        Mask All@(*)
        AuthType Basic
        }
Protect * PROXY-PROT
```

Starting the Caching Proxy and Access Manager plug-in

After configuring the proxy server and the Access Manager plug-in, use the command **wslstartwte** instead of **ibmproxy start** to start the proxy server. The **wslstartwte** command automatically loads environment variables that the Access Manager plug-in requires in order to initialize. If you do not use **wslstartwte** when starting the proxy server, error messages are displayed about the Access Manager plug-in. The corresponding stop command, **ibmproxy stop**, is still valid when the plug-in is used.

Chapter 29. Using the PAC-LDAP authorization module

Overview

The PAC-LDAP authorization module enables the Caching Proxy to access a Lightweight Directory Access Protocol (LDAP) server when performing authorization or authentication routines. The module consists of two component sets: a pair of shared libraries that add LDAP functionality to the Caching Proxy API and a Policy Authentication Control (PAC) daemon. A ServerInit directive within the ibmproxy.conf file instructs the shared library to initialize one or more PAC daemons when the caching proxy starts up. The shared libraries read a paccp.conf file to determine the number and characteristics of the PAC daemons. During initialization, the daemon reads the pac.conf file for configuration directives and the pacpolicy.conf for policy information. Then, either an Authentication directive within the ibmproxy.conf file instructs the proxy server to call the shared library whenever authentication is required, or an Authorization directive usurps the caching proxy workflow during standard HTTP request processing.

Authentication

The process of authentication determines if a supplied set of credentials – user name and password – are valid. This process includes verifying that a user is in the registry and that the supplied password matches the password stored in the registry. These are the actions performed by using the PAC-LDAP module during the authentication step.

When the PAC-LDAP authorization module is enabled for authentication, it becomes the default repository from which user IDs, passwords, and groups are retrieved. As an HTTP request passes through the caching proxy workflow, each Protect directive compares the requested URL to its request template. If a match occurs, the Protect directive invokes a protection schema, which includes the server ID, the type of authentication to use, masking rules to apply to the requesting client, and the locations of the passwords and groups files. If the passwords file is not defined, then the user ID and password are retrieved via the PAC-LDAP authorization module. Type 0, 1, 2, and 3 policies define authentication schemes. If authentication passes, the request is served; otherwise, the caching proxy returns a 401 error to the client.

Authorization

The process of authorization determines if a user has the necessary permission to access a protected resource. When the PAC-LDAP module is used, this includes applying authorization rules residing in the pacpolicy.conf file for the HTTP request.

When the PAC-LDAP authorization module is enabled for authorization, authorization rules within the pacpolicy.conf file are applied to the HTTP request. As the HTTP request passes through the caching proxy workflow, each Protect directive compares the requested URL to its request template. If a match occurs, the Protect directive invokes a protection schema. In this case, the protection schema is the authorization routine usurped by the PAC-LDAP authorization module. The Authorization directive compares the requested URL to its request template, and, if a match occurs, the PAC-LDAP authorization module is invoked.

Type 4 policies defined within the pacpolicy.conf file further refine the authentication required for various URL requests.

Lightweight Directory Access Protocol (LDAP)

LDAP provides interactive access to X.500 directories with a minimal consumption of system resources. The IANA has assigned TCP port 389 and UDP port 389 to LDAP. For more information, refer to RFC 1777, which defines LDAP.

Installation

All components of the PAC-LDAP authorization module are automatically installed when the Caching Proxy system of WebSphere Application Server, Version 6.0.2 is installed. On Linux and UNIX systems, a Caching Proxy library (./lib/) directory, a PAC-LDAP authorization module library (./lib/plugins/pac/) directory, a binary (./bin/) directory, and a configuration (./etc/) directory are created within the /opt/ibm/edge/cp/ directory. Symbolic links are then created from the /usr/lib/, /usr/sbin/, and /etc directories to these product-specific ones.

Directory structure

Linux and UNIX directory	Windows directory	Content
/opt/ibm/edge/cp/	\Program Files\IBM\edge\cp\	Caching Proxy base directory (*cp_root*)
cp_root/sbin/	\Program Files\IBM\edge\cp\Bin\	Caching Proxy binaries and scripts
/usr/sbin/		Symbolic links to *cp_root*/sbin/
cp_root/etc/	\Program Files\IBM\edge\cp\etc\	Caching Proxy configuration files
/etc/		Symbolic links to *cp_root*/etc/
cp_root/lib/	\Program Files\IBM\edge\cp\lib\ plugins\	Caching Proxy libraries
cp_root/lib/ plugins/pac/	\Program Files\IBM\edge\ cp\lib\plugins\pac\	PAC-LDAP authorization module libraries
/usr/lib/		Symbolic links to *cp_root*/lib/ and *cp_root*/lib/ plugins/pac/
cp_root/server_root/pac/data/	\Program Files\IBM\ edge\cp\server_root\pac\data\	PAC-LDAP authorization module data storage
cp_root/server_root/ pac/creds/	\Program Files\IBM\ edge\cp\server_root\pac\creds\	PAC-LDAP authorization module credentials

LDAP Plug-in files

Linux and UNIX file name	Windows file name	Description
libpacwte.so	pacwte.dll	shared library

Linux and UNIX file name	Windows file name	Description
libpacman.so	pacman.dll	shared library
pacd_restart.sh	pacd_restart.bat	PAC daemon restart script
paccp.conf, pac.conf, pacpolicy.conf	paccp.conf, pac.conf, pacpolicy.conf	configuration and policy files

Additional requirements for secure PACD-LDAP server connections

GSKit is required by LDAP client package

To enable secure sockets layer (SSL) connections between the PACD daemon and the LDAP server, you should install the GSKit package that is required by the LDAP client package. GSKit 7 is required and provided by default on the Caching Proxy machine, but it may not be the version that is required by the LDAP client on the machine. It is possible to use different GSKit versions on the same machine for different processes.

Place the GSKit key file to `$pacd_creds_dir/pac_keyring.kdb` and the password to `$pacd_creds_dir/pac_keyring.pwd`.

Note: For GSKit requirement information on the LDAP server refer to the IBM Directory Server documentation at the following Web site: http://www.ibm.com/software/tivoli/products/directory-server/

LD_PRELOAD environment variable must be set for Linux systems

On Linux systems, the LD_PRELOAD environment variable must be configured as follows in order to enable SSL connections between the PACD daemon and the LDAP server. Set the variable to the following value:

```
LD_PRELOAD=/usr/lib/libstdc++-libc6.1-1.so.2
```

The GSKit requirement referenced previously in this section also applies to Linux systems.

Editing the ibmproxy.conf file to enable the PAC-LDAP authorization module

Three directives, ServerInit, Authorization or Authentication, and ServerTerm must be added to the API directives section of the ibmproxy.conf file to initialize the PAC-LDAP authorization module. To create these directives, either manually edit the ibmproxy.conf file, or if the proxy server is already running, connect to the Configuration and Administration forms with an Internet browser and open the API Request Processing form (Click **Server Configuration** –> **Request Processing**–> **API Request Processing**). Each directive must appear on a single line in the proxy configuration file, regardless of whether or not the examples given in this section contain line breaks for clarity.

Note that prototype directives (in the form of comments) are given in the API section of the ibmproxy.conf file. These API directives are in a purposeful order. When adding API directives to enable new features and plug-in modules, order the

directives as shown in the prototype section of the configuration file. Alternatively, uncomment and edit, if necessary, API directives to include support for each desired function or plug-in.

The ServerInit directive has three arguments: (1) the fully qualified path of the shared library, (2) the function call, and (3) the fully qualified path of the paccp.conf file. The first and second arguments are delimited by a colon (:). The second and third arguments are delimited by a space. The first and third arguments are system specific and depend on where the plug-in components are installed. The second argument is hard-coded into the shared library and must be typed exactly as shown. When creating a ServerInit directive using the API Request Processing form, both the second and third arguments must be entered in the **Function Name** field. The third argument is displayed in the **IP Template** column.

The Authorization directive has three arguments: (1) a request template, (2) the fully qualified path of the shared library, and (3) the function name. HTTP requests are compared to the request template to determine whether the application function is called. The request template can include a protocol, domain, and host; can be preceded by a slash (/); and can use an asterisk (*) as a wildcard. For example, /front_page.html , http://www.ics.raleigh.ibm.com, /pub*, /*, and * are all valid. The function name is the name given to your application function within the program. It is hard coded and must be typed exactly as shown. The first two arguments are delimited by a space. The last two arguments are delimited by a colon (:).

The Authentication directive has two arguments: (1) the fully qualified path of the shared library and (2) the function name. These arguments are delimited by a colon (:). The first argument is system specific and depends on where the shared library is installed. The URL template for the first argument must start at the document root (/) when using Caching Proxy as a reverse proxy. The second argument is hard-coded into the shared library and must be typed exactly as shown.

The ServerTerm directive has two arguments: (1) the fully qualified path of the shared library and (2) the function name. These arguments are delimited by a colon (:). The first argument is system specific and depends on where the shared library is installed. The second argument is hard-coded into the shared library and must be typed exactly as shown. This directive terminates the PAC daemon when the proxy server shuts down. If the owner of the daemon is different from the owner of the proxy server, the proxy server might be unable to stop the daemon, in which case an administrator must manually stop the daemon.

```
ServerInit path_of_shared_library:pacwte_auth_init path_of_conf_policy_file
```

Linux and UNIX example:

```
ServerInit /usr/lib/libpacwte.so:pacwte_auth_init /etc/pac.conf
```

Windows example:

```
ServerInit C:\Progra ~1\IBM\edge\cp\lib\plugins\
  pac\pacwte.dll:pacwte_auth_init C:\Progra ~1\IBM\edge\cp
Authorization request-template path_of_shared_library:pacwte_auth_policy
```

Linux and UNIX example:

```
Authorization http://* /usr/lib/libpacwte.so:pacwte_auth_policy
```

Windows example:

```
Authorization http://* C:\Program Files\IBM\edge\cp\lib\plugins\
  pac\pacwte.dll:pacwte_auth_policy
Authentication BASIC path_of_shared_library:pacwte_auth_policy
```

Linux and UNIX example:

```
Authentication BASIC /usr/lib/plugins/pac/libpacwte.so:pacwte_auth_policy
```

Windows example:

```
Authentication BASIC C:\Program Files\IBM\edge\cp\lib\plugins\
  pac\pacwte.dll:pacwte_auth_policy
ServerTerm path_of_shared_library:pacwte_shutdown
```

Linux and UNIX example:

```
ServerTerm /usr/lib/libpacwte.so:pacwte_shutdown
```

Windows example:

```
ServerTerm BASIC C:\Program Files\IBM\edge\cp\lib\plugins\
  pac\bin\pacwte.dll:pacwte_shutdown
```

Editing the PAC-LDAP authorization module configuration files

The PAC-LDAP authorization module configuration and policy files must be manually edited with a text editor. A directive name is separated from its first argument by a colon (:). Multiple arguments are delimited by commas (,). Remarks are included in the configuration and policy file to assist in editing it. Key policy directives are shown below.

paccp.conf

The paccp.conf file is read by the shared libraries during the initialization of the Caching Proxy and contains the definitions ([PAC_MAN_SERVER] stanza) of each PAC daemon that will start. Each PAC daemon must have its own [PAC_MAN_SERVER] stanza.

```
[PAC_MAN_SERVER]
hostname:                    # name of PAC daemon
port:                        # port pacd is listening on

[PACWTE_PLUGIN]
hostname_check:[true|false]  # enables DNS lookup. Must have
                             # DNS lookup turned on for ibmproxy to work.
```

pac.conf

The pac.conf file specifies the LDAP server with which the PAC daemon attempts to connect.

```
[PAC_MAN_SERVER]
hostname:                    # name of PAC daemon
port:                        # port pacd is listening on
conn_type:ssl                # comment out if you do not use SSL
authentication_sequence: [primary|secondary|none]
authorization_sequence:  [primary|secondary|none]

[LDAP_SERVER]
hostname:                    # LDAP Server hostname
port:389                     # Port LDAP is listening on
ssl_port:636                 # SSL port used by the LDAP server
admin_dn:                    # User with permission to access the LDAP server
                             # specify admin_dn:NULL to enable anonymous binding
```

```
search_base:                # Portion of LDAP tree to search for policy info
                            # If not required, specify search_base:NULL
search_key:                 # ID field to search

[CACHE]
cred_cache_enabled [TRUE|FALSE] # turn credentials cache on
cred_cache_min_size:100     # minimum number of credentials to cache in pacd
cred_cache_max_size:64000   # maximum number of credentials to cache in pacd
cred_cache_expiration:86400  # when a credential expires
policy_cache_enabled:[TRUE|FALSE] # turns policy cache on/off
policy_cache_min_size:100    # min. number of policy related items to cache
policy_cache_max_size:64000  # max. number of policy related items to cache
policy_cache_expiration:86400 # when a policy related item expires
```

pacpolicy.conf

Every LDAP policy uses the following template within the configuration and policy file. Each policy must begin with the uppercase keyword POLICY in brackets.

```
[POLICY]
default_policy:[grant|deny]  # describes the default policy for users
                            # that are not described in the POLICY section
pac_client_hotname:         # the instances of Caching Proxy that are allowed
                            # to use a policy list
id:                         # the id for the LDAP entry or ip/hostname
                            # (wildcard supported, such as *.ibm.com)
grant:[true|false]          # true means to grant access, false means
                            # to deny access
type:[0|1|2|3|4]            # 0 LDAP entry that is a group,
                            # 1 LDAP entry that is not a group,
                            # 2 IP address
                            # 3 hostname
                            # 4 URL
propagate:[true|false]      # true means that the access rights (grant
                            # or deny) will be propagated to all
                            # descendants or members
stop_entry:[entry|NULL]     # Propagation of the access right stops
                            # at this entry. If the id is a group,
                            # stop_entry must be set to NULL.
                            # stop_entry may be applied to an IP
                            # address or hostname. Each stop_entry
                            # must be on its own line
exception_entry:[entry|NULL] # Assignment of the access right skips
                            # these entries, but continues through their
                            # subtrees. This may be a list of entries.
                            # exception_entry may be applied to a group,
                            # IP address, or hostname. Each
                            # exception_entry must be on its own line.
Exception_type:
Exception:
```

The wildcard (*) is supported only for the last position of an IP address or the first position of a host name in the `id` and `stop_entry` directives. Wildcards are not supported in the `exception_entry`. Wildcards are not supported for any LDAP entries in any fields.

Multiple policies are supported, and the value false always takes priority if policies conflict. In other words, a single denial in any policy blocks access. The order in which the policies are listed in the configuration and policy file is irrelevant and does not establish a priority.

For a set of policy examples, refer to the pacpolicy.conf file in the configuration files directory.

Note: Nested groups do not inherit policies from parent groups. The only policies that are enforced on a group are those policies for which the group is an explicit member.

Creating pac_ldap.cred

Create a plain text file named pac_ldap.cred in */cp_root*/server_root/pac/creds. This file contains the password corresponding to the user name in the admin_dn directive, which is in the pac.conf file.

Note: To enable anonymous binding, change the admin_dn directive in pac.conf to admin_dn:NULL and put a dummy string in the pac_ldap.cred file.

The PAC daemon encrypts the password the first time that it reads the file.

To create the file pac_ldap.cred on Linux and UNIX platforms, issue the following commands:

```
cd cp_root/server_root/pac/creds
echo "password" > pac_ldap.cred
chown nobody pac_ldap.cred
chgrp nobody pac_ldap.cred
(on SUSE Linux, use chgrp nogroup pac_ldap.cred.)
```

To create the file on a Windows platform, type the password in a text file and store the file in the server_root\pac\creds\ directory.

Starting and stopping pacd

The LDAP authorization daemon runs as the pacd process. You can restart the LDAP authorization daemon without interrupting Caching Proxy by using scripts that are provided. Run the pacd script as follows:

- On Linux and UNIX platforms:

  ```
  /usr/sbin/pacd_restart.sh pacd_user_id
  ```

- On Windows platforms:

  ```
  C:\Program Files\IBM\edge\cp\Bin\pacd_restart.bat CP_install_root
  ```

Note: It is possible for the pacd process to continue to run after the caching proxy server is shut down by using the **stopsrc -ibmproxy** command on AIX systems or the **ibmproxy -stop** command on HP-UX, Linux and Solaris systems. The pacd process can be safely shut down by using the **kill** command as follows:

```
kill -15 pacd_process_ID
```

On HP-UX: The PAC-LDAP plug-in and pacd may not load all their dependent shared libraries at runtime. Before using them, ensure that the system variables are set as follows

```
SHLIB_PATH=/usr/lib:/usr/IBMldap/lib
PATH=/usr/IBMldap/bin:$PATH
PATH=/usr/IBMldap/bin
```

/usr/IBMldap/ is the default installation path for the LDAP client on HP-UX. Please adjust PATH and SHLIB_PATH accordingly if the LDAP client is installed in a different location. *Without* setting these variables, the following errors may occur:

- After enabling the PAC-LDAP plug-in, the following message will appear in the error log

```
"Serverinit Error: server did not load functions
 from DLL module /opt/ibm/edge/cp/lib/plugins/pac/libpacwte.sl"
```

- When attempting to start /usr/sbin/pacd the following link error will display

```
"/usr/lib/dld.sl: Can't find path for shared library: libibmldap.sl
 /usr/lib/dld.sl: No such file or directory
 Abort"
```

On Linux: For SUSE Linux Enterprise Server 9, the ldd pacd may report that the libldap.so is not found. To work around this problem, create the following symbolic:

```
ln -s /usr/lib/libldap.so.19  /usr/lib/libldap.so
```

On AIX: When starting pacd with IBM Tivoli Directory Server 5.2, the PAC-LDAP module may be unable to load resulting in the following error:

```
exec(): 0509-036 Cannot load program /usr/sbin/pacd because of the following errors:
        0509-022 Cannot load module /usr/lib/libpacman.a.
        0509-150 Dependent module libldap.a could not be loaded.
        0509-022 Cannot load module libldap.a.
```

To work around this problem, create the following symbolic:

```
ln -s /usr/lib/libibmldap.a /usr/lib/libldap.a
```

Note: After configuring the Caching Proxy to use LDAP authentication, it will show the following error:

```
Could not extract a value for: Uid, return code:3
```

This error will be displayed even when the LDAP authentication is functioning correctly and can be disregarded.

Part 6. Monitoring the Caching Proxy

This part provides instructions for monitoring the Caching Proxy using logs and the Server Activity Monitor.

This part contains the following chapters:

Chapter 30. Configuring logging

To customize logging, you can use the Configuration and Administration forms or edit directives in the proxy configuration file. You can set the following options:

- Paths and file names to store log files
- Filters to include and exclude information from access log files
- Maintenance options to archive or delete logs

About logs

The Caching Proxy can create three types of access logs, in addition to an event log and an error log:

- Access logs:
  - **Access log**—Tracks local administrative requests to the Caching Proxy itself.
  - **Cache access log**—Tracks requests for objects that are in the cache.
  - **Proxy access log**—Tracks requests that are proxied from origin servers.
- **Event log**—Tracks cache information messages.
- **Error log**—Tracks error messages related to the Caching Proxy.

The Caching Proxy creates new log files each day at midnight. If the proxy is not running at midnight, new logs are created the first time it is started that day. You can specify the directory and file name prefix for each log file; each log file that is created also contains a date suffix in the form *.Mmmddyyyy* (for example, .Apr142000).

Because logs can use a large amount of space, consider storing your log files on a storage device that is separate from the operating system and cache, to prevent errors. In addition, configure log maintenance routines, as specified in "Maintaining and archiving logs" on page 140.

Log file names and basic options

To specify the basic configuration of the proxy server logs, in the Configuration and Administration forms, select **Server Configuration –> Logging –> Log Files**. Specify the path and file name for each log file that you want to use. The current file name for each log is displayed in the corresponding text box; if you have not specified a path, the default path is displayed.

Information that is logged into the proxy logs is not automatically written to the system log, but you can configure the Caching Proxy to write to the system log in addition to or instead of to its own logs. On the **Log Files** form, select the **Log information to Syslog** check box. Note that the system log must be created before this option is chosen.

To specify that proxy server log information is written only to the system log, you must edit the proxy configuration file; see the reference section for "LogToSyslog — Specify whether to send access information to the system log (Linux and UNIX only)" on page 216.

Related configuration file directives

To set up logs by using the proxy configuration file, see the reference sections in Appendix B, "Configuration file directives," on page 161 for the following directives:

- "AccessLog — Name the path for the access log file" on page 163
- "CacheAccessLog — Specify the path for the cache access log files" on page 175
- "ErrorLog — Specify the file where server errors are logged" on page 196
- "EventLog — Specify the path for the event log file" on page 198
- "LogToSyslog — Specify whether to send access information to the system log (Linux and UNIX only)" on page 216
- "ProxyAccessLog — Name the path for the proxy access log file" on page 239

Access log filters

Access logs record the activity of the host machine, the proxy, and the cache. For each access request that your proxy receives, an entry in the appropriate access log includes the following information:

- What was requested
- When it was requested
- Who requested it
- The method of the request
- The type of file that your server sent in response to the request
- The return code, which indicates whether or not the request was successful
- The size of the data that was sent

Access errors are logged in the server's error log.

Reasons to control what is logged

There are several reasons to restrict what is logged:

- To reduce log size

  The log files of a busy server can become large enough to fill the server's disk space. By default, all access requests are logged, which means that log entries are made not only for an HTML page but for each image that the page contains. Including only meaningful access requests can significantly reduce the number of entries in the log. For example, you might want to configure your access logs so that they include log entries for HTML page access requests, but not for requests for GIF images.
- To collect targeted information

  For example, if you are interested in who is accessing your server from outside your company, you can filter out access requests that originate from IP addresses within your company. If you are interested in finding out the size of the audience for a particular Web site, you can create an access log that shows only the access requests for that URL.

Information that is excluded from access logs is not recorded in any access report and is not available for future use. Therefore, if you are unsure about how much access information you need to track, apply exclusion filters conservatively until you gain experience in monitoring your server.

Configuring access log filters

Access log entries can be filtered based on any of the following attributes:

- URL (files or directories)

- IP address or host name
- User agents
- Method
- MIME type
- Return code

To specify your filters, in the Configuration and Administration forms, select **Server Configuration –> Logging –> Access Log Exclusions**. Specify only those exclusions that you want. You do not need to use all the categories.

- In the section headed **Do not log requests to the following Directories or Files in the Access Log**, list the URL paths for which you want to exclude log entries.
- In the section headed **Do not log requests from the following user-agents**, list the proxy agents for which you want to exclude log entries.
- In the section headed **Do not log requests from the following Hostnames or IP addresses**, list the host names or IP addresses for which you want to exclude log entries.
- In the section headed **Do not log requests with the following Methods:**, check the boxes for any methods for which you want to exclude log entries.
- In the section headed **Do not log requests for files of the following MIME types**, check the boxes for any MIME types for which you want to exclude log entries.

  Note: This directive only affects the Proxy access log. It is not possible to filter a log listing these cached objects by their MIME type. Use AccessLogExcludeURL to do this.
- In the section headed **Do not log requests with the following Return Codes**, check the boxes for request return codes for which you want to exclude log entries.

Click **Submit**.

Related configuration file directives

To set access log filters by using the proxy configuration file, see the reference sections in Appendix B, "Configuration file directives," on page 161 for the following directives:

- "AccessLogExcludeMethod — Suppress log entries for files or directories requested by a specified method" on page 164
- "AccessLogExcludeMimeType — Suppress Proxy access log entries for specific MIME types" on page 164
- "AccessLogExcludeReturnCode — Suppress log entries for specific return codes" on page 165
- "AccessLogExcludeURL — Suppress log entries for specific files or directories" on page 165
- "AccessLogExcludeUserAgent — Suppress log entries from specific browsers" on page 166
- "NoLog — Suppress log entries for specific hosts or domains that match a template" on page 223

Default log settings

- **Default paths**

All logs are enabled in the Caching Proxy default configuration. They are stored in the logs/ subdirectory of the installation directory. The default paths are the following:

– Local (administrative) access logs:
  - Linux and UNIX: /opt/ibm/edge/cp/server_root/logs/local
  - Windows: *drive*:\Program Files\IBM\edge\cp\logs\local
– Cache access log:
  - Linux and UNIX: /opt/ibm/edge/cp/server_root/logs/cache
  - Windows: *drive*:\Program Files\IBM\edge\cp\logs\cache
– Proxy access log:
  - Linux and UNIX: /opt/ibm/edge/cp/server_root/logs/proxy
  - Windows: *drive*:\Program Files\IBM\edge\cp\logs\proxy
– Error log:
  - Linux and UNIX: /opt/ibm/edge/cp/server_root/logs/error
  - Windows: *drive*:\Program Files\IBM\edge\cp\logs\error
– Event log:
  - Linux and UNIX: /opt/ibm/edge/cp/server_root/logs/event
  - Windows: *drive*:\Program Files\IBM\edge\cp\logs\event

Each log file name is a combination of the base name and a date suffix in the form *.Mmmddyyyy*, for example, proxy.Feb292000.

- **Default formats**

  Logs are stored in the common file format by default. A combined log format is also available and can be set by adding the following line to the proxy configuration file (ibmproxy.conf).

  ```
  LogFileFormat combined
  ```

  The combined log format is similar to the common format but has added fields that show the referrer, the user agent, and cookie information. Local time format is the default time format.
- **Default content**

  By default, all access requests are recorded in the appropriate access log, and no access information is recorded in the system log. Error log information is written to the error log only, and event log information is written to the event log only.
- **Default maintenance**

  In the default configuration, logs are not archived or deleted.

Maintaining and archiving logs

The Caching Proxy uses a plug-in to manage logs. For further information, see the reference page in Appendix B, "Configuration file directives," on page 161 for the configuration file directive "Midnight — Specify the API plugin used to archive logs" on page 221.

You can specify how to archive and remove daily logs. The basic options are:

- Compress and remove logs that are older than a specified age.
- Remove logs after they reach a specified age or after that log category reaches a specified collective size.
- Run your own program at midnight each night to maintain and archive the logs.

By default, the current and previous days' logs are never deleted by any maintenance agent. All of the current day's logs and the previous day's cache access log are never compressed by any maintenance agent.

To configure log maintenance, in the Configuration and Administration forms, select **Server Configuration –> Logging –> Log Archiving**. In this form, use the drop-down box to specify the maintenance method.

- If you have choose **Purge**, set the age, file size, or both, to be used in determining which logs to delete. When files are purged by age and size, files older than the maximum age are purged before files larger than the maximum size. When files are purged by size, older logs are deleted first.
- If you choose **Compress**, set the age of logs to be compressed and the command to use for compressing log files (include all parameters). Also set the maximum age of logs. After compressing the logs, the maintenance agent deletes compressed logs older than the maximum age.

Related configuration file directives

To configure log archiving using the proxy configuration file, see the reference pages in Appendix B, "Configuration file directives," on page 161 for the following directives:

- "CompressAge — Specify when to compress logs" on page 185
- "CompressDeleteAge — Specify when to delete logs" on page 186
- "CompressCommand — Specify the compression command and parameters" on page 185
- "LogArchive — Specify the behavior of log archiving" on page 215
- "Midnight — Specify the API plugin used to archive logs" on page 221
- "PurgeAge — Specify the age limit for a log" on page 243
- "PurgeSize — Specify the limit for the size of the log archive" on page 243.

Log file scenario

The following example shows how you can customize logging to meet your needs. Suppose you have just purchased and installed the Caching Proxy. You want to set up your server to log access and error information with the following requirements:

- Logs must carry a local time stamp and must be in a common log file format.
- Access logs must be purged when they are more than 30 days old or when the logs reach a collective size of 25 MB.
- The following request types must not be logged to the access logs:
  - Requests for GIF images
  - Requests from hosts with IP addresses that match the pattern 130.128.*.*
  - Redirection requests (these requests yield a return code between 300 and 399)

To configure the Caching Proxy to keep logs according to these criteria, in the Configuration and Administration forms, select **Server Configuration –> Logging**.

1. Optionally, select the **Log Files form** to specify paths for the access log files. (Default paths are provided.)
2. Use the **Log Archiving** form to specify how to archive the files:
   - Specify **Purge** as the archiving method.
   - Under **When using Purge**, fill in the fields as follows:

- **Delete logs older than** 30 **Days**
- **Delete logs larger than** 25 **MB**

3. Use the **Access Log Exclusions** form to filter log entries as follows:
   - In the **Do not log requests from the following hostnames or IP addresses** list, add 130.128.*.* in the **Excluded Host** field.
   - Under **Do not log requests for files of the following MIME types**, select the **image/gif** check box.
   - Under **Do not log requests with the following Return Codes**, select the **(3xx) Redirection** check box.

Following these directions produces the following lines in the proxy configuration file:

```
LogArchive purge
PurgeAge 30
PurgeSize 25
AccessLogExcludeURL *.gif
NoLog 130.128.*.*
AccessLogExcludeReturnCode 300
```

Chapter 31. Using the Server Activity Monitor

The Caching Proxy's Server Activity Monitor displays server and network performance statistics, server and network status, and access log entries. The monitor can be used remotely and does not need to be on the same machine that is running the proxy server. The Server Activity Monitor is enabled by default and requires no configuration.

There are two ways to open the Server Activity Monitor:

- In any connected Web browser, enter the following URL, using your server's full name where indicated.

  `http://your.server.name/Usage/Initial`
- In the Configuration and Administration forms, select **Server Activity Monitor**.

Unlike other forms in the configuration client, the forms in this category do not set configurations for the server, but display data about server usage. These forms provide considerably more information than can be displayed in a single console window.

The following sections show the type of information that the **Server Activity Monitor** provides, and suggest how to use the information to tune performance.

Several **Server Activity Monitor** pages are available:

- **Activity Statistics**
- **Network Statistics**
- **Access Statistics**
- **Proxy Access Statistics**
- **Cache Statistics**
- **Cache Refresh Summary**

Each of the pages has a **Refresh** button, which can be used to update the information.

Activity Statistics

Table 4 shows an example of the **Activity Statistics** page.

Table 4. Activity Statistics

Activity Statistics	
Connections	1 Active, 431 maximum
Response time	Not available
Throughput	0 connections/second
Requests processed today	0
Total requests processed	114
Request errors	3

These server activity statistics can be used to monitor the server traffic in terms of number of access requests, response time, throughput, requests processed today, total requests processed, and errors. The following configuration changes affect the statistics on the **Activity** page.

- **Number of active threads**—This specifies how many threads to use for server requests. You can increase or decrease the number of threads available, depending on how much memory you have. To change the number of active threads, in the Configuration and Administration forms, select **Server Configuration –> Server Management –> Performance**, or edit the MaxActiveThreads directive in the configuration file. (See "MaxActiveThreads — Specify the maximum number of active threads" on page 218.)
- **Persistent connections**—Whether the proxy allows persistent connections with a client can have an effect on network throughput. To change this setting in the Configuration and Administration forms, select **Proxy Configuration –> Proxy Performance** to enable or disable persistent connections, and select **Server Configuration –> Server Management** to define how to maintain the connections. To alter these settings by using the configuration file, see the reference sections for the following directives:
  - "MaxPersistRequest — Specify the maximum number of requests to receive on a persistent connection" on page 219
  - "PersistTimeout — Specify the time to wait for the client to send another request" on page 228
  - "ProxyPersistence — Allow persistent connections" on page 241

Network Statistics

Table 5 shows an example of the **Network Statistics** page.

Table 5. Network Statistics

Network Statistics	
Outgoing data:	1K bytes/second
Incoming data:	1K bytes/second
Bandwidth saved:	3 K bytes (0K bytes/second)
Bandwidth saved today:	0 K bytes (0 K bytes/second)

The **Network Statistics** form provides information about the network the proxy is running on, including the data speeds for bytes sent and received.

Access Statistics

The **Access Statistics** page displays the 20 most recent entries in the access logs. This page displays recent entries in the Proxy Access log (in black type) and the cache access log (in blue type). You can customize what is displayed by customizing what is logged. For more information on the access log statistics, see "Access log filters" on page 138.

Proxy Access Statistics

The **Proxy Access Statistics** form provides information on the proxy activity, such as which URLs were requested and whether they were served from the cache. Following the URLs are the return codes given to the clients and the file size in bytes. The following settings can improve the proxy access statistics:

- Use automatic cache refresh to increase the likelihood that a requested document is in the cache. See Chapter 20, “Configuring the cache agent for automatic refreshing and preloading,” on page 87 for details.
- Increase the minimum hold time for cached files. See “Configuring cache freshness” on page 84 for details.
- Do not cache files served from your local domain. Although this setting tends to decrease the number of requests served from the cache, there is no loss of performance if files are served from your local intranet as quickly as they are served from the cache (in some cases, it is faster). See Chapter 18, “Controlling what is cached,” on page 77 for details.

Cache Statistics

If caching is enabled, the **Cache Statistics** page shows recent cache access information. It provides information on the cache and index, including the following:

- Whether the cache is currently operational or is being reindexed from a server start
- Whether garbage collection is running
- Cache hit rates

Many cache configuration options change the cache statistics results (see Part 4, “Configuring proxy server caching,” on page 67).

Cache Refresh Summary

If the cache agent is configured to preload files in the cache, the **Cache Refresh Summary** page shows information on the most recent run of the cache agent. The cache agent must have run at least once to display any information. To change the way the cache refresh agent works, consider the following:

- If most of the traffic on your intranet is not to local Web sites, consider disabling caching for the local domain. See Chapter 18, “Controlling what is cached,” on page 77 for details.
- If many clients request a page that does not appear in the cache access log, you can manually configure the URL to be loaded. See “Related proxy configuration file directives” on page 90 for instructions.
- Adjust the number of popular URLs to preload. See “Related proxy configuration file directives” on page 90 for instructions.
- Specify the maximum time the cache agent can run. See “Related proxy configuration file directives” on page 90 for instructions.

Appendix A. Using Caching Proxy commands

This appendix provides a reference for proxy server commands.

cgiparse command

Purpose

Use the **cgiparse** command to parse the QUERY_STRING environment variable for CGI scripts. If the QUERY_STRING environment variable is not set, the command reads CONTENT_LENGTH characters from its standard input. All returned output is written to its standard output.

Format

```
cgiparse -Flag [Modifier]
```

Parameters

Flags, their one-character equivalents (-k -f -v -r -i -s -p -c -q -P) and their function, include:

-keywords | -k
Parses QUERY_STRING for keywords. Keywords are decoded and written to standard output, one per line.

-form | -f
Parses QUERY_STRING as form request. Returns a string which, when evaluated by the shell, sets shell variables with the prefix FORM_ followed by a field name. Field values are the contents of the variables.

-value *field-name* **| -v** *field-name*
Parses QUERY_STRING as form request. Returns only the value of *field-name*.

-read | -r
Reads CONTENT_LENGTH characters from standard input and writes them to standard output.

-init | -i
If QUERY_STRING is not set, reads the value of standard input and returns a SET statement that sets QUERY_STRING to this value. This can be used with both the GET and POST methods. A typical use is:

```
eval 'cgiparse -init'
```

This sets the QUERY_STRING environment variable, regardless of whether the GET or POST method was used.

cgiparse can be called multiple times in the same script when the GET method is used, but it must only be called once if the POST method is used. With the POST method, after standard input is read, the next **cgiparse** finds standard input empty and waits indefinitely.

-sep *separator* **| -s** *separator*
Specifies the string used to separate multiple values. If you are using the **-value** flag, the default separator is newline. If you are using the **-form** flag, the default separator is a comma (,).

-prefix *prefix* **| -p** *prefix*
Used with **-POST** and **-form**, specifies the prefix to use when creating environment variable names. The default is ″FORM_″.

-count | -c
Used with **-keywords**, **-form**, and **-value**, returns a count of items related to these flags.

-keywords | -k
Returns the number of keywords.

-form | -f
Returns the number of unique fields (multiple values are counted as one).

-value *field-name* **| -v** *field-name*
Returns the number of values for *field-name* (if there is not a field named *field-name*, output is `0`).

-*number*
Used with **-keywords**, **-form**, and **-value**, returns the specified occurrence related to these flags.

-keywords
Returns the *n*'th keyword. (For example **-2 -keywords** outputs the second keyword.)

-form
Returns all the values of the *n*'th field. (For example **-2 -form** outputs all the values of the second field.)

-value *field-name*
Returns the *n*'th of the multiple values of field *field-name*. (For example **-2 -value -whatsit** outputs the second value of the **whatsit** field).

-quiet | -q
Suppresses all error messages. (Non-zero exit status still indicates error.)

-POST | -P
Information from standard input (or if a filename is intended, the stdin file) is directly decoded and parsed into shell variables, QUERY_STRING is not used. **-POST** is equivalent to consecutive use of the **-init** and **-form** options.

Examples

The following examples ignore the fact that, in reality, QUERY_STRING is already set by the server. In the following examples, $ is the Bourne shell prompt.

- Keyword search

```
$ QUERY_STRING="is+2%2B2+really+four%3F"
$ export QUERY_STRING
$ cgiparse -keywords
is
2+2
really
four?
$
```

- Parsing all form fields

```
$ export QUERY_STRING="name1=Value1&name2=Value2%3f+That%27s+right%21";
$ cgiparse -form
FORM_name1='Value1'; FORM_name2='Value2? That'\''s right!'
$ eval `cgiparse -form`
$ set | grep FORM
FORM_name1="Value1"
FORM_name2="Value2? That's right!"
$
```

- Extracting only one field value

```
$ QUERY_STRING="name1=value1&name2=Second+value%3F+That'\''s%27s
$ cgiparse -value name1
value1
$ cgiparse -value name2
Second value? That's right!
$
```

Results

0 Success

1 Illegal command line

2 Environment variables not set correctly

3 Failed to get requested information (for example, there is no such field or QUERY_STRING contains keywords when form field values are requested)

Note: When you receive one of these error codes, you may receive additional informational messages, too. The message varies depending on the command issued.

cgiutils command

Purpose

Use the **cgiutils** command in no-parse header programs to produce a full HTTP 1.0 response.

Note: If you want to supply your own no-parse header (nph) programs specifically to return your own return values, the name of the program must begin with **nph-**. This prevents the server header from overriding your return value with the standard server return value.

Format

```
cgiutils -Flag [Modifier]
```

If *Modifier* contains blanks, enclose it in quotation marks (*""*).

Parameters

-version
Returns version information.

-nodate
Does not return the **Date:** header.

-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 want only the **Expires:** header.

-reason *explanation*
Specifies the reason line for the HTTP response. You can use only this flag with the **-status** *nnn* flag.

-ct [*type/subtype***]**
Specifies MIME Content-Type header. This example specifies a MIME content type of text/html:

```
cgiutils -ct text/html
```

If you omit the *type/subtype*, the MIME content type is set to the default text/plain. This example sets the MIME content type to text/plain.

```
cgiutils -ct
```

-ce *encoding*
Specifies MIME Content-Encoding header. For example:

```
cgiutils -ce  x-compress
```

-cl *language-code*
Specifies MIME Content-Language header. For example:

```
cgiutils -cl en_UK
```

-length *nnn*
Specifies MIME Content-Length header.

-expires *Time-Spec*
Specifies MIME **Expires:** header. This flag specifies the time to live (the

expiration date of a document) in any combination of days, hours, minutes, and seconds. This is the length of time a document is considered valid. For example:

```
cgiutils -expires 2 days 12 hours
```

The **cgiutils** command adds the time you specify to the current Greenwich Mean Time to determine the expiration date. The expiration date is put in the **Expires:** header in the HTTP format.

-expires now
Produces an **Expires:** header that matches the **Date:** header.

-uri *URI*
Specifies the Universal Resource Identifier (URI) for the returned document. URI can be considered to be the same as URL.

-extra *xxx: yyy*
Specifies an extra header that cannot otherwise be specified for the **cgiutils** command.

Examples

- This example calculates the expiration date for the **Expires:** header.

  ```
  cgiutils -expires "1 year 3 months 2 weeks 4 days 12 hours 30 mins"
  ```

- The following example specifies a status code and reason, and sets the **Expires:** header equal to the **Date:** header.

  ```
  cgiutils -status 200 -reason "Virtual doc follows" -expires now
  ```

  This might produce headers similar to these:

  ```
  HTTP/1.0 200 Virtual doc follows
  MIME-Version: 1.0
  Server: IBM-ICS
  Date: Tue, 05 Jan 1996 03:43:46 GMT
  Expires: Tue, 05 Jan 1996 03:43:46 GM
  ```

  The cgiutils command automatically produces the **Server:** header because it is available in the CGI environment. The **Date:** field is also automatically generated unless the **-nodate** flag is specified.

  This includes a blank line after the output to mark the end of the MIME header section. If you want to follow this with some more headers yourself, use the **-noel** (NO-Empty-Line) flag as shown in the next example.

- If you do not want the blank line after the header line, use the **-noel** flag:

  ```
  cgiutils -noel -expires "2 days" -nodate
  ```

  ```
  HTTP/1.0 200 Virtual doc follows
  MIME-Version: 1.0
  Server: IBM-ICS
  Expires: Tue, 07 Jan 1996 03:43:46 GMT
  ```

htadm command

Purpose

Use the **htadm** command to control your server password files. Your server uses password files to control access to your files. You can add a user name to a password file, delete a user from a password file, verify a user's password, and create an empty password file. You can also change a password for a user by first deleting the user's password and then creating a new one.

Note: When you use the **htadm** command to add a user, change a password, or check a password, you must enter the password on the command line. Because the command destroys the password from the command line as soon as possible, it is highly unlikely that you can see a user's password by looking at the process listing on the machine (with the **ps** command, for example).

Format

```
htadm -Flag  [Modifier]
```

Parameters

-adduser *password-file user-name* **[***password* **[***real-name***]]**

Adds a user and password into the password file. If you enter the command with only *password-file*, you are prompted for the other parameters.

password-file

The path and name of the password file to which you want to add the user.

user-name

The name of the user you want to add.

Use only alphabetic and numeric characters for the user name; do not use special characters.

The command fails if there is already a user of the same name in the password file.

password

The password you want to define for the user name.

Passwords can be up to 32 characters long. Use only alphabetic and numeric characters for the password; do not use special characters.

Notes:

1. Some browsers are unable to read and send passwords longer than eight characters. Because of this limitation, if you define a password longer than eight characters, the server recognizes either the complete password or just the first eight characters of the password as valid.
2. The administrator user name and password are case-sensitive even if the operating system is not case-sensitive. Be sure to input the exact user name and password entered using the `htadm` command when accessing the Configuration and Administration forms.

real-name

A comment or name you want to use to identify the user name you are adding. Whatever you enter will be written into the password file.

-deluser *password-file* **[***user-name***]**
Deletes a user from the password file. If you enter the command with only *password-file*, you are prompted for the *user-name* parameter.

password-file
The path and name of the password file from which you want to delete a user.

user-name
The name of the user you want to delete. The command fails if the user name you specify does not exist in the password file.

-passwd *password-file* **[***user-name* **[***password***]]**
Changes the password for a user name already defined in the password file. If you enter the command with only *password-file*, you are prompted for the other parameters.

password-file
The path and name of the password file that contains the user name whose password you want to change.

user-name
The user name whose password you want to change. The command fails if the user name you specify does not exist in the password file.

password
The new password you want to define for the user name.

Passwords can be up to 32 characters long. Use only alphabetic and numeric characters for the password; do not use special characters.

Notes:

1. Some browsers are unable to read and send passwords longer than eight characters. Because of this limitation, if you define a password longer than eight characters, the server recognizes either the complete password or just the first eight characters of the password as valid.
2. The administrator user name and password are case-sensitive even if the operating system is not case-sensitive. Be sure to input the exact user name and password entered using the `htadm` command when accessing the Configuration and Administration forms.

-check *password-file* **[***user-name* **[***password***]]**
Verifies the password for a user name already defined in the password file and lets you know if it is correct or not. If you enter the command with only *password-file*, you are prompted for the other parameters.

password-file
The path and name of the password file that contains the user name whose password you want to verify.

user-name
The user name whose password you want to verify. The command fails if the user name you specify does not exist in the password file.

password
The password that you want to verify. If the password you enter is the one defined for the user name, the command writes `Correct` to standard output and completes with a `0` return code. If the password you enter is not the one defined for the user name, the command writes `Incorrect` to standard output.

-create *password-file*
Create an empty password file.

password-file
The path and name of the password file that you want to create.

Examples

- To add a user to a password file:
  - Linux and UNIX systems:

    ```
    htadm -adduser /opt/ibm/edge/cp/server_root/protect/heroes.pwd
    clark superman "Clark Kent"
    ```

  - Windows systems:

    ```
    htadm -adduser "C:\Program Files\IBM\edge\cp\server_root\protect\
    heroes.pwd" clark superman "Clark Kent"
    ```

  Note: The **htadm** command must be on one line. It is shown here on more than one line for readability. Enter the actual command on one line with at least one space between `clark` and `superman`.
- To delete a user from a password file:
  - Linux and UNIX systems:

    ```
    htadm -deluser /opt/ibm/edge/cp/server_root/protect/
         heroes.pwd clark superman "Clark Kent"
    ```

  - Windows systems:

    ```
    htadm -deluser "C:\Program Files\IBM\edge\cp\server_root\protect\
         heroes.pwd" clark superman "Clark Kent"
    ```

htcformat command

Purpose

Use the **htcformat** command to prepare a raw device or file to hold the proxy cache. This format command must be used before the device is specified for use with the proxy cache.

The device path must specify the raw device. See your file system documentation for details on how to access raw devices. Examples are available in Part 4, "Configuring proxy server caching," on page 67.

Note: Linux 2.2 kernels do not support caching to raw devices. On Linux platforms, only files and memory can be used for cache storage.

The minimum size for a Caching Proxy cache is 16392 KB, which is 2049 blocks.

Format

```
htcformat device [-blocksize <block size>] [-blocks number of blocks]
htcformat -file filepath [-blocksize block size] -blocks number of blocks
```

Parameters

-blocksize
This sets the size of blocks in the medium of the cache device. Block size is in bytes. The default is 8192 and should be used for all situations.

-blocks
Number of blocks to create on the device or in the file. When formatting a file, this argument is required in order to specify the file size. This argument also can be used to limit the amount of a particular device or partition to be used for cache storage. If no blocks argument is specified, as many blocks as will fit on the partition will be created.

-file
Format a file instead of a storage device.

Usage

The caching system additionally segregates cache files or devices into containers for indexing and garbage collection. The size of containers is set to a certain number of blocks; container size cannot be configured. In order for garbage collection to run, a minimum of two containers is required; the minimum cache size is 16392 KB.

The **htcformat** command rejects a format request that yields a cache device with fewer than two containers.

Examples

The following example formats a disk partition called c0t0d0s0 on Solaris.

```
htcformat /dev/rdsk/c0t0d0s0
```

The following example formats a disk partition called lv02 on AIX.

```
htcformat /dev/rlv02
```

The following example formats a disk partition called d: on Windows.

```
htcformat \\.\d:
```

The following example formats a file named filecache to be about 1 GB large.

```
htcformat -file /opt/ibm/edge/cp/filecache -blocks 131072
```

ibmproxy command

Purpose

Use the **ibmproxy** command to start the server.

You can set all these flags (except **-r**) using the directives in the server configuration file.

It is common practice to create a file named README containing instructions or notices to be read by anyone new to the directory. By default, the **ibmproxy** command embeds any README file in the hypertext version of a directory. The README file instructions can also be set with the DirReadme configuration directive.

Format

```
ibmproxy [-Flag [-Flag [-Flag..]]]
```

Parameters

-nobg
Runs the server as a foreground process, not as a background process. The default is to run as a background process.

-nosnmp
Turns SNMP support off.

-p *port-number*
Listens on this port number. The default port number is 80. This flag overrides the Port directive specified in the configuration file. To use the default value or the value specified in the configuration file, omit this flag.

-r *configuration-file*
Specifies the file to use as the configuration file. You must use this flag if you want to start the server with a configuration file other than the default configuration file. This allows use of multiple configuration files.

-restart
Restarts a server that is currently running. The **ibmproxy** command gets the process number of the server that is running from the PidFile and sends the process number to the HangUP signal (HUP). It then reloads its configuration files and reopens its log files. To avoid corruption do not run two instances of the server at the same time using the same PidFile, log files, and proxy cache.

Because the **http** daemon must read the configuration file the server is currently using in order to access the PidFile, you must specify the same configuration file when restarting. If you used the **-r** flag and a specific configuration file when you started the server, then you must specify this flag and the same file with **-restart**.

-snmp
Turns SNMP support on.

-unload
On AIX, this unloads the transparent proxy kernel extension. On Linux, this removes the associated firewall rules.

Signal handling options also exist on Linux and UNIX platforms only. On Linux and UNIX platforms, the following options are available.

SIGTERM
The **ibmproxy** command stops and exits when complete. You can use SIGKILL or CANCEL to immediately terminate.

SIGHUP
If running, the **ibmproxy** command restarts, reloads the configuration file, and continues processing.

Examples

- To start the server on port 8080, using the /usr/etc/ibmproxy.conf configuration file instead of the default, /etc/ibmproxy.conf, enter:

  ```
  ibmproxy -p 8080 -r /usr/etc/ibmproxy.conf
  ```

- On AIX, to start a server with the default configuration file using the System Resource Controller, enter:

  ```
  startsrc -s ibmproxy
  ```

  If the default configuration file does not exist, the **ibmproxy** command exports the /Public directory tree. This tree can contain soft links to other directory trees.

Appendix B. Configuration file directives

This appendix describes the directives contained in the ibmproxy.conf configuration file.

- **On Linux and UNIX systems.** These directives are located in the ibmproxy.conf configuration file in the /etc/ directory.
- **On Windows systems.**These directives are typically located in C:\Program Files\IBM\edge\cp\.

Use this information as a reference if you configure the server by editing the ibmproxy.conf file. If you use the Configuration and Administration forms, you do not need to refer to this chapter.

Directives are listed alphabetically.

Directives not changed on restart

Some directives are not refreshed when the server is restarted. If the following directives are changed while the server is running, you must manually stop and then manually start the server again. (See Chapter 5, “Starting and stopping the Caching Proxy,” on page 15.)

Table 6. Directives not refreshed on a restart

Directive group	Directives
CGI	DisinheritEnv, InheritEnv
Caching	Caching
Logging	AccessLog, CacheAccessLog, ErrorLog, ProxyAccessLog, ServerRoot
Network Access	BindSpecific, Hostname, ListenBacklog, Port
Performance	MaxActiveThreads
RTSP	All RTSP directives
SSL	All SSL directives
Linux and UNIX Process Control	GroupId, UserId
Miscellaneous	TransparentProxy

Overview of directives

This appendix provides the following information about each directive:

- A heading with the name and a brief description of the directive
- Usage instructions
- The format for the directive, which follows the general syntax:

  ```
  DirectiveName value
  ```

- Where appropriate, an example of a possible setting of the directive in the configuration file

Note: Examples in directives with Windows-specific paths sometimes contain *server_root*, which is the root directory for the server selected at installation.

- The default value or values of the directive

  These are the original values coded in the default configuration file. Change only the parts of the configuration file that you want to be different from the defaults. A directive that has no default value initially coded appears in the file preceded by a comment marker (#). If you want to specify a value for the directive, remove the comment marker and add the value to the line in the configuration file.

Acceptable values

The following list includes values that are accepted in the configuration file:

- In the reference information for some directives, the *value* portion contains templates for requests, path names, or host names. Except where otherwise indicated, you can use the asterisk character (*) in templates. For the template to be matched, an asterisk can be replaced by any character string or single character.
- Configuration directives that allow you to enter a positive string accept these values:
  - `Yes`
  - `On`
  - `OK`
  - `Enable`
- Configuration directives that allow you to enter a negative string accept these values:
  - `No`
  - `Off`
  - `None`
  - `Disable`
- Configuration directives that allow you to specify an amount of time accept any combination of the following:
  - *hh*—hours
  - *hh*`:`*mm*—hours and minutes
  - *hh*`:`*mm*`:`*ss*—hours, minutes, and seconds
  - *n* `years`—number of 365-day years
  - *n* `months`—number of 30-day months
  - *n* `weeks`—number of 7-day weeks
  - *n* `days`—number of 24-hour days
  - *n* `hours`—number of 60-minute hours
  - *n* `minutes`—number of 60-second minutes
  - *n* `seconds`—number of seconds
  - *n* `fortnights`—number of 14-day intervals

  All entries are converted to seconds and added together.
- Blank characters are not allowed within a file name specified in the configuration file. Blank spaces are treated as delimiters.

Syntax of configuration file records

When editing the configuration file, remember the following requirements:

- Each directive must begin on a new line.
- Values are separated by one or more blanks. No distinction is made between the space character and the Tab character.
- The beginning of a comment is indicated by a # symbol. All characters from the # symbol to the end of the line are ignored.
- If either a number sign or a blank needs to be specified for a directive, use the backslash character (\) as an escape character before it. An escape character indicates that the next character is to be interpreted as a character rather than as a command; for example, if \# is found on a line, the server interprets this as a # symbol, not as the beginning of a comment, and continues reading characters. If the character \ is found on a line, the server interprets this as a blank, not as a value delimiter, and continues reading characters to build the value.

Caching Proxy directives

The Caching Proxy directives follow.

AcceptAnything — Serve all files

Use this directive to serve files to the client even if the MIME type of the file does not match an `ACCEPT:` header sent by the client. If this directive is set to `OFF`, files whose MIME types differ from the types that the client can accept are not displayed. An error page is displayed instead.

Format

```
AcceptAnything  {on | off}
```

Example

```
AcceptAnything off
```

Default

```
AcceptAnything on
```

AccessLog — Name the path for the access log file

Use this directive to specify the directory and file where you want the server to log access statistics. By default, the server writes an entry to this log each time a client sends the server a request for data stored on the local server. Typically, these entries include only requests from the configuration client or accesses when the Caching Proxy machine is used as an origin server. This log does not contain proxy or cache access information.

Use the NoLog directive to specify clients whose requests you do not to be logged. For a description of the NoLog directive, refer to "NoLog — Suppress log entries for specific hosts or domains that match a template" on page 223.

The server starts a new log file each day at midnight if it is running. Otherwise, the server starts a new log file the first time you start it on any day. When creating the file, the server uses the file name you specify and appends a date suffix. The date suffix is in the format *Mmmddyyyy*, where *Mmm* is the first three letters of the month; *dd* is the day of the month; and *yyyy* is the year.

Note: If you change the server defaults for the user ID, group ID, or log directory paths, create the new directories and update the permissions and ownership of the directories. To enable the server to write information to a user-defined log directory, set the permission for that directory as 755, and set the user-defined server user ID as the owner. For example, if you change the user ID of the server from the default to jdoe, and the default logs directory to server_root/account, then the server_root/account directory must have the permission 755 and be owned by jdoe.

It is a good idea to remove old log files because they can use a significant amount of space on your hard drive.

Format

```
AccessLog  /directory_path/logfile_name
```

Example

```
AccessLog  /logs/accesslog
```

Defaults

- **Linux and UNIX systems:** `AccessLog /opt/ibm/edge/cp/server_root/logs/local`
- **Windows systems:** `AccessLog drive:\Program Files\IBM\edge\cp\logs\local`

AccessLogExcludeMethod — Suppress log entries for files or directories requested by a specified method

Use this directive to prevent the logging of requests made by a specific method for access to files or directories. For example, you might not want to log DELETE requests for files or directories.

You can have multiple occurrences of this directive in your configuration file. You can also put multiple methods on the same directive if you separate them by one or more spaces.

Format

```
AccessLogExcludeMethod method  [...]
```

Examples

```
AccessLogExcludeMethod GET
AccessLogExcludeMethod PUT
AccessLogExcludeMethod POST
AccessLogExcludeMethod DELETE
AccessLogExcludeMethod GET   PUT
```

Default

None. The server includes in the access log the files and directories requested by all types of methods.

AccessLogExcludeMimeType — Suppress Proxy access log entries for specific MIME types

Use this directive to specify that you do not want record in the Proxy access log requests for access to directories or files of a specified MIME type. (Examples of MIME types are text/html, image/gif, and image/jpeg.) For example, you might not want to log access requests for GIF images.

You can have multiple occurrences of this directive in your configuration file. You can also put multiple MIME types on the same directive if you separate them by one or more spaces.

Note: This directive only affects the Proxy access log. It is not possible to filter a log listing these cached objects by their MIME type. Use AccessLogExcludeURL to do this.

Format

```
AccessLogExcludeMimeType MIME_type  [...]
```

Example

```
AccessLogExcludeMimeType image/gif
AccessLogExcludeMimeType text/html
AccessLogExcludeMimeType image/gif   text/html
```

Default

None. The access log includes requests to the server for files and directories of all MIME types.

AccessLogExcludeReturnCode — Suppress log entries for specific return codes

Use this directive to specify that you do not want to log access requests that fall within a specified range of error code numbers. These error code numbers are proxy server status codes. You cannot specify individual codes. Specifying 300 indicates that you want to exclude access requests with redirection return codes (301, 302, 303, and 304).

You can have multiple occurrences of this directive in your configuration file. You can also put multiple return codes on the same directive if you separate them by one or more spaces.

Format

```
AccessLogExcludeReturnCode range
```

Example

```
AccessLogExcludeReturnCode 300
```

Default

None. The access log includes all requests to the server, regardless of the code.

AccessLogExcludeURL — Suppress log entries for specific files or directories

Use this directive to specify that you do not want to log requests for access to specific files or directories that match a specified URL template. For example, you might not want to log access requests for GIF files, or you might not want to log access requests to a particular file or directory on your server.

You can have multiple occurrences of this directive in your configuration file. You can also put multiple entries for the same directive if you separate them by one or more spaces.

Format

```
AccessLogExcludeURL  file_or_type [...]
```

Examples

```
AccessLogExcludeURL  *.gif
AccessLogExcludeURL  /Freebies/*
AccessLogExcludeURL  *.gif   /Freebies/*
```

Default

None. The server logs requests for access to all files and directories.

AccessLogExcludeUserAgent — Suppress log entries from specific browsers

Use this directive to specify that you do not want to log access requests made by specific user agents (for example, Internet Explorer 5.0).

You can have multiple occurrences of this directive in your configuration file. You can also put multiple entries for the same directive if you separate them by one or more spaces.

Format

```
AccessLogExcludeUserAgent user_agent [...]
```

Example

```
AccessLogExcludeUserAgent  *Mozilla/2.0
AccessLogExcludeUserAgent  *MSIE 5*
```

Default

By default, the ibmproxy.conf file contains the following definitions for the AccessLogExcludeUserAgent directive:

```
AccessLogExcludeUserAgent IBM_Network_Dispatcher_HTTP_Advisor
AccessLogExcludeUserAgent IBM_Network_Dispatcher_WTE_Advisor
```

The user agents listed above are those defined for certain Load Balancer advisors that typically sit in front of the Caching Proxy server. To improve performance by minimizing the number of writes to the log, these user agents are not logged. By default, the server logs access requests made by all other user agents.

AddBlankIcon — Specify the URL for the icon used to align the headings of directory listings

Use this directive to specify an icon to use for aligning the headings on directory listings that are returned when the server acts as a proxy for FTP requests. The icons appear beside the associated files to help users differentiate the files.

The icon can either be a blank icon or another icon that you specify to appear on the headings of your directory listings. For proper alignment, the icon you use must be the same size as the other icons you are using on your directory listings.

Format

```
AddBlankIcon icon_URL   alternative_text
```

icon_URL

Specifies the last part of the URL for the icon. The server adds this value to the /icons/ directory to form the complete URL request. If the request is for a local file, the server translates the request through the mapping directives. For the icon to be retrieved, the mapping directives must allow the request to be passed.

If you are using the server as a proxy, the complete request must be a fully qualified URL pointing to your server.

alternative_text
Specifies the alternative text to use for the icon if the requesting browser is not displaying graphics.

Example

```
AddBlankIcon logo.gif  logo
```

Defaults

- **Linux and UNIX:** `AddBlankIcon blank.m.pm.gif`
- **Windows:** `AddBlankIcon blank.gif`

The default does not specify alternative text because the icon is blank.

AddDirIcon — Specify the icon URL for directories on directory listings

Use this directive to specify an icon for representing a directory on a directory listing.

Format

`AddDirIcon` *icon_URL* *alternatIve_text*

icon_URL

Specifies the last part of the URL for the icon. The server adds this value to the /icons/ directory to form the complete URL request. If the request is for a local file, the server translates the request through the mapping directives. For the icon to be retrieved, the mapping directives must allow the request to be passed.

If you are using the server as a proxy, the complete request must be a fully qualified URL pointing to your server. You must map the URL to a local file and make sure that the mapping directives allow the URL to be passed.

alternative_text
Specifies the alternative text to use for the icon if the requesting browser is not displaying graphics.

Example

```
AddDirIcon  direct.gif  DIR
```

Defaults

- **Linux and UNIX:** `AddDirIcon dir.m.pm.gif  DIR`
- **Windows:** `AddDirIcon dir.gif DIR`

AddEncoding — Specify the MIME content encoding of files with particular suffixes

Use this directive to bind files with a particular suffix to a MIME encoding type. This directive is seldom used.

Format

`AddEncoding` *.extension encoding*

.extension
Specifies the file suffix pattern.

encoding
Specifies the MIME encoding type that you want to bind to files that match the corresponding suffix pattern.

Example
```
AddEncoding .qp   quoted_printable
```

Default
```
AddEncoding .Z  x-compress
```

AddIcon — Bind an icon to a MIME content type or encoding type

Use this directive to specify icons for representing files with a specific MIME content type or encoding type. The server uses the icons on directory listings, including FTP directory listings.

Format
```
AddIcon icon_URL  alternative_text  MIME_type_template
```

icon_URL

Specifies the last part of the URL for the icon. The server adds this value to the /icons/ directory to form the complete URL request. If the request is for a local file, the server translates the request through the mapping directives. For the icon to be retrieved, the mapping directives must allow the request to be passed.

If you are using the server as a proxy, the complete request must be a fully qualified URL pointing to your server. You must map the URL to a local file and make sure that the mapping directives allow the URL to be passed.

alternative_text
Specifies the alternative text to use for the icon if the requesting browser is not displaying graphics.

type_template
Specifies either a MIME content-type or encoding-type template. Content-type templates always contain a slash (/). Encoding-type templates never have a slash.

Example
```
AddIcon   video_file.m.pm.gif    MOV    video/*
```

Defaults
Numerous defaults are set for the AddIcon directive in the ibmproxy.conf configuration file.

AddParentIcon — Specify the URL for the icon representing a parent directory on directory listings

Use this directive to specify an icon for representing a parent directory on directory listings.

Format
```
AddParentIcon   icon_URL   alternative_text
```

icon-URL

Specifies the last part of the URL for the icon. The server adds this value to the /icons/ directory to form the complete URL request. If the request is for a local file, the server translates the request through the mapping directives. For the icon to be retrieved, the mapping directives must allow the request to be passed.

If you are using the server as a proxy, the complete request must be a fully qualified URL pointing to your server. You must map the URL to a local file and make sure that the mapping directives allow the URL to be passed.

alternative_text
Specifies the alternative text to use for the icon if the requesting browser is not displaying graphics.

Example

```
AddParentIcon  parent.gif  UP
```

Default

```
AddParentIcon   dir-up.gif    UP
```

AddType — Specify the data type of files with particular suffixes

Use this directive to bind files with a particular suffix to a MIME type and subtype. You can have multiple occurrences of this directive in your configuration file. The server provides defaults for the most-commonly used suffixes.

Format

AddType *.extension type/subtype encoding* [*quality*[*character_set*]]

.extension
The file suffix pattern. You can use the wildcard character (*) only on the following two special suffix patterns:

. Matches all file names that contain a dot character (.) and are not matched by other rules.

***** Matches all file names that do not contain a dot character (.) and are not matched by other rules.

type/subtype
The MIME type and subtype you want to bind to files that match the corresponding suffix pattern.

encoding
The MIME content encoding to which the data has been converted. Encoding is also used by an FTP proxy server to determine whether the file is retrieved in binary mode. In most cases, the appropriate encoding is `7bit`, `8bit`, or `binary`, and is determined as follows:

7bit Data is all represented as short (less than 1000 characters) lines of 8859-1 ASCII data. Source code or plain text files usually fall into this category. Exceptions are files containing line-drawing characters or accented characters.

8bit Data is represented as short lines, but can contain characters with the high bit set (for example, line-drawing characters or accented characters). PostScript files and text files from European sites usually fall into this category.

binary This encoding can be used for all data types. Data can contain not only

non-ASCII characters but also long (greater than 1000 characters) lines. Almost every file of type `image/*`, `audio/*`, and `video/*` falls into this category, as do binary data files of type `application/*`.

Any other value of encoding receives the same treatment as binary and is passed in MIME headers as a content encoding MIME header. The specification `7bit` and `8bit` is not sent in MIME headers.

quality
Specifies an optional indicator of relative value (on a scale of 0.0 to 1.0) for the content type. The quality value is used if multiple representations of a file are matched by a request. The server selects the file that is associated with the highest quality value. For example, if the file internet.ps is requested, and the server has the following AddType directives set, the server uses the `application/postscript` line because its quality number is higher.

```
AddType  .ps application/postscript  8bit  1.0
AddType  *.* application/binary   binary 0.3
```

character_set
An optional indicator of the character set you want to associate with text files. For the files to which you assign a character set, the server tells client browsers which character set to use when displaying the file. If you set a value for the *character_set* field, you must also include a value for the *quality* field.

Example

```
AddType .bin  application/octet-stream binary  0.8
```

Defaults

Numerous default settings for the AddType directive are contained in the configuration file (ibmproxy.conf).

AddUnknownIcon — Specify the icon URL for unknown file types on directory listings

Use this directive to specify an icon for representing files with an unknown file type on a directory listing.

Format

```
AddUnknownIcon   icon_URL   alternative_text
```

icon_URL

Specifies the last part of the URL for the icon. The server adds this value to `/icons/` to form the complete URL request. If the request is for a local file, the server translates the request through the mapping directives. For the icon to be retrieved, the mapping directives must allow the request to be passed.

If you are using the server as a proxy, the complete request must be a fully qualified URL pointing to your server. You must map the URL to a local file and make sure that the mapping directives allow the URL to be passed.

alternative_text
Specifies the alternative text to use for the icon if the requesting browser is not displaying graphics.

Example

```
AddUnknownIcon saywhat.gif  unknown
```

Defaults

- **Linux and UNIX:** `AddUnknownIcon unknown.gif ???`
- **Windows:** `AddUnknownIcon unknown.gif ???`

AdminPort — Specify the port for requesting administrative pages or forms

Use this directive to specify a port that administrators can use to access server status pages or configuration forms. Requests to this port are not queued with all other incoming requests on the standard port or ports that are defined with the Port directive. However, requests on the AdminPort go through the same normal access-control and request-mapping rules as, for example, Pass, Exec, Protect.

Note: The administration port must *not* be the same as the standard port or ports defined with the Port directive.

Format

`AdminPort` *`port_number`*

Example

```
AdminPort 2001
```

Default

```
AdminPort 8008
```

AggressiveCaching — Specify caching for noncacheable files

Use this directive to specify whether files returned by the origin server and marked noncacheable are to be cached anyway. Noncacheable files that are cached according to this directive are marked as `must revalidate`. Each time the file is requested, the proxy server sends an `If-Modified-Since` request to the origin server to revalidate the response before the response is served from the cache. Currently, the only noncacheable files affected by this directive are responses from the origin server that contain a `cache-control: no-cache` header. This directive can be specified multiple times.

Format

`AggressiveCaching` *`url_pattern`*

Examples

```
AggressiveCaching http://www.hosta.com/*
AggressiveCaching http://www.hostb.com/*
```

For backwards compatibility, the previous syntax for this directive (`AggressiveCaching {on | off}`) is now treated as follows:

`AggressiveCaching on` is treated as `AggressiveCaching * .`

`AggressiveCaching off` is ignored.

Note: If both `AggressiveCaching off` and `AggressiveCaching` *`url_pattern`* are specified, `AggressiveCaching off` is ignored, and a warning message is displayed.

Default

None

AlwaysWelcome — Specify whether to search the requested directory for welcome files

For requests that contain a directory name but not a file name, the AlwaysWelcome directive controls whether the server looks in the directory for a welcome file to return. By default, AlwaysWelcome is set to a value of `on`. This means the server always looks in the requested directory for a file matching a name specified on a Welcome directive. If a match is found, the file is returned to the requester. If the server finds more than one match between files in a directory and file names on Welcome directives, the order of the Welcome directives determines which file is returned. The server uses the Welcome directive closest to the top of the configuration file.

Format

```
AlwaysWelcome  on | off
```

Default

```
AlwaysWelcome  on
```

Related directives

- "Welcome — Specify the names of welcome files" on page 263

appendCRLFtoPost — Append CRLF to POST requests

Use this directive to specify URLs for which the Caching Proxy appends the carriage return and line-feed characters to the end of the body of a POST request. This directive can be specified multiple times.

Note: Specify this directive only for URLs that have a known problem processing POST requests.

Format

```
appendCRLFtoPost  url_pattern
```

Example

```
appendCRLFtoPost http://www.hosta.com/
```

Default

None

ArrayName — Name the remote cache array

Use this directive to specify the remote cache array to be shared by the servers.

Note: When setting up an array, configure the Hostname directive identically on all members of the array.

Format

```
ArrayName array_name
```

Default

None

Authentication — Customize the Authentication step

Use this directive to specify a customized application function you want the server to call during the Authentication step of the server request process. This code is executed according to the authentication scheme. Only `BASIC` authentication is supported.

Note: Authentication is part of the authorization process; it occurs only when authorization is required.

Format

`Authentication` *`type /path/file:function_name`*

`type`
Specifies an authentication scheme that further determines whether your application function is called. Both an asterisk (`*`) and `BASIC` are accepted values.

`/path/file`
Specifies the fully qualified file name of the compiled program, including the extension.

`function_name`
Specifies the name you give to the application function within your program.

Example

```
Authentication BASIC /ics/api/bin/icsextpgm.so:basic_authentication
```

Default

None

Authorization — Customize the Authorization step

Use this directive to specify a customized application function that the server calls during the Authorization step of the server request process. This code verifies that the requested object can be served to the client.

Format

`Authorization` *`request_template /path/file:function_name`*

`request_template`
Specifies a template for requests that further determine whether your application function is called. The specification can include the protocol, domain, and host; it can be preceded by a slash character (`/`) and can use an asterisk (`*`) as a wildcard. For example, `/front_page.html`, `http://www.ics.raleigh.ibm.com`, `/pub*`, `/*`, and `*` are all valid. The request template must start at the document root (`/`) when using Caching Proxy as a reverse proxy.

`/path/file`
Specifies the fully qualified file name of the compiled program, including the extension.

`function_name`
Specifies the name you give your application function within your program.

Example

```
Authorization /index.html /api/bin/icsextpgm.so:auth_url
```

Default
None

AutoCacheRefresh — Specify whether cache refreshing is to be used

Use this directive to turn cache refreshing on or off. If refreshing is turned on, the cache content is refreshed automatically. If refreshing is turned off, the cache agent is not invoked and all of its settings are ignored. If you are starting the cache agent by another method, for example, by using a **cron** job on Linux and UNIX systems, set this directive to `off`.

Format
```
AutoCacheRefresh {on | off}
```

Default
```
AutoCacheRefresh On
```

BindSpecific — Specify whether the server binds to one or all IP addresses

Use this directive on a multihomed system to specify whether the server listens on a single network address. If you set the value to `On`, the server binds to the IP address specified in the Hostname directive, instead of binding to all local IP addresses.

If this directive is not specified, the server binds to the default Hostname.

If you change this directive, you must manually stop and then start the server again. The server does not make the change if you only restart it. (See Chapter 5, "Starting and stopping the Caching Proxy," on page 15.)

Format
```
BindSpecific {on | off}  [OutgoingSrcIp ip_addr | host_name]
```

[OutgoingSrcIp *ip_addr* **|** *host_name***]**
The OutgoingSrcIp options allows Caching Proxy to use a specific source IP address when making outgoing connections. It is useful for Caching Proxy settings in DMZ and when special firewall rules requires this.

Default
```
BindSpecific Off
```

BlockSize — Specify the size of blocks in the cache

This directive specifies the size (in bytes) of blocks in the medium of the caching device. By default, the value is 8192. Because it is the only size supported, do not change the value. For more information, see the reference section for "htcformat command" on page 156.

Format
```
BlockSize size
```

Default
By default, there is no setting for BlockSize in the configuration file. (The default value is 8192.)

CacheAccessLog — Specify the path for the cache access log files

Use this directive to specify the path and file name where you want the server to store a log of accesses to the proxy cache. This directive is valid only if the server is running as a proxy. See "CacheRefreshTime — Specify when to start the cache agent" on page 183 for more information.

To enable logging of requests to the proxy cache, the Caching directive must be set to `ON`, and values must be set for the CacheMemory and the CacheAccessLog directives. Optionally, one or more cache devices can be defined by using the CacheDev directive.

The value of CacheAccessLog can be either an absolute path or a path relative to ServerRoot. (One example is shown of each.)

Format

```
CacheAccessLog  path/file
```

Examples

```
CacheAccessLog  /absolute/path/logfile
CacheAccessLog  /logs/logfile
```

Defaults

- **Linux and UNIX systems:** `CacheAccessLog /opt/ibm/edge/cp/server_root/logs/cache`
- **Windows systems:** `CacheAccessLog drive:\Program Files\IBM\edge\cp\logs\cache`

CacheAlgorithm — Specify the cache algorithm

Use this directive to specify the cache algorithm that the server uses during garbage collection.

Format

```
CacheAlgorithm  {bandwidth | responsetime | blend}
```

bandwidth
Attempts to maximize the saving of network bandwidth.

responsetime
Attempts to minimize user response time.

blend
Uses a balanced combination of `bandwidth` and `responsetime`.

Default

```
CacheAlgorithm bandwidth
```

CacheByIncomingUrl — Specify the basis for generating cache file names

Use this directive to specify whether the generated cache file names are based on the incoming URL of the request.

If this directive is set to `on`, the cache file names are generated based on the incoming URL. If this directive is set to `off`, the incoming URL is first passed through all applicable name translation plugins, MAP rules, and PROXY rules, and the generated cache file name is based on the resulting URL.

Note: When defining the cache filters, in a reverse proxy scenario for URL-based cache filters, use a format starting with a document root of / (forward slash). For example: `/test/index.html`. The format should *not* include a protocol, for example, *http://*.

Format

```
CacheByIncomingUrl {on | off}
```

Default

```
CacheByIncomingURL off
```

CacheClean — Specify how long to keep cached files

Use this directive to specify how long you want the server to keep cached files. When garbage collection runs, the server deletes cached files that have exceeded this time, regardless of the files' expiration date. Any time a file is requested that has been in the cache longer than the specified time, the server will revalidate the file to ensure it is valid before serving it.

Format

```
CacheClean  time_specification
```

Example

```
CacheClean 2 weeks
```

Default

```
CacheClean 1 month
```

CacheDefaultExpiry — Specify the default expiration time for files

Use this directive to set a default expiration time for files for which the server did not provide either an `Expires` or a `Last-Modified` header. Specify a URL template and the expiration time for files with URLs that match the template. Multiple occurrences of this directive can be included in the configuration file. Include a separate directive for each template. The URL template must include the protocol. Specify the time value in any combination of months, weeks, days, and hours.

Format

```
CacheDefaultExpiry URL_template expiration_time
```

Defaults

```
CacheDefaultExpiry ftp:*  1 day
CacheDefaultExpiry gopher:*  2 days
CacheDefaultExpiry http:*  0 days
```

Note: The default expiration for the HTTP protocol is `0 days`. Keeping this value is recommended because many script programs do not give an expiration date, yet their output expires immediately. A value other than `0` can cause clients to see out-of-date content.

CacheDev — Specify a storage device for the cache

Use this directive to specify a cache storage device. Either a file or a raw disk partition can be specified. On AIX platforms, a raw logical volume can be specified. (If a memory cache is not used, raw disk caching yields the best performance.)

Note that cache devices must be prepared before they are specified. To prepare a cache device, format it by using the **htcformat** command. For more information, see "htcformat command" on page 156.

Multiple cache devices can be specified. Each device is associated with the same CacheMemory and BlockSize values. However, each cache device incurs a memory overhead in the proxy server machine of about 8 MB. Fewer large devices are more efficient than more smaller devices. For best efficiency, use an entire disk as one large partition with nothing else on the disk. More information about cache storage is included in "Optimizing disk cache performance" on page 101.

Format

```
CacheDev  {raw_disk_partition | file}
```

Examples

AIX: `CacheDev /dev/rlv02`

HP-UX: `CacheDev /dev/rdsk/c1t15d0`

Linux: `CacheDev /opt/IBMWTE/filecache1`

Solaris: `CacheDev /dev/rdsk/c1t3d0s0`

Windows: `CacheDev \\.\E:`

Default

None

CacheExpiryCheck — Specify whether the server returns expired files

Use this directive to specify whether the server returns cached files that have expired. Set the value to `Off` if you want the server to be able to return expired files. Use the default value of `On` if you want the proxy to check with the origin server for a more recent version when a client requests an expired file. Generally, administrators do not want the server to return expired files except possibly when they are demonstrating the server and do not particularly care about the content being returned.

Format

```
CacheExpiryCheck {on | off}
```

Default

```
CacheExpiryCheck On
```

CacheFileSizeLimit — Specify the maximum size for files to be cached

Use this directive to specify the maximum size of files to be cached. Files larger than this size are not cached. The value can be specified in bytes (B), kilobytes (K), megabytes (M), or gigabytes (G). It does not matter whether the specification includes a space between the number and the unit of measurement (B, K, M, G).

Format

```
CacheFileSizeLimit maximum {B | K | M | G}
```

Default

```
CacheFileSizeLimit 4000 K
```

CacheLastModifiedFactor — Specify the value for determining expiration dates

Use this directive to specify the value to use for calculating expiration dates for specific URLs or for all the URLs that match a template.

HTTP servers frequently give the last-modified time for a file but not an expiration date. Similarly, FTP files can have a last-modified time stamp, but not an expiration date. The Caching Proxy calculates an expiration date for these files based on the last-modified time. It uses the last-modified time to determine the length of time since the file was modified and multiplies that by the value on a CacheLastModifiedFactor directive. The result of this calculation is the lifetime of the file, or the length of time before the file becomes stale.

You can also specify `off` or `-1` to turn the directive off and not calculate an expiration date. The proxy server reads the CacheLastModifiedFactor directives in the order they appear in the configuration file. It uses the first directive that it can apply to the cached file.

Format

`CacheLastModifiedFactor` *`url factor`*

`url`
Specifies the full URL, including the protocol, of the file being cached. You can use a URL template with asterisks (`*`) as wildcards, to apply a mask.

`factor`
Specifies the factor to use in the calculation. The values `off` or `-1` can also be specified.

Examples

```
CacheLastModifiedFactor  *://hosta/*    off
CacheLastModifiedFactor  ftp://hostb/*  0.30
CacheLastModifiedFactor  ftp://*        0.25
CacheLastModifiedFactor  http://*       0.10
CacheLastModifiedFactor  *              0.50
```

Defaults

```
CacheLastModifiedFactor http://*/ 0.10
CacheLastModifiedFactor http://*.htm* 0.20
CacheLastModifiedFactor http://*.gif 1.00
CacheLastModifiedFactor http://*.jpg 1.00
CacheLastModifiedFactor http://*.jpeg 1.00
CacheLastModifiedFactor http://*.png 1.00
CacheLastModifiedFactor http://*.tar 1.00
CacheLastModifiedFactor http://*.zip 1.00
CacheLastModifiedFactor http:* 0.15
CacheLastModifiedFactor ftp:*  0.50
CacheLastModifiedFactor     *  0.10
```

The default of 0.14 causes files modified a week ago to expire in one day.

CacheLocalDomain — Specify whether to cache the local domain

Use this directive to specify whether to cache URLs from hosts in the same domain as the proxy. Local sites on an intranet usually do not need to be cached because

the internal bandwidth is sufficient to load URLs quickly. Not caching local sites saves cache space for URLs that take longer to retrieve.

Format

```
CacheLocalDomain {on | off}
```

Default

```
CacheLocalDomain on
```

CacheMatchLanguage — Specify the language preference for the returned cache content

If the backend server has the capability of returning language variants to customers for the same URL, use this directive to support caching different languages for the same URL. The directive allows Caching Proxy to verify the language preference in the requests with the language of the cached response.

When CacheMatchLanguage is enabled, before Caching Proxy loads the cached content, it compares the language preference in your request's Accept-Language header with the language of the cached content. Caching Proxy also compares the preference distance. If a preference distance is less than a specified limit, it returns the cached copy; otherwise, the proxy forwards the request to the backend server to get a fresh copy in your requested language.

Format

```
CacheMatchLanguage {on | off}  lang-prefer-distance-limit special-id-for-all-lang
```

lang-prefer-distance-limit
Specify a value within the range of 0.001– 0.9999.

special-id-for-all-lang
Specify a language string returned from the server in the Content-Language header to inform the proxy that the response could be used for all language preferences.

Examples

The following is a configuration example of the directive, cache object, and the request.

```
CacheMatchLanguage On 0.2
```

If the cache object is simplified Chinese (zh_cn) and the request is:

```
GET / HTTP/1.1
...
Accept-Language: en_US;q=1.0, zh_cn;q=0.7, ja;q=0.3
....
```

For this request, the customer requests a page in English (with code and quality en_US/1.0), then simplified Chinese (with code and quality zh_cn/0.7), and then Japanese (its code and quality is ja/0.3). The cached object is in simplified Chinese. The preference distance between best expected quality and matched language quality is 1.0 - 0.7 = 0.3. Because the limit is set to 0.2 by the CacheMatchLanguage directive, and 0.3 is greater than the limit, the proxy asks the server for a new copy of that URL instead of returning the cached object.

If the server does not specify a language or does not specify special-id-for-all-lang in the Content-Language header when returning a response, when the next request comes in, the proxy does not match language preference and returns the cached copy.

Default

```
CacheMatchLanguage off
```

CacheMaxExpiry — Specify the maximum lifetime for cached files

Use this directive to define the maximum length of time files can stay in the cache. The lifetime of a cached file defines the length of time it can be served from the cache without checking the origin for updates. In some cases, the computed lifetime for a cached file can be longer than you want to keep the file. The lifetime of the file, either specified by the origin or computed by the Caching Proxy, cannot exceed the limit specified by the CacheMaxExpiry directive.

Multiple occurrences of this directive are allowed in the configuration file. Include a separate directive for each template.

Format

```
CacheMaxExpiry URL lifetime
```

URL
Specifies a fully specified URL, including the protocol, of the file being cached. You can use a URL template with asterisks (*) as wildcards, to apply a mask.

lifetime
Specifies the maximum lifetime for cached files matching the URL template. The time can be specified in any combination of months, weeks, days, hours, minutes, or seconds.

Examples

```
CacheMaxExpiry ftp:* 1 month
CacheMaxExpiry http://www.santaclaus.np/* 2 days 12 hours
```

Default

```
CacheMaxExpiry 1 month
```

CacheMemory — Specify the cache RAM

Use this directive to specify the amount of memory to associate with the cache. For optimum performance of disk caches, a minimum cache memory value of 64 MB is recommended for cache infracture support, including the cache index. As the cache size increases, the cache index increases, and more cache memory is required to store the index. A cache memory value of 64 MB is large enough to provide cache infrastructure support and store a cache index for a disk cache of up to approximately 6.4 GB. For larger disk caches, the cache memory should be 1% of the cache size.

If memory caching is used, set this directive to include both the cache and the amount of memory needed for the cache index.

The maximum recommended value for this directive is 1600 MB. This limit is determined by the fact that the Caching Proxy, as a 32-bit application, can use a maximum of 2 GB of memory. If the amount of memory required for the cache plus the amount of memory used for routine processing approaches or exceeds 2 GB, the Caching Proxy does not operate normally.

The amount can be specified in one of the following units: bytes (B), kilobytes (K), megabytes (M), and gigabytes (G).

Format

```
CacheMemory amount {B | K | M | G}
```

Default

```
CacheMemory 64 M
```

CacheMinHold — Specify how long to keep files available

Use this directive to specify URLs for files whose expiration is to be overridden. Some sites set files to expire before the end of their lifetime, requiring the server to request the file more frequently. The CacheMinHold directive causes the expired file to be held in the cache for the specified amount of time before it is requested again. This directive can be specified multiple times.

Note: If expiration dates are overridden, the files in the cache can become obsolete or out-of-date.

Example

```
CacheMinHold http://www.cachebusters.com/* 1 hour
```

Default

None

CacheNoConnect — Specify the stand-alone cache mode

Use this directive to specify whether the proxy server retrieves files from remote servers. The default value (`Off`) enables the server to retrieve files from remote servers. The value `On` sets the server to run in stand-alone cache mode. This means that the server can return only files already stored in its cache. Typically, when running the server in this mode, you also set the CacheExpiryCheck directive to `Off`.

Running the server in stand-alone cache mode can be useful if you are using the server for demonstrations. If you know that all the files you want to use for a demonstration are stored in the cache, then you do not need a network connection.

Format

```
CacheNoConnect  {on | off}
```

Default

```
CacheNoConnect Off
```

CacheOnly — Cache only the files with URLs that match a template

Use this directive to specify that only files with URLs that match a specified template are to be cached. You can use multiple occurrences of this directive in the configuration file. Include a separate directive for each template. The URL template must include the protocol. If no value is set for this directive, any URLs that do not match a NoCaching directive can be cached. If neither the CacheOnly nor the NoCaching directive is included in the configuration file, any URL can be cached.

Format

```
CacheOnly  url_pattern
```

Example

```
CacheOnly http://realstuff/*
```

Default
None

CacheQueries — Specify cache responses to URLs containing a question mark (?)

Use this directive to specify the URLs for which responses to query requests are cached. If the value `PUBLIC` *`url_pattern`* is used, responses to GET requests that contain a question mark in the URL are cached if the origin server includes the `cache-control: public` header and the response is otherwise cacheable. If the value `ALWAYS` *`url_pattern`* is specified, responses to GET requests that contain a question mark in the URL are cached if the responses are otherwise cacheable.

This directive can be specified multiple times.

```
CacheQueries {ALWAYS | PUBLIC} url_pattern
```

Examples

```
CacheQueries ALWAYS http://www.hosta.com/*
CacheQueries PUBLIC http://www.hostb.com/*
```

Note: For backward compatibility, the previous syntax of `CacheQueries {ALWAYS | PUBLIC | NEVER}` is treated as follows:

- `CacheQueries ALWAYS` and `CacheQueries PUBLIC` are treated as `CacheQueries ALWAYS * and CacheQueries PUBLIC *`.
- `CacheQueries NEVER` is ignored.
- If both `CacheQueries NEVER` and `CacheQueries` *`url_pattern`* are specified, then `CacheQueries NEVER` is ignored, but a warning message is issued.

Default
None

CacheRefreshInterval — Specify the time interval for revalidating cached objects

Use this directive to specify when to check with the origin server to determine whether a cached file is changed.

Although the CacheClean directive appears to be similar to this directive, there is a difference. CacheRefreshInterval specifies only that the proxy revalidates a file before using it, whereas the CacheClean directive causes the file to be removed from the cache after a specified period of time.

Format

- The following format specifies the refresh interval for any files that match the URL pattern:

  ```
  CacheRefreshInterval URL_pattern time_period
  ```

- The following format specifies the refresh interval for any files that do *not* match a URL pattern. Only a refresh interval is specified.

  ```
  CacheRefreshInterval time_period
  ```

Examples

```
CacheRefreshInterval *.gif 8 hours
CacheRefreshInterval 1 week
```

Default

```
CacheRefreshInterval 2 weeks
```

CacheRefreshTime — Specify when to start the cache agent

Use this directive to specify when to start the cache agent. You can start the cache agent at a specific time.

Format

```
CacheRefreshTime HH:MM
```

Default

```
CacheRefreshTime 03:00
```

CacheTimeMargin — Specify the minimum lifetime for caching a file

The CacheTimeMargin directive specifies the minimum lifetime of a file that is required in order for it to be cached.

The Caching Proxy computes an expiration time for each file. If it is unlikely that another request for the file will be received before the file expires, the Caching Proxy considers the lifetime of the file to be too short for the file to be cached. By default, the Caching Proxy does not cache files whose lifetime is less than 10 minutes. If your cache is not close to its maximum capacity, leave this directive at its initial value. If your cache is filled close to capacity, consider increasing the value for the minimum lifetime.

Format

```
CacheTimeMargin minimum_lifetime
```

Default

```
CacheTimeMargin 10 minutes
```

Note: Setting this directive to more than four hours dramatically reduces the efficiency of the cache.

CacheUnused — Specify how long to keep unused cached files

Use this directive to set the maximum length of time for the server to keep unused cached files with URLs that match a specified template. The server deletes unused files with URLs matching the template after they are cached for the specified time, regardless of their expiration date. You can include multiple occurrences of this directive in the configuration file. Include a separate directive for each template. The URL template must include the protocol. Specify the time value in any combination of months, weeks, days, and hours.

Format

```
CacheUnused url_template time_length
```

Examples

```
CacheUnused ftp:* 3 weeks
CacheUnused gopher:* 3 days 12 hours
CacheUnused * 4 weeks
```

Defaults

```
CacheUnused ftp:* 3 days
CacheUnused gopher:* 12 hours
CacheUnused http:* 2 days
```

Caching — Enable proxy caching

Use this directive to enable the caching of files. With caching turned on, the proxy server stores the files it retrieves from other servers in a local cache. The proxy server then responds to subsequent requests for the same files without needing to retrieve them from other servers.

Format

```
Caching {on | off}
```

Default

```
Caching On
```

Note: If you change the Caching directive, you must manually stop and then start the server. (See Chapter 5, "Starting and stopping the Caching Proxy," on page 15.)

CdfAware — Designate this instance of Caching Proxy as part of a Content Distribution Framework

Use this directive to specify whether the Caching Proxy is part of a Content Distribution Framework.

Format

```
CdfAware {yes | no}
```

Default

```
CdfAware no
```

CdfRestartFile — Specify the file to store a file name to url mapping

Use this directive to designate the name of the file that stores filename to URL mapping data so that it is persistent over multiple instances of ibmproxy in a Content Distribution Framework. The Caching Proxy maintains a mapping table that associates a requested URL with its file name on the Web server. This file provides persistent storage of this table across restarts. Use this directive only if the CdfAware directive is set to yes.

Format

CdfRestartFile *path/filename*

Example

- **Linux and UNIX:** `CdfRestartFile /opt/ibm/edge/cd/cdfRestartFile`
- **Windows:** `CdfRestartFile C:\progra~1\ibm\edge\cd\cdfRestartFile.txt`

Default

None

CompressAge — Specify when to compress logs

Use this directive to specify the age after which logs are compressed. When the logs are older than the value set for CompressAge, they are compressed. If CompressAge is set to `0`, no logs are ever compressed. The logs for the current and previous days are never compressed.

Format

`CompressAge` *`number_of_days`*

Default

```
CompressAge 1
```

Related directives

- "CompressDeleteAge — Specify when to delete logs" on page 186
- "CompressCommand — Specify the compression command and parameters"
- "LogArchive — Specify the behavior of log archiving" on page 215
- "Midnight — Specify the API plugin used to archive logs" on page 221
- "PurgeAge — Specify the age limit for a log" on page 243
- "PurgeSize — Specify the limit for the size of the log archive" on page 243

CompressCommand — Specify the compression command and parameters

Use this directive to build a command that identifies the compression utility used to compact the logs and that passes parameters to that utility. Include the path for the archived logs.

The compression utility must be installed in a directory listed in the path for that machine.

Format

`CompressCommand` *`command`*

`command`
Includes the command and parameters you want to use, entered on a single line. Typically, parameters include `%%LOGFILES%%` and `%%DATE%%`.

`%%LOGFILES%%`
Specifies the list of log files that are available for a particular `%%DATE%%`.

`%%DATE%%`
Specifies the date stamp on a log file.

Examples

- **Linux and UNIX:**

```
CompressCommand tar -cf /logarchs/log%%DATE%%.tar   %%LOGFILES%% ;
                    gzip /logarchs/log%%DATE%%.tar
CompressCommand tar -cf /logarchs/log%%DATE%%.tar   %%LOGFILES%% ;
                    compress /logarchs/log%%DATE%%.tar
CompressCommand zip -q /logarchs/log%%DATE%%.zip   %%LOGFILES%%
```

Note: The command and all parameters must be entered on a single line. In the preceding examples, the first two command examples are divided for readability.

- **Windows:**

```
CompressCommand pkzip -q d:\logarchs\log%%DATE%%.tar  %%LOGFILES%%
```

Default
None

Related directives
- "CompressAge — Specify when to compress logs" on page 185
- "CompressDeleteAge — Specify when to delete logs"
- "LogArchive — Specify the behavior of log archiving" on page 215
- "Midnight — Specify the API plugin used to archive logs" on page 221
- "PurgeAge — Specify the age limit for a log" on page 243
- "PurgeSize — Specify the limit for the size of the log archive" on page 243

CompressDeleteAge — Specify when to delete logs

Use this directive to specify when to delete a log after it has been compressed. When a log is older than the number of days set for the value of CompressDeleteAge, it is deleted. If CompressDeleteAge is set to `0`, or if its value is less than the value set for the CompressAge directive, a log is not deleted.

Note: The compression plugin never deletes logs for the current day or the preceding day.

Format
`CompressDeleteAge` *`number_of_days`*

Default
```
CompressDeleteAge 7
```

Related directives
- "CompressAge — Specify when to compress logs" on page 185
- "CompressCommand — Specify the compression command and parameters" on page 185
- "LogArchive — Specify the behavior of log archiving" on page 215
- "Midnight — Specify the API plugin used to archive logs" on page 221
- "PurgeAge — Specify the age limit for a log" on page 243
- "PurgeSize — Specify the limit for the size of the log archive" on page 243

ConfigFile — Specify the name of an additional configuration file

Use this directive to specify the name and location of an additional configuration file. Directives found in the specified configuration file are processed after the current configuration file.

Note: Ensure that the additional configuration file has its permission set to `Read` for user `nobody` to allow the cache agent to read this file.

Examples
- **Linux and UNIX:** `ConfigFile /etc/rca.conf`
- **Windows:** `ConfigFile c:\WINNT\rca.conf`

Default
None

ConnThreads — Specify the number of connection threads to be used for connection management

Use this directive to defines the number of connection threads to be used for connection management.

Format

```
ConnThreads number
```

Default

```
ConnThreads 5
```

Related directives

- "MaxActiveThreads — Specify the maximum number of active threads" on page 218

ContinueCaching — Specify how much of a file is required for caching

Use this directive to specify how much of a requested file must be transferred for the Caching Proxy to complete the creation of the cache file, even though the client connection is terminated. Valid values for this variable are integers in the range of 0 – 100.

For example, if `ContinueCaching 75` is specified, the Caching Proxy continues transferring the file from the content server and generates the cache file if 75% or more of the file is already transferred before the Caching Proxy detects that the client connection is terminated.

Format

```
ContinueCaching percentage
```

Default

```
ContinueCaching 75
```

DefinePicsRule — Supply a content-filtering rule

Use this directive to supply the proxy with the necessary information to filter URLs for content including rating service information. You can specify this directive multiple times.

Format

```
DefinePicsRule "filter_name"  {
```

Default

```
DefinePicsRule "RSAC Example" {
```

DefProt — Specify default protection setup for requests that match a template

Use this directive to associate a default protection setup with requests that match a template.

Note: For protection to work properly, the DefProt and Protect directives must be placed before any Pass or Exec directives in the configuration file.

Format

```
DefProt request_template  setup_name [FOR server_IP_address | host_name]
```

request_template
Specifies a template for requests that you want to associate with a default protection setup. The server compares incoming client requests to the template and associates a protection setup if there is a match.

Protection is not actually activated for requests matching the template unless the request also matches a template on a subsequent Protect directive. See "Protect — Activate a protection setup for requests that match a template" on page 231 for an explanation of how the Protect directive is used with DefProt.

setup
The named protection setup that is defined in the configuration file and that you want to associate with requests that match *request_template*. The protection setup is defined with protection subdirectives. This parameter can take one of three forms:

- A full path and file name specifying a separate file that contains the protection subdirectives.
- A protection setup label name that matches a name defined previously on a Protection directive. The Protection directive contains the protection subdirectives.
- The actual protection subdirectives. The subdirectives must be enclosed in braces ({}). The left brace character must be the last character on the same line as the DefProt directive. Each subdirective follows on its own line. The right brace character must be on its own line following the last subdirective line. No comment lines can appear between the braces. For descriptions of the protection subdirectives, see:
  - "AuthType — Specify the authentication type" on page 236
  - "DeleteMask — Specify the user names, groups, and addresses that are allowed to delete files" on page 236
  - "GetMask — Specify the user names, groups, and addresses allowed to get files" on page 236
  - "GroupFile — Specify the location of the associated group file" on page 237
  - "Mask — Specify the user names, groups, and addresses allowed to make HTTP requests" on page 237
  - "PasswdFile — Specify the location of the associated password file" on page 237
  - "PostMask — Specify the user names, groups, and addresses allowed to post files" on page 237
  - "PutMask — Specify the users names, groups, and addresses allowed to put files" on page 238
  - "ServerID — Specify a name to associate with the password file" on page 238

[FOR *Server_IP_address* | *host_name***]**
If you are using multiple IP addresses or virtual hosts, use this parameter to specify an IP address or host name. The server uses the directive only for requests that come to the server on this IP address or for this host. For an IP address, this is the address of the server's network connection, not the address of the requesting client.

You can specify an IP address (for example, `FOR 240.146.167.72`) or you can specify a host name (for example, `FOR hostA.bcd.com`).

This parameter is optional. Without this parameter, the server uses the directive for all requests regardless of the IP address on which the requests come in or the host name in the URL.

Notes:

1. You can use this parameter only with the *setup* parameter specified in the form of a path and file name or a protection setup label. You cannot use this parameter with the *setup* parameter specified in the form of actual protection subdirectives enclosed in braces.
2. To use this parameter, you must put FOR, or some other character string (without blanks), between the *setup* parameter and the *IP_address* or *host_name*.

A wildcard character cannot be specified for a server's IP address.

Note: The directive must be typed on one line.

Examples

- The following example identifies a separate file that contains the protection subdirectives.

```
DefProt   /secret/* /server/protect/setup1.acc
```

- The following example uses a label name to point to the protection subdirectives. The label name must match a label name on a Protection directive. The Protection directive must come before the DefProt directive.

```
DefProt   /secret/*   SECRET-PROT
```

- The following example includes the protection subdirectives as part of the DefProt directive.

```
DefProt   {
  AuthType Basic
  ServerID restricted
  PasswdFile /docs/etc/WWW/restrict.password
  GroupFile /docs/etc/WWW/restrict.group
  GetMask authors
  PutMask authors
  }
```

- The following examples use the optional IP address parameter. If your server receives requests that begin with /secret/, it associates a different default protection setup with the request based on the IP address of the network connection the request comes in on. For requests coming in on 0.67.106.79, the server associates the request with default protection defined on a Protection directive with a label of CustomerA-PROT. For requests coming in on 0.83.100.45, the server associates the request with default protection defined on a Protection directive with a label of CustomerB-PROT.

```
DefProt   /secret/*   CustomerA-PROT   0.67.106.79
DefProt   /secret/*   CustomerB-PROT   0.83.100.45
```

- The following examples use the optional host name parameter. If your server receives requests that begin with /secret/, it associates a different default protection setup with the request based on the host name in the URL. For requests coming in for hostA, the server associates the request with default protection defined on a Protection directive with a label of CustomerA-PROT. For requests coming in for hostB, the server associates the request with default protection defined on a Protection directive with a label of CustomerB-PROT.

```
DefProt   /secret/*   CustomerA-PROT   hostA.bcd.com
DefProt   /secret/*   CustomerB-PROT   hostB.bcd.com
```

Default
None

DelayPeriod — Specify pausing between requests

Use this directive to specify whether the cache agent waits between sending requests to destination servers. Specifying a delay between requests reduces the load on the proxy machine and your network link, as well as on the destination servers. Specifying no delay lets the cache agent run at maximum speed. For slow Internet connections, consider specifying no delay period in order to achieve maximum use of your network.

Note: If your connection to the Internet is faster than 128 kbps, set DelayPeriod to `On` to avoid sending too many requests too rapidly to sites being refreshed.

Format
```
DelayPeriod {on | off}
```

Default
```
DelayPeriod On
```

DelveAcrossHosts — Specify caching across domains

Use this directive to specify whether the cache agent follows hypertext links across hosts. If a cached URL contains links to other servers, the server can ignore the link or follow it. If the directive DelveInto is set to `never`, this directive is not applied.

Format
```
DelveAcrossHosts {on | off}
```

Default
```
DelveAcrossHosts Off
```

DelveDepth — Specify how far to follow links while caching

Use this directive to specify the number of link levels to follow when searching for pages to load into the cache. If the directive DelveInto is set to `never`, this directive is not applied.

Format
DelveDepth *number_of_levels*

Default
```
DelveDepth 2
```

DelveInto — Specify whether the cache agent follows links

Use this directive to specify whether the cache agent loads pages linked from cached URLs.

Format
```
DelveInto {always | never | admin | topn}
```

always
The cache agent follows links from all previously cached URLs.

never
The cache agent ignores all links on URLs.

admin
The cache agent follows links only on URLs specified in the LoadURL directives

topn
The cache agent follows links only from the most-frequently retrieved files in the cache.

Default
`DelveInto always`

DirBackgroundImage — Specify a background image to directory listings

Use this directive to apply a background image to directory listings generated by the proxy server. Directory listings are generated when the proxy server is used to browse FTP sites.

Specify an absolute path for the background image. If the image is located on another server, the background image must be specified as a full URL. If no background image is specified, a plain white background is used.

Format
`DirBackgroundImage /path/file`

Examples
```
DirBackgroundImage /images/corplogo.png
DirBackgroundimage http://www.somehost.com/graphics/embossed.gif
```

Default
None

DirShowBytes — Show byte count for small files on directory listings

Use this directive to specify whether directory listings include the exact byte count for files smaller than 1 KB. A value of `Off` means the directory listing shows a size of 1 KB for all files that are 1 KB or smaller.

Format
`DirShowBytes {on | off}`

Default
`DirShowBytes Off`

DirShowCase — Use case when sorting files on directory listings

Use this directive to specify whether directory listings distinguish between uppercase and lowercase letters when sorting file names.

A value of `On` means that uppercase letters are placed before lowercase letters in the list of files.

Format
`DirShowCase {on | off}`

Default

```
DirShowCase On
```

DirShowDate — Show the date of last modification on directory listings

Use this directive to specify whether directory listings include the date each file was last modified.

Format

```
DirShowDate {on | off}
```

Default

```
DirShowDate On
```

DirShowDescription — Show descriptions for files on directory listings

Use this directive to specify whether directory listings include descriptions for HTML files. The descriptions are taken from the files' HTML `<title>` tags.

Descriptions for FTP directory listings show the MIME types for files if they can be determined.

Format

```
DirShowDescription {on | off}
```

Default

```
DirShowDescription On
```

DirShowHidden — Show hidden files on directory listings

Use this directive to specify whether directory listings include any hidden files in the directory. The server considers any file that has a name beginning with a period (.) to be a hidden file.

Format

```
DirShowHidden {on | off}
```

Default

```
DirShowHidden On
```

DirShowIcons — Show icons in directory listings

Use this directive to specify whether the server includes icons in directory listings. Icons can be used to provide a graphic representation of the content type of the files in the listing. The icons themselves are defined by the AddBlankIcon, AddDirIcon, AddIcon, AddParentIcon, and AddUnknownIcon directives.

Format

```
DirShowIcons {on | off}
```

Default

```
DirShowIcons On
```

DirShowMaxDescrLength — Specify the maximum length for descriptions on directory listings

Use this directive to set the maximum number of characters to show in the description field on directory listings.

Format

```
DirShowMaxDescrLength number_of_characters
```

Default

```
DirShowMaxDescrLength 25
```

DirShowMaxLength — Specify the maximum length for file names on directory listings

Use this directive to set the maximum number of characters that are used for file names on directory listings.

Format

```
DirShowMaxDescrLength number_of_characters
```

Default

```
DirShowMaxLength 25
```

DirShowMinLength — Specify the minimum length for file names on directory listings

Use this directive to set the minimum number of characters that are always reserved for file names on directory listings. File names in the directory can exceed this number. However, file names cannot be longer than the number specified on the DirShowMaxLength directive.

Format

```
DirShowMinLength number_of_characters
```

Default

```
DirShowMinLength 15
```

DirShowSize — Show the file size on directory listings

Use this directive to specify whether directory listings include the size of each file.

Format

```
DirShowSize {on | off}
```

Default

```
DirShowSize On
```

Disable — Disable HTTP methods

Use this directive to specify which HTTP methods the server does not accept. For each method that the server is to reject, enter a separate Disable directive.

In the default configuration file, the GET, HEAD, OPTIONS, POST, and TRACE methods are enabled and all other supported HTTP methods are disabled. To disable a method that is currently enabled, delete it from the Enable directive and add it to the Disable directive.

Format

```
Disable method
```

Note: The Configuration and Administration forms use the POST method to make updates to the server configuration. If you disable the POST method, you will not be able to use the Configuration and Administration forms.

Defaults

```
Disable   PUT
Disable   DELETE
Disable   CONNECT
```

DisInheritEnv — Specify the environment variables that are disinherited by CGI programs

Use this directive to specify which environment variables you do not want your CGI programs to inherit (other than the CGI environment variables that are specific to CGI processing).

By default, all environment variables are inherited by CGI programs. Use this directive to exclude individual environment variables from being inherited.

Format

```
DisInheritEnv environment_variable
```

Examples

```
DisInheritEnv PATH
DisInheritEnv LANG
```

In this example, all environment variables except PATH and LANG are inherited by CGI programs.

Default

None

DNS-Lookup — Specify whether the server looks up client host names

Use this directive to specify whether the server looks up the host names of requesting clients.

Format

```
DNS-Lookup {on | off}
```

The value you use affects the following things about how your server works:

- The performance of the server. Using the default value of `Off` improves the performance and response time of the server because it does not use resources to perform the host name lookup.
- The information your server records about clients when writing to log files.
  `Off`—Clients are identified by IP address.
  `On`—Clients are identified by host name.
- Whether you can use host names on address templates in protection setups, server group files, and access control list (ACL) files.
  `Off`—You cannot use host names on address templates; you must use IP addresses.

On—You can use host names on address templates; you cannot use IP addresses.

Note: To use domain names in your protection rules, you must set the DNS-Lookup directive to On.

Default

```
DNS-Lookup   Off
```

Enable — Enable HTTP methods

Use this directive to specify which HTTP methods the server accepts.

You can enable as many of the HTTP methods as you need. For each method the server is to accept, enter a separate Enable directive.

Format

```
Enable method
```

If no Service directive exists for a particular URL, you can use the Enable directive to perform customized programming for any HTTP method. The program you specify on this directive overrides the standard processing for that method.

```
Enable method /path/fileDLL:function_name
```

Defaults

```
Enable GET
Enable HEAD
Enable POST
Enable TRACE
Enable OPTIONS
```

EnableTcpNodelay — Enable TCP NODELAY socket option

Use this directive to enable the TCP NODELAY socket option.

The EnableTcpNodelay directive improves performance when small IP packets, such as an SSL handshake or a short HTTP response, are transmitted between Caching Proxy and the client. By default, the TCP NODELAY option is enabled for all sockets.

Format

```
EnableTcpNodelay {All | HTTP | HTTPS | None}
```

Default

```
EnableTcpNodelay  All
```

Error — Customize the Error step

Use this directive to specify a customized application function that you want the server to call during the Error step. This code is executed to provide customized error routines when an error is encountered.

Format

```
Error  request_template /path/file:function_name
```

request_template
Specifies a template for requests that further determine whether your application function is called. The specification can include the protocol, domain, and host; it can be preceded by a slash (/), and it can use an asterisk

(*) as a wildcard. For example, /front_page.html, http://www.ics.raleigh.ibm.com, /pub*, /*, and * are all valid.

/path/file
Specifies the fully qualified file name of the compiled program, including the extension.

function_name
Specifies the name of the application function within your program.

Example

```
Error  /index.html /ics/api/bin/icsext05.so:error_rtns
```

Default

None

ErrorLog — Specify the file where server errors are logged

Use this directive to specify the path and file name where you want the server to log internal errors.

Note: If you change the server defaults for the user ID, group ID, or log directory paths, create the new directories and update the permissions and ownership of the directories. To enable the server to write information to a user-defined log directory, set the permission for that directory as 755, and set the user-defined server user ID as the owner. For example, if you change the user ID of the server from the default to jdoe, and you change the default logs directory to server_root/account, then the server_root/account directory must have the permission 755 and be owned by jdoe.

If it is running, the server starts a new log file each day at midnight. Otherwise, the server starts a new log file the first time it is started on any day. When creating the file, the server uses the file name you specify and appends a date suffix. The date suffix is in the format *Mmmddyyyy*, where *Mmm* represents the first three letters of the month; *dd* represents the day of the month; and *yyyy* represents the year.

Format

```
ErrorLog  /path/logs_directory/file_name
```

Defaults

- **Linux and UNIX systems:** `ErrorLog /opt/ibm/edge/cp/server_root/logs/error`
- **Windows systems:** `ErrorLog drive:\Program Files\IBM\edge\cp\logs\error`

ErrorPage — Specify a customized message for a particular error condition

Use this directive to specify the name of a file that is sent to the requesting client when the server encounters a particular error condition. The configuration file ibmproxy.conf provides ErrorPage directives that associate the error keywords with error message files.

To customize error messages, you can modify the ErrorPage directives to associate the error keywords with different files, or you can modify the provided error message files. For example, you can change a message to include more information about the cause of the problem and suggest possible solutions to fix it. For internal networks, you might provide a contact person for your users to call.

ErrorPage directives can be placed anywhere in the configuration file. When the error occurs, the file is processed according to the mapping rules defined in your configuration file. Therefore, the file you want to send must be in a location that can be reached through the mapping rules as defined by the Fail, Map, NameTrans, Pass, Redirect, and Service directives. At a minimum, you need a Pass directive that allows the server to pass the error message file.

Format

`ErrorPage` *`keyword /path/filename.html`*

keyword
Specifies one of the keywords associated with an error condition. Keywords are listed in the ErrorPage directives in the file ibmproxy.conf. You cannot change the keywords.

/path/filename.html
Specifies the fully qualified Web name of your error file, as viewed by a client on the Web. Default error message files are in /HTML/errorpages/.

Example

```
ErrorPage scriptstart /HTML/errorpages/scriptstart.htmls
```

In this example, when a `scriptstart` condition is encountered, the server sends the scriptstart.htmls file found in the /HTML/errorpages/ directory to the client.

The following HTML text is an example what the file might contain:

```
<HTML>
<HEAD>
<TITLE>Message for SCRIPTSTART condition</TITLE>
</HEAD>
<BODY>
The CGI program could not be started.
<P>
<A HREF="mailto:admin@websvr.com">Notify the administrator</A>
of this problem.
</BODY>
</HTML>
```

If the directive that matches the above path in the server's configuration file is `PASS /* /wwwhome/*`, then the full path for this message file is `/wwwhome/HTML/errorpages/scriptstart.htmls`.

Customizing the error messages that the server returns

Each error condition is identified by a keyword. To decide which error messages you want to customize, first review the error message files provided with the Caching Proxy, which you can find in /HTML/errorpages. The error page includes the error number, the default message, an explanation of the cause, and an appropriate recovery action.

Then, do one of the following to change an error message:

- Modify the existing HTML or HTMLS file (create a backup copy first) or create a new HTML or HTMLS file with the desired text. You can use an HTML editor or an ASCII editor. An HTMLS file must be used if you want to use server-side includes.
- If you created an error message file with a different name (or in a different path), modify the ErrorPage directive for that keyword to point to the file.

Error conditions, causes, and default messages

All error keywords and default error message files are listed in the file ibmproxy.conf in the ErrorPage directive section. The error message files include the error message number, keyword, default message, explanation, and user response (action).

Defaults

Numerous defaults are included in the file ibmproxy.conf

If you do not modify an ErrorPage directive for an error condition, the server's default error page for that condition is sent.

EventLog — Specify the path for the event log file

Use this directive to specify the event log path and file name. The event log captures informational messages about the cache itself.

Note: If you change the server defaults for the user ID, group ID, or log directory paths, create the new directories and update the permissions and ownership of the directories. To enable the server to write information to a user-defined log directory, set the permission for that directory as 755, and set the user-defined server user ID as the owner. For example, if you change the user ID of the server from the default to jdoe and you change the default logs directory to server_root/account, then the server_root/account directory must have the permission 755 and be owned by jdoe.

If it is running, the server starts a new log file each day at midnight . Otherwise, the server starts a new log file the first time it is started on any day. When creating the file, the server uses the file name you specify and appends a date suffix. The date suffix is in the format *Mmmddyyyy*, where *Mmm* represents the first three letters of the month; *dd* represents the day of the month; and *yyyy* represents the year.

Format

`EventLog` */path/logs_directory/file_name*

Defaults

- **Linux and UNIX systems:** `EventLog /opt/ibm/edge/cp/server_root/logs/event`
- **Windows systems:** `EventLog` *drive*`:\Program Files\IBM\edge\cp\logs\event`

Exec — Run a CGI program for matching requests

Use this directive to specify a template for requests to accept and respond to by running a CGI program. After a request matches a template on an Exec directive, the request is not compared to request templates on any subsequent directives.

Format

`Exec` *request_template program_path* [*Server_IP_address* | *host_name*]

request_template
Specifies a template for requests that the server is to accept and respond to by running a CGI program.

You must use an asterisk (*) as a wildcard in both the `request_template` and `program_path`. The part of the request that matches the `request_template` wildcard must begin with the name of the file that contains the CGI program.

The request can also contain additional data that is passed to the CGI program in the PATH_INFO environment variable. The additional data follows the first

slash character (/) that comes after the CGI program file name on the request. The data is passed according to CGI specifications.

program_path
Specifies the path to the file that contains the CGI program that the server executes for the request. *program_path* must also contain a wildcard. The wildcard is replaced with the name of the file that contains the CGI program.

The Exec directive is recursive and applies to all subdirectories. You do not need a separate Exec directive for each directory under cgi-bin and admin-bin.

[*Server_IP_address* | *host_name*]
If you are using multiple IP addresses or virtual hosts, use this parameter to specify an IP address or a host name. The server uses the directive only for requests that come to the server on this IP address or for this host. For an IP address, this is the address of the server's network connection, not the address of the requesting client.

You can specify an IP address (for example, `240.146.167.72`) or you can specify a host name (for example, `hostA.bcd.com`).

This parameter is optional. Without this parameter, the server uses the directive for all requests regardless of the IP address the requests come in on or the host name in the URL.

Wildcard characters cannot be used to specify server IP addresses.

Examples

In the following example, if the server receives a request of `/idd/depts/plan/c92`, it runs the CGI program in /depts/bin/plan.exe with `c92` passed to the program as input.

The following example uses the optional IP address parameter. If your server receives requests that begin with `/cgi-bin/`, it serves the request from a different directory based on the IP address of the network connection on which the request comes in. For requests coming in on 130.146.167.72, the server uses the /CGI-BIN/customerA directory. For requests coming in on any connection with an address of 0.83.100.45, the server uses the /CGI-BIN/customerB directory.

```
Exec    /cgi-bin/*    /CGI-BIN/customerA/*   130.129.167.72
Exec    /cgi-bin/*    /CGI-BIN/customerB/*   0.83.100.45
```

The following example uses the optional host name parameter. If your server receives requests that begin with `/cgi-bin`, it serves the request from a different directory based on the host name in the URL. For requests coming in for hostA.bcd.com, the server uses the /CGI-BIN/customerA directory. For requests coming in for hostB.bcd.com, the server uses the /CGI-BIN/customerB directory.

```
Exec    /cgi-bin/*    /CGI-BIN/customerA/*   hostA.bcd.com
Exec    /cgi-bin/*    /CGI-BIN/customerB/*   hostB.bcd.com
```

Defaults

- **Linux and UNIX systems**

```
Exec  /cgi-bin/*      /opt/ibm/edge/cp/server_root/cgi-bin/*
Exec  /admin-bin/*    /opt/ibm/edge/cp/server_root/admin-bin/*
```

- **Windows systems**

```
Exec  server_root/cgi-bin/*
Exec  server_root/admin-bin/*
Exec  server_root/DOCS/admin-bin/*
```

ExportCacheImageTo — Export cache memory to disk

Use this directive to export the cache contents to a dump file. This is useful when memory cache gets lost during restart, or when deploying the same cache for multiple proxies.

Format

```
ExportCacheImageTo export_file_name
```

Default

None

ExternalCacheManager — Configure the Caching Proxy for dynamic caching from IBM WebSphere Application Server

Use this directive to configure the Caching Proxy to recognize a IBM WebSphere Application Server (that is configured with a Caching Proxy adapter module) from which it can cache dynamically created resources. The Caching Proxy saves copies of JSP results that also are stored in the application server's dynamic cache. The Caching Proxy caches only contents from a IBM WebSphere Application Server whose group ID matches an ExternalCacheManager entry.

Note that it is also necessary to add a Service directive to the Caching Proxy configuration file to enable this feature. Additional configuration steps also are necessary at the application server. Refer to Chapter 22, "Caching dynamically generated content," on page 97 for complete information.

Format

```
ExternalCacheManager External_Cache_Manager_ID  Maximum_Expiry_Time
```

External_Cache_Manager_ID
: The ID assigned to the IBM WebSphere Application Server that is serving the proxy. The ID must match the ID set in the `externalCacheGroup: group id` attribute in the dynacache.xml file on the application server.

Maximum_Expiry_Time
: The default expiration time set for resources cached on behalf of the external cache manager. If the external cache manager does not invalidate a cached resource within the specified time, the resource expires in the specified time. The time can be specified in minutes or seconds.

Example

The following entry defines an external cache manager (a IBM WebSphere Application Server) that is within the www.xyz.com domain and whose resources expire in 20 seconds or earlier.

```
ExternalCacheManager   IBM-CP-XYZ-1  20 seconds
```

Default

None

Fail — Reject matching requests

Use this directive to specify a template for requests that the server is not to process. After a request matches a template on a Fail directive, the request is not compared to request templates on any subsequent directives.

Format

```
Fail request_template [Server_IP_address | host_name]
```

request_template
Specifies a template for requests that the server is to reject. If a request matches the template, the server sends the requester an error message.

You can use an asterisk as a wildcard in the template. The tilde character (~) just after a slash (/) must be explicitly matched; a wildcard cannot be used to match it.

[*Server_IP_address* | *host_name*]
If you are using multiple IP addresses or virtual hosts, use this parameter to specify an IP address or a host name. The server uses the directive only for requests that come to the server on this IP address or for this host. For an IP address, this is the address of the server's network connection, not the address of the requesting client.

You can specify an IP address (for example, `240.146.167.72`) or you can specify a host name (for example, `hostA.bcd.com`).

This parameter is optional. Without this parameter, the server uses the directive for all requests regardless of the IP address the requests come in on or the host name in the URL.

A wildcard character cannot be specified for a server's IP address.

Examples

In the following example, the server rejects any requests beginning with `/usr/local/private/`.

```
Fail /usr/local/private/*
```

The following examples use the optional IP address parameter. The server rejects any requests beginning with `/customerB/` if the request comes in on a network connection with the IP address 240.146.167.72. The server rejects any requests beginning with `/customerA/` if the request comes in on a network connection with the IP address 0.83.100.45.

```
Fail    /customerB/*    240.146.167.72
Fail    /customerA/*    0.83.100.45
```

The following examples use the optional host name parameter. The server rejects any requests beginning with `/customerB/` if the request comes in for hostA.bcd.com. The server rejects any requests beginning with `/customerA/` if the request comes in for hostB.bcd.com.

```
Fail    /customerB/*    hostA.bcd.com
Fail    /customerA/*    hostB.bcd.com
```

Default

None

FIPSEnable — Enable Federal Information Processing Standard (FIPS) approved ciphers for SSLV3 and TLS

Use this directive to enable FIPS approved ciphers for SSLV3 and TLS protocol in SSL connections. When this directive is enabled, the list of supported cipher specifications for SSLV3 (V3CipherSpecs directive) is ignored. Also, the allowed TLS cipher specifications will be set to 352F0AFF09FE, and the SSLV3 cipher specifications will be set to FFFE.

Format

```
FIPSEnable {on | off}
```

Default

```
FIPSEnable  off
```

flexibleSocks — Enable flexible SOCKS implementation

Use this directive to instruct the proxy to use the SOCKS configuration file to determine the type of connection to make.

Format

```
flexibleSocks {on | off}
```

Default

```
flexibleSocks on
```

FTPDirInfo — Generate a welcome or description message for a directory

Use this directive to enable FTP servers to generate a welcome or descriptive message for a directory. This message can optionally be displayed as part of the FTP listings. The FTPDirInfo directive allows you to control where the message is displayed.

Format

```
FTPDirInfo  {top | bottom | off}
```

top
Display the welcome message at the top of the page, before the listing of files in the directory.

bottom
Display the welcome message at the bottom of the page, after the listing of files in the directory.

off
Do not display the welcome page.

Default

```
FTPDirInfo top
```

ftp_proxy — Specify another proxy server for FTP requests

If your proxy server is part of a chain of proxies, use this directive to specify the name of another proxy that this server contacts for FTP requests. You must specify a full URL, including the trailing slash character (/). For information on using an optional domain name or template, refer to“no_proxy — Specify templates for connecting directly to domains” on page 224.

Format

```
ftp_proxy full_URL [domain_name_or_template]
```

Example

```
ftp_proxy http:// outer.proxy.server/
```

Default

None

FTPUrlPath — Specify how FTP URLs are interpreted

Use this directive to specify whether the path information in FTP URLs is interpreted as relative to the logged-in user's working directory or to the root directory.

Format

```
FTPUrlPath  {relative | absolute}
```

If the FTPUrlPath directive is set to `absolute`, then the logged-in user's FTP working directory must be included in the FTP URL path. If `FTPUrlPath Relative` is specified, then the logged-in user's FTP working directory must be omitted from the FTP URL path. For example, to access the file test1.html, contained in the working directory /export/home/user1 for a logged-in user, the following URL paths are required, depending on the setting of the FTPUrlPath directive:

- If the setting is `FTPUrlPath absolute`, the required URL path is `ftp://ftphost/export/home/user1/test1.html`.
- If the setting is `FTPUrlPath relative`, the required URL path is `ftp://ftphost/test1.html`.

Default

None

Gc — Specify garbage collection

Use this directive to specify whether garbage collection is used. If caching is enabled, the server uses the garbage collection process to delete files that must no longer be cached. Files are deleted based on their expiration date and other proxy directive values. Generally, if caching is enabled, garbage collecting is used. If garbage collection is not used, the proxy cache is used inefficiently.

Format

```
Gc {on | off}
```

Default

```
Gc On
```

GCAdvisor — Customize the garbage collection process

Use this directive to specify a customized application you want the server to use for garbage collection.

Format

`GCAdvisor` */path/file*`:`*function_name*

/path/file
: Specifies the fully qualified file name of the compiled program, including the extension.

function_name
: Specifies the name of the application function within your program.

Example

```
GCAdvisor /api/bin/customadvise.so:gcadv
```

GcHighWater — Specify when garbage collection begins

Use this directive to specify the percentage of the total cache capacity that must be filled to trigger garbage collection. This percentage is called the *high-water mark*.

The high-water mark is specified as a percentage of the total cache capacity. Garbage collection continues until the low-water mark has been reached—see "GcLowWater — Specify when garbage collection ends" for information about setting this. The percentage for the high-water mark can be set between 50 and 95.

Format

`GcHighWater percentage`

Default

`GcHighWater 90`

GcLowWater — Specify when garbage collection ends

Use this directive to specify the percentage of the total cache capacity that triggers the end of garbage collection. This percentage is known as the *low-water mark*. The low-water mark is specified as a percentage of the total cache capacity. It must be set to a value lower than the value set for the high-water mark; see "GcHighWater — Specify when garbage collection begins" on page 203 for information about setting the high-water mark.

Format

`GcLowWater percentage`

Default

`GcLowWater 60`

gopher_proxy — Specify another proxy server for Gopher requests

If the proxy server is part of a chain of proxies, use this directive to specify the name of another proxy that this server contacts for Gopher requests. You must specify a full URL including the trailing slash (/). For information on using an optional domain name or template, refer to"no_proxy — Specify templates for connecting directly to domains" on page 224.

Format

`gopher_proxy full_URL[domain_name_or_template]`

Example

```
gopher_proxy http://outer.proxy.server/
```

Default

None

GroupId — Specify the group ID

Use this directive to specify the group name or number to which the server changes before accessing files.

If you change this directive, you must manually stop and then start your server again in order for the change to take effect. The change does not take effect if you only restart the server. (See Chapter 5, "Starting and stopping the Caching Proxy," on page 15.)

Note: If you change the server defaults for the user ID, group ID, or log directory paths, create the new directories and update the permissions and ownership of the directories. To enable the server to write information to a user-defined

log directory, set the permission for that directory as 755 and the set the user-defined server user ID as the owner. For example, if you change the user ID of the server from the default to jdoe, and the default logs directory to server_root/account, then the server_root/account directory must have the permission 755 and be owned by jdoe.

Format

```
GroupId { group_name | group_number}
```

Defaults

AIX: `GroupId nobody`

HP-UX: `GroupId other`

Linux:

Red Hat: `GroupId nobody`

SUSE: `GroupId nogroup`

Solaris: `GroupId nobody`

HeaderServerName — Specify the name of the proxy server returned in the HTTP header

Use this directive to specify the name of the proxy server returned in the HTTP header

Format

```
HeaderServerName name
```

Default

None

Hostname — Specify the fully qualified domain name or IP address for the server

Use this directive to specify the domain name or an IP address returned to clients from file requests. If you specify a domain name, a domain name server must be able to resolve the name into an IP address. If you specify an IP address, the domain name server is not needed or accessed.

Note: When an array is set up, the Hostname directive must be configured identically on all members of that array.

Format

```
Hostname {name | IP address}
```

Default

By default, this directive is not specified in the initial configuration file. If you do not specify this directive in the configuration file, the value defaults to the host name defined in your domain name server.

http_proxy — Specify another proxy server for HTTP requests

If the proxy server is part of a chain of proxies, use this directive to specify the name of another proxy that this server contacts for HTTP requests. You must specify a full URL, including the trailing slash (/). For information on using an

optional domain name or template, refer to“no_proxy — Specify templates for connecting directly to domains” on page 224.

Format

```
http_proxy full_URL[domain_name_or_template]
```

Example

```
http://outer.proxy.server/
```

Default

None

HTTPSCheckRoot — Filter HTTPS requests

Use this directive to specify whether the Caching Proxy retrieves the insecure home page for the URL and attempts to find labels in it. If labels are found, they are applied to the secure request. For example, if you request `https://www.ibm.com/`, the Caching Proxy retrieves http://www.ibm.com/, searches it for labels, and uses any labels it finds to filter https://www.ibm.com/.

If HTTPSCheckRoot is set to `off`, the Caching Proxy does not retrieve the insecure home page and search it for labels.

Format

```
HTTPSCheckRoot {on | off}
```

Default

```
HTTPSCheckRoot  on
```

ICP_Address — Specify IP address for ICP queries

Use this subdirective to specify an IP address that is used for sending and receiving ICP queries. It must be enclosed within the **<MODULEBEGIN> ICP** and **<MODULEEND>** directives.

Format

```
ICP_Address IP_address
```

Default

By default, this directive is not specified in the initial configuration file. If you do not specify this directive in the configuration file, the value defaults to accepting and sending ICP queries on any interface.

ICP_MaxThreads — Specify maximum threads for ICP queries

Use this subdirective to specify the number of threads spawned to listen for ICP queries. It must be enclosed within the **<MODULEBEGIN> ICP** and **<MODULEEND>** directives.

Note: On Redhat Linux 6.2 and lower, this number must be low because the maximum number of threads that can be created per process is small. Specifying a large number threads for ICP use might limit the number of threads available for use in servicing requests.

Format

```
ICP_MaxThreads number_of_threads
```

Default

```
ICP_MaxThreads   5
```

Occupier — Specify a member of an ICP cluster

If the proxy server is part of an ICP cluster, use this subdirective to specify the ICP peers. It must be enclosed within the **<MODULEBEGIN> ICP** and **<MODULEEND>** directives.

When a new peer is added to the ICP cluster, ICP peer information must be added to the configuration file of all existing peers. Use one line for each peer. Note that the current host can also be included in the peer list. When ICP initializes, it ignores the current host's entry. This makes it possible to have a single configuration file that can be copied to other peer machines without editing it to remove the current host.

Format

```
ICP_Peer hostname http_port icp_port
```

hostname
The name of the peer

http_port
The peer's proxy port

icp_port
The peer's ICP server port

Example

The following line adds the host abc.xcompany.com, whose proxy port is 80 and ICP port 3128, as a peer.

```
ICP_Peer  abc.xcompany.com  80  3128
```

Default

None

ICP_Port — Specify port number for ICP queries

Use this subdirective to specify the port number on which the ICP server listens for ICP queries. It must be enclosed within the **<MODULEBEGIN> ICP** and **<MODULEEND>** directives.

Format

```
ICP_Port port_number
```

Default

```
ICP_Port 3128
```

ICP_Timeout — Specify maximum wait time for ICP queries

Use this subdirective to specify the maximum time that the Caching Proxy waits for responses to ICP queries. The time is specified in milliseconds. It must be enclosed within the **<MODULEBEGIN> ICP** and **<MODULEEND>** directives.

Format

```
ICP_Timeout  timeout_in_milliseconds
```

Default

```
ICP_Timeout  2000
```

IgnoreURL — Specify URLs that are not refreshed

Use this directive to specify URLs that are not to be loaded by the cache agent. This directive is useful when the cache agent is loading pages linked from cached URLs. You can use multiple occurrences of the IgnoreURL directive to specify different URLs or URL masks. The value for this directive can contain asterisks (*) as wildcards, to apply a mask.

Format

```
IgnoreURL URL
```

Examples

```
IgnoreURL http://www.yahoo.com/
IgnoreURL http://*.ibm.com/*
```

Default

```
IgnoreURL */cgi-bin/*
```

imbeds — Specify whether server-side include processing is used

Use this directive to specify whether you want server-side include processing to be performed for files served from the file system, CGI programs, or both. Server-side include processing is done on files with a content type of `ext/x-ssi-html`. Optionally, you can specify that server-side include processing is also done for files with a content type of `text/html`. For more information about content types, see "AddType — Specify the data type of files with particular suffixes" on page 169.

You can use server-side include processing to dynamically insert information into the file being returned. Such information can include the date, the size of a file, the last change date of a file, CGI or server-side include environment variables, or text files. Server-side include processing is performed only on files originated locally. The Caching Proxy does not perform server-side include processing on proxied or cached objects.

Server-side include processing causes the server to search your files for special commands each time they are served. This can affect the server's performance and slow down response time to clients.

Format

```
imbeds {on | off | files | cgi | noexec}  {SSIOnly | html}
```

on Server-side include processing is done for files from the file system and from CGI programs.

off
Server-side include processing is not done for any files.

files
Server-side include processing is done only for files from the file system.

cgi
Server-side include processing is done only for files returned by CGI programs.

noexec

SSIOnly
Server-side include processing is done for files with a content type of `text/x-ssi-html`.

html
Server-side include processing is done for files with a content type of text/html and a content type of `text/x-ssi-html`.

The server checks the content type of each file it retrieves and the output of each CGI program it processes.

Server-side include processing is normally done only for files having a content type of `text/x-ssi/html`. However, you can specify that files with a content type of `text/html` are processed for server-side includes.

Note: The server treats html, .html, and .htm as `html`. Anything else is treated as `SSIOnly`.

Each suffix must have an AddType directive defined with the correct content type. If you use suffixes other than .htm or .html, make sure an AddType directive is defined with a content type of `text/x-ssi/html`.

Default
`imbeds on SSIOnly`

ImportCacheImageFrom — Import cache memory from a file
Use this directive to import the cache contents from a dump file. This is useful when memory cache gets lost during restart, or when deploying the same cache for multiple proxies.

Format
`ImportCacheImageFrom` *`import_file_name`*

Default
None

InheritEnv — Specify which environment variables are inherited by CGI programs
Use this directive to specify which environment variables you want your CGI programs to inherit (other than the CGI environment variables that are specific to CGI processing).

If you do not include an InheritEnv directive, all environment variables are inherited by CGI programs. If you include any InheritEnv directive, only those environment variables specified on InheritEnv directives are inherited along with the CGI-specific environment variables. The directive allows you to optionally initialize the value of the variables that are inherited.

Format
`InheritEnv` *`environment_variable`*

Examples
```
InheritEnv PATH
InheritEnv LANG=ENUS
```

In this example, only the PATH and LANG environment variables are inherited by CGI programs, and the LANG environment variable is initialized with the value of ENUS.

Default
None. By default, all environment variables are inherited by CGI programs.

InputTimeout — Specify the input timeout

Use this directive to set the time allowed for a client to send a request after making a connection to the server. A client first connects to the server and then sends a request. If the client does not send a request within the amount of time specified with this directive, the server closes the connection. Specify the time value in any combination of hours, minutes (or mins), and seconds (or secs).

Format
```
InputTimeout time
```

Example
```
InputTimeout 3 mins 30 secs
```

Default
```
InputTimeout 2 minutes
```

JunctionReplaceUrlPrefix — Replace URL instead of insert prefix when used with JunctionRewrite plugin

This directive will override the default action of the JunctionRewrite plugin, allowing the proxy to correct certain URL links in the html page. It is used in conjunction with the JunctionRewrite directive.

The JunctionReplaceUrlPrefix directive will direct the JunctionRewrite plugin to replace the URL from *url_pattern_1* to *url_pattern_2*, instead of inserting a prefix at the beginning of the URL.

Format
```
JunctionReplaceUrlPrefix url_pattern_1 url_pattern_2
```

Example
```
JunctionReplaceUrlPrefix /server1.internaldomain.com/*  /server1/*
```

In this example, assume the URL is /server1.internaldomain.com/notes.nsf and assume the prefix is /server1. Instead of inserting the prefix to rewrite the URL to /server1/server1.internaldomain.com/notes.nsf, the JunctionRewrite plugin will change the URL to /server1/notes.nsf.

Default
None

JunctionRewrite — Enables URL rewriting

This directive enables the junction rewriting routine within the Caching Proxy to rewrite responses from origin servers to ensure that server relative URLs get mapped to the appropriate origin server when junctions are used. The junction rewriting plugin must also be enabled if you set **JunctionRewrite on** without the UseCookie option. Junctions are defined by the proxy mapping rules.

See “UseCookie as an alternative to JunctionRewrite” on page 44 and “Sample transmogrifier plugin to extend JunctionRewrite functionality” on page 45 for additional information about JunctionRewrite.

Format

```
JunctionRewrite {on | on UseCookie | off}
```

Default

```
JunctionRewrite off
```

JunctionRewriteSetCookiePath — Rewrite the path option in the Set-Cookie header, when used with JunctionRewrite plugin

The directive will allow the proxy to rewrite the path option in the Set-Cookie header when the cookie name is matched. If the response needs junction, and a junction prefix is defined, the prefix will be inserted before each path. It can be used with the JunctionRewrite plugin, or it can be used with the RewriteSetCookieDomain directive.

Format

```
JunctionRewriteSetCookiePath cookie-name1 cookie-name2...
```

cookie-name
A cookie name in the Set-Cookie header.

Default

None

JunctionSkipUrlPrefix — Skip rewriting URLs that already contain the prefix, when used with JunctionRewrite plugin

This directive will override the default action of the JunctionRewrite plugin, skipping the URL rewrite if the URL-pattern already matches. It works with the JunctionRewrite plugin providing a way to correct some of the URL links in the html page. Normally, the directive is used to skip the URLs that already include a prefix.

Format

```
JunctionSkipUrlPrefix url_pattern
```

Example

```
JunctionSkipUrlPrefix  /server1/*
```

In this example, assume the URL is /server1/notes.nsf and assume the junction prefix is /server1/. Instead of rewriting the URL to /server1/server1/notes.nsf, the JunctionRewrite plugin will skip rewriting the URL, and the URL remains unchanged as /server1/notes.nsf.

Default

None

KeepExpired — Specify returning the expired copy of the resource if that resource is being updated on the proxy

Use this directive to help prevent flooding backend servers with requests while a cache object is being revalidated.

When a cache object is being revalidated with the content on the backend server, requests for the same resource will be proxied to the backend server. Sometimes the flood for the same requests will cause the backend server to go down. Enabling

this directive can help prevent this situation from occurring. When the directive is enabled, an expired or staled copy of the resource will be returned if that resource is being updated on the proxy.

Format

```
KeepExpired {on | off}
```

Default

```
KeepExpired off
```

KeyRing — Specify the file path to the key ring database

Use this directive to specify the file path to the key ring database that the server uses for SSL requests. Key ring files are generated via the iKeyman key manager utility.

Note: The SSL directives are not supported on SUSE Linux.

Format

```
KeyRing filename
```

Examples

Windows:
```
KeyRing c:\Program Files\IBM\edge\cp\\key.kdb
```

Linux and UNIX:
```
KeyRing /etc/key.kdb
```

Default

None

KeyRingStash — Specify the file path to the key ring database's password file

Use this directive to to specify the file path to the key ring database's password file. The password file is generated via the iKeyman key manager utility when a key ring database file is built.

Note: The SSL directives are not supported on SUSE Linux.

Format

```
KeyRingStash file_path
```

Examples

Windows:
```
KeyRingStash c:\Program Files\IBM\edge\cp\key.sth
```

Linux and UNIX:
```
KeyRingStash /etc/key.sth
```

Default

None

LimitRequestBody — Specify the maximum body size in PUT or POST requests

Use this directive to control the maximum body size in PUT or POST requests. The LimitRequest directives are used to protect the proxy from attack.

The value can be specified in kilobytes (K), megabytes (M), or gigabytes (G).

Format

```
LimitRequestBody max_body_size {K | M | G}
```

Default

```
LimitRequestBody 10 M
```

LimitRequestFields — Specify the maximum number of headers in client requests

Use this directive to specify the maximum number of headers that can be sent in client requests. The LimitRequest directives are used to protect the proxy from attack.

Format

```
LimitRequestFields number_headers
```

Default

```
LimitRequestFields 32
```

LimitRequestFieldSize — Specify the maximum header length and request line

Use this directive to specify the maximum length of the request line and the maximum length of each header in a request. The LimitRequest directives are used to protect the proxy from attack.

The value can be specified in bytes (B) or kilobytes (K).

Format

```
LimitRequestFieldSize max_hdr_length {B | K}
```

Default

```
LimitRequestFieldSize 4096 B
```

ListenBacklog — Specify the number of listen backlog client connections that the server can carry

Use this directive to specify the number of listen backlog client connections that the server carries before it sends connection refused messages to clients. This number depends on the number of requests that your server can process in a few seconds. Do not set it to a value higher than the number the server can process before the clients time out and abort the connection.

Note: If the ListenBacklog value is greater than the SOMAXCONN value supported by TCP/IP, the SOMAXCONN value is used instead.

Format

```
ListenBacklog number_of_requests
```

Default

```
ListenBacklog 128
```

LoadInlineImages — Control the refreshing of imbedded images

Use this directive to specify whether inline images are retrieved by the cache agent. If LoadInlineImages is set to `on`, images that are imbedded in a page that is being cached are also cached. If it is set to `off`, imbedded images are not cached.

Format

```
LoadInlineImages {on | off}
```

Default

```
LoadInlineImages on
```

LoadTopCached — Specify the number of popular pages to refresh

Use this directive to instruct the cache agent to access the previous night's cache access log and load the most-requested URLs.

The Caching directive must be set to `On`, and a value must be set for the CacheAccessLog directive when a value is set for the LoadTopCached directive.

Format

```
LoadTopCached number_of_pages
```

Default

```
LoadTopCached 100
```

LoadURL — Specify the URLs to refresh

Use this directive to specify URLs to be loaded into the cache by the cache agent. Multiple LoadURL directives can be included in the configuration file, but wildcards cannot be used.

Format

```
LoadURL url
```

Example

```
LoadURL http://www.ibm.com/
```

Default

None

Log — Customize the Log step

Use this directive to specify a customized application function that the server calls during the Log step. This code provides logging and other processing that is performed after the connection is closed.

Format

```
Log request_template /path/file:function_name
```

request_template
: Specifies a template for requests that further determine whether your application function is called. The specification can include the protocol, domain and host; it can be preceded by a slash (/) and can use an asterisk (*) as a wildcard. For example, `/front_page.html`, `http://www.ics.raleigh.ibm.com`, `/pub*`, `/*`, and `*` are all valid.

/path/file
Specifies the fully qualified file name of the compiled program, including the extension.

function_name
Specifies the name of the application function within your program. You must supply the names of the open, write, and close functions.

Example

```
Log /index.html /api/bin/icsextpgm.so:log_url
```

Default

None

LogArchive — Specify the behavior of log archiving

Use this directive to specify the behavior of the archiving routine. This directive affects all logs with global settings. It specifies that logs are compressed or purged, or that nothing is done with them.

If you specify `Compress`, use the CompressAge and CompressDeleteAge directives to specify when the logs are compressed or deleted. Use the CompressCommand directive to specify which command and its parameters to use.

If you specify `Purge`, use the PurgeAge and PurgeSize directives to specify when the logs are purged.

Format

```
LogArchive {Compress | Purge | none}
```

Compress
Specifies that the archiving routine compresses the logs.

Purge
Specifies that the archiving routine erases the logs.

none
Specifies that the archiving routine does nothing.

Default

```
LogArchive Purge
```

Related directives

- "CompressAge — Specify when to compress logs" on page 185
- "CompressDeleteAge — Specify when to delete logs" on page 186
- "CompressCommand — Specify the compression command and parameters" on page 185
- "Midnight — Specify the API plugin used to archive logs" on page 221
- "PurgeAge — Specify the age limit for a log" on page 243
- "PurgeSize — Specify the limit for the size of the log archive" on page 243

LogFileFormat — Specify the access log format

Use this directive to specify the file format of the access log files.

Format

```
LogFileFormat  {common | combined}
```

By default, logs are displayed in the NCSA Common Log Format. Specify `combined` to display logs in NCSA Combined Log Format instead. The combined format adds fields for Referring URL, User Agent, and Cookie (if present in the request).

Default

```
LogFileFormat  common
```

LogToGUI (Windows only) — Display log entries in the server window

Windows systems only. When running the proxy via the command line, use this directive to output to the access log. To optimize server performance, by default this directive is set to `off` (disabled).

Note: This directive has no effect when running the proxy as a service.

Format

```
LogToGUI  {on | off}
```

Default

```
LogToGUI off
```

LogToSyslog — Specify whether to send access information to the system log (Linux and UNIX only)

Linux and UNIX systems only. Use this directive to specify whether the server logs access requests and errors to the system log in addition to the access and error log files.

Format

```
LogToSyslog {on | off}
```

The system log file must be present on your server before you specify that error log information is written to it. You can choose whether to log access information, error information, or both.

To send only error information to the system log, add the following line to your /etc/syslog.conf file:

`user.err` *`syslog_output_file_for_error_information`*

To send only access information to the system log, add the following line to your /etc/syslog.conf file:

`user.info` *`syslog_info_file_for_access_information`*

To send both error and access information to the system log, add both of the lines to your /etc/syslog.conf file:

Specify *syslog_output_file* and *syslog_info_file* in the following formats:

- **AIX:** `/var/adm/`*`name_of_syslog_file`*
- **HP-UX:** `/var/adm/syslog/syslog.log`
- **Linux:** `/var/adm/messages`
- **Solaris:** `/var/adm/messages`

After you create the system log file, you can restart it with the following command:

```
kill -HUP 'cat /etc/syslog.pid'
```

Default

```
LogToSyslog Off
```

Map — Change matching requests to a new request string

Use this directive to specify a template for requests that you want to change to a new request string. After the server changes the request, it takes the new request string and compares it to the request templates on subsequent directives.

Format

`Map` *request_template new_request* [*server_IP_address* | *host_name*]

request_template
: Specifies a template for requests that the server changes and then continues comparing the new request string to other templates.

  You can use an asterisk (*)as a wildcard in the template. The tilde character (~) just after a slash (/) must be explicitly matched; a wildcard cannot be used to match it.

new_request
: Specifies the new request string with which the server continues to compare the request templates on subsequent directives. The string specified with *new_request* can contain a wildcard if the *request_template* has one. The part of the request that matches the *request_template* wildcard is inserted in place of the wildcard in *new_request*.

[*server_IP_address* | *host_name*]
: If you are using multiple IP addresses or virtual hosts, use this parameter to specify an IP address or a host name. The server uses the directive only for requests that come to the server on this IP address or for this host. For an IP address, this is the address of the server's network connection, not the address of the requesting client.

  You can specify an IP address (for example, `240.146.167.72`) or you can specify a host name (for example, `hostA.raleigh.ibm.com`).

  This parameter is optional. Without this parameter, the server uses the directive for all requests regardless of the IP address the requests come in on or the host name in the URL.

  A wildcard character cannot be specified for a server's IP address.

Examples

- In the following example, the server takes any requests starting with `/stuff/` and changes the `/stuff/` portion of the request to `/good/stuff/`. Anything that follows `/stuff/` on the original request is also included in the new request string. Therefore, `/stuff/whatsup/` changes to `/good/stuff/whatsup/`. The server takes the new request string and continues to compare it to request templates on subsequent directives.

  ```
  Map  /stuff/*   /good/stuff/*
  ```

- The following examples use the optional IP address parameter. If your server receives requests that begin with `/stuff/`, it changes the request to a different request string based on the IP address of the network connection on which the request comes in. For requests coming in on 240.146.167.72, the server changes the `/stuff/` portion of the request to `/customerA/good/stuff/`. For requests coming in on any connection with an address of 0.83.100.45, the server changes the `/stuff/` portion of the request to `/customerB/good/stuff/`.

```
Map   /stuff/*   /customerA/good/stuff/*   240.146.167.72
Map   /stuff/*   /customerB/good/stuff/*   0.83.104.45
```

- The following examples use the optional host name parameter. If the server receives requests that begin with `/stuff/`, it changes the request to a different request string based on the host name in the URL. For requests coming in for hostA, the server changes the `/stuff/` portion of the request to `/customerA/good/stuff//`. For requests coming in for hostB, the server changes the `/stuff/` portion of the request to `/customerB/good/stuff/`.

```
Map   /stuff/*   /customerA/good/stuff/*   hostA.bcd.com
Map   /stuff/*   /customerB/good/stuff/*   hostB.bcd.com
```

Default

None

MaxActiveThreads — Specify the maximum number of active threads

Use this directive to set the maximum number of threads that are active at one time. If the maximum is reached, the server holds new requests until another request finishes and threads become available. Generally, the more power a machine has, the higher the value that is set for this directive. If a machine starts to spend too much time on overhead tasks, such as swapping memory, try reducing this value.

Format

`MaxActiveThreads` *`number_of_threads`*

Default

```
MaxActiveThreads 100
```

MaxContentLengthBuffer — Specify the size of the buffer for dynamic data

Use this directive to set the size of the buffer for dynamic data generated by the server. Dynamic data is output from CGI programs, server-side includes, and API programs.

The value can be specified in bytes (B), kilobytes (K), megabytes (M), or gigabytes (G). It does not matter whether there is a space between the number and the value (B, K, M, G).

Format

`MaxContentLengthBuffer` *`size`*

Default

```
MaxContentLengthBuffer 100 K
```

MaxLogFileSize — Specify the maximum size for each log file

Use this directive to specify the maximum size for each log file. Each log file cannot exceed the size defined by this directive. Once a log file reaches the maximum defined size, the current log file will be closed, and a new log file will be created with the same name, appended by the next incremental integer value.

Notes:

1. Caching Proxy is a 32-bit application and opens its log files with a 32-bit function. Because of this constraint, do *not* specify a MaxLogFileSize greater

than 2 GB. Caching Proxy may hang if the log file exceeds 2 GB in size when Caching Proxy attempts to write to the log file while still actively processing requests.

2. On Linux and UNIX platforms, the log files will not be created if the permissions of the directory that the log files reside do not have write permissions for at least the group that the ibmproxy daemon runs under. In other words, the log file locations for the logging directives in the ibmproxy.conf file must have write permissions for at least the group defined by the GroupId directive in the ibmproxy.conf file. This is only a problem when the default location of the log files has been changed or when the default UserId or GroupId directive has been changed in the ibmproxy.conf file.

The maximum size can be specified in one of the following units: bytes (B), kilobytes (K), megabytes (M), and gigabytes (G).

Format

`MaxLogFileSize` *`maximum`* `{B | K | M | G}`

Default

`MaxLogfileSize 128 M`

MaxPersistRequest — Specify the maximum number of requests to receive on a persistent connection

Use this directive to specify the maximum number of requests the server receives on a persistent connection. When determining this number, consider the number of images used in your pages. Each image requires a separate request.

Format

`MaxPersistRequest` *`number`*

Default

`MaxPersistRequest 5`

MaxQueueDepth — Specify the maximum number of URLs to queue

Use this directive to specify the maximum depth for the cache agent's queue of outstanding page retrieval requests. If you have a large system with a large amount of memory, you can define a larger queue of page retrieval requests without consuming all the available memory.

The queue of URLs to cache is determined at the beginning of each run of the cache agent. If you instruct the cache agent to follow the hypertext links to other URLs, these other URLs are not counted in the cache queue depth. After the value specified in the MaxURLs directive is reached, the cache agent stops, even if there are more URLs in the queue.

Format

`MaxQueueDepth` *`maximum_depth`*

Default

`MaxQueueDepth 250`

MaxRuntime — Specify the maximum time for a cache agent run

Use this directive to specify the maximum amount of time for the the cache agent to retrieve URLs during a particular run. A value of `0` means the cache agent runs until completion.

Format

```
MaxRuntime {0 | maximum_time}
```

Example

```
MaxRuntime 2 hours 10 minutes
```

Default

```
MaxRuntime 2 hours
```

MaxSocketPerServer — Specify the maximum open sockets for server

Use this directive to set the maximum number of open sockets to maintain to any one origin server. Use this directive only if the ServerConnPool directive is set to `on`.

Format

```
MaxSocketPerServer num
```

Example

```
MaxSocketPerServer 10
```

Default

```
MaxSocketPerServer 5
```

MaxUrls — Specify the maximum number of URLs to refresh

Use this directive to specify the maximum number of URLs the cache agent retrieves during a particular run. A value of `0` means there is no limit. When the automatic mode of the cache agent is used, the LoadURL and LoadTopCached directives take precedence over MaxURLs.

Format

```
MaxURLs maximum_number
```

Default

```
MaxURLs 2000
```

Member — Specify a member of an array

Use this directive to specify members of the arrays that are shared by the servers using remote cache access.

Note: When setting up an array, configure the Hostname directive identically on all members of that array.

Format

```
Member name {
subdirective
subdirective
.
.
}
```

The following subdirectives are included:

RCAAddr
This required subdirective identifies the IP address or host name for RCA communication.

RCAPort
This required subdirective identifies the port for RCA communication. The port number must be greater than 1024 and less than 65535.

CacheSize {*n* bytes | *n* Kbytes | *n* Mbytes | *n* Gbytes}
This required subdirective identifies the size of this member's cache, which must be a positive value.

[Timeout *n* milliseconds | *n* seconds | *n* hours | *n* days | *n* months | *n* years | forever]
Identifies how long to wait for this member. *n* must be a positive integer. `Timeout` is optional; the default is `1000 milliseconds`. Timeout values typically are set in seconds or milliseconds.

[BindSpecific {on | off}]
Allows communications to occur on a private subnet, providing a measure of security. `BindSpecific` is optional; the default is `On`.

[ReuseAddr {on | off}]
Allows faster rejoining of the array; Setting this to `On` allows other processes to steal the port, which can cause undefined behavior. `ReuseAddr` is optional; the default is `Off`.

Example

```
Member bittersweet.chocolate.ibm.com {
  RCAAddr      127.0.0.1
  RCAPort      6294
  CacheSize    25G
  Timeout      500 milliseconds
  BindSpecific On
  ReuseAddr    Off
  }
```

Default

None

Midnight — Specify the API plugin used to archive logs

Use this directive to specify the application plugin that runs at midnight to archive the logs. This directive is initialized during installation. If you do not include this directive in the configuration file, archiving is not performed.

Format

```
Midnight /path/file:function_name
```

/path/file
Specifies the fully qualified file name of the compiled program, including the extension.

function_name
Specifies the name of the application function within your program.

Defaults

- **Linux and UNIX:** `Midnight /usr/lib/archive.so:begin`
- **Windows:** `Midnight C:\Program Files\IBM\edge\cp\\bin\archive.dll:begin`

NameTrans — Customize the Name Translation step

Use this directive to specify a customized application function the server calls during the Name Translation step. This code supplies the mechanism for translating the virtual path in the request to the physical path on the server, mapping URLs to specific objects.

Note: This is not a terminal-mapping rule. The transformed URL still must match one of the terminal-mapping rule directives, such as Exec, Fail, Map, Pass, Redirect, and Service.

Format

```
NameTrans request_template /path/file:function_name
  [Server_IP_address | host_name]
```

request_template
Specifies a template for requests that further determine whether the application function is called. The specification can include the protocol, domain and host; it can be preceded by a slash (/) and can use an asterisk (*) as a wildcard. For example, `/front_page.html`, `http://www.ics.raleigh.ibm.com`, `/pub*`, `/*`, and `*` are all valid.

/path/file
Specifies the fully qualified file name of the compiled program, including the extension.

function_name
Specifies the name of the application function within your program.

[*Server_IP_address* **|** *host_name***]**
If you are using multiple IP addresses or virtual hosts, determines whether your application function is called only for requests coming in on a specific IP address or for a specific host.

A wildcard character cannot be specified for a server's IP address.

Note: The directive must be typed on one line, even though it is shown here on two lines for readability.

Example

```
NameTrans /index.html /api/bin/icsextpgm.so:trans_url
```

Default

None

NoBG — Run the Caching Proxy process in foreground

On Linux and UNIX platforms, use this directive to prevent the Caching Proxy server process from automatically running in the background. The directive, which is set to off by default, has the format:

```
NoBG [on | off]
```

Note: The -**nobg** option for the **ibmproxy** command is not valid for Windows systems.

Example

```
NoBG on
```

Default

```
NoBG off
```

NoCaching — Specify that files with URLs that match a template are not cached

Use this directive to specify that the server does not cache files with URLs matching the specified template. You can include multiple occurrences of this directive in the configuration file. Include a separate directive for each template. The URL template must include the protocol.

If neither the CacheOnly or the NoCaching directive is set, any URL is a candidate for caching.

Format

```
NoCaching URL_pattern
```

Example

```
NoCaching http://joke/*
```

Default

None

NoLog — Suppress log entries for specific hosts or domains that match a template

Use this directive to specify that access requests made from specific hosts or domains that match a specified template are not logged. For example, you might not want to log access requests from local hosts.

You can include multiple occurrences of this directive in your configuration file. You can also put multiple templates on the same directive if you separate them by one or more spaces. You can use host names or IP number addresses on the templates.

Note: To use host name templates, you must set the DNS-Lookup directive to `On`. If the DNS-Lookup directive is set to `Off` (the default), you can use IP address templates only.

Format

```
NoLog  {host_name | IP_address}  [...]
```

Example

```
NoLog 128.0.*  *.edu  localhost.*
```

Default

None

no_proxy — Specify templates for connecting directly to domains

If you are using the directive http_proxy, ftp_proxy, or gopher_proxy for proxy chaining, you can use this directive to specify the domains that the server connects to directly instead of going through a proxy.

Specify the value as a string of domain names or domain name templates. Separate each entry in the string with a comma (,). Do *not* use any spaces in the string.

Templates on this directive are entered differently than on other directives. Most importantly, you *cannot* use the wildcard character (*). You *can* specify a template by including only the last part of a domain name. The server connects directly to any domains that end with a string matching the templates you specify. This directive applies only to proxy chaining and is equivalent to a direct `@/=` line in the SOCKS configuration file.

Format

```
no_proxy  domain_name_or_template[,...]
```

Example

```
no_proxy   www.someco.com,.raleigh.ibm.com,.some.host.org:8080
```

In this example, the server does not go through a proxy for the following requests:

- Any requests to domains ending with www.someco.com
- Any requests to domains ending with .raleigh.ibm.com, such as blugrass.raleigh.ibm.com or keystone.raleigh.ibm.com
- Any requests to port 8080 of domains ending with .some.host.org, such as myname.some.host.org:8080. (This does not include requests to any other ports of the same domain, such as myname.some.host.org, which assumes the default port 80.)

Default

None

NoProxyHeader — Specify the client headers to block

Use this directive to specify client URL headers to block. Any HTTP header sent by a client can be blocked, including required headers. Use extreme care when blocking headers. Common headers include:

- `Pragma:`—Usually used to instruct browsers and servers with caches to fetch the file from the original server every time the file is requested.
- `Referer:`—URL of the file from which the Request-URI was obtained.

See the HTTP protocol specification for details of these and other headers. You can specify this directive multiple times.

Format

```
NoProxyHeader header
```

Example

```
NoProxyHeader Referer:
```

Default

None

NumClients — Specify the number of cache agent threads to use

Use this directive to specify the number of threads the cache agent uses to retrieve pages in the queue. Base the number of threads on the speed of your internal network and your connection to the Internet. The allowable range is 1 through 100.

Note: Using more than six threads can possibly lead to excessively rapid requests on content servers.

Format

```
NumClients number
```

Default

```
NumClients 4
```

ObjectType — Customize the Object Type step

Use this directive to specify a customized application function that the server calls during the Object Type step. This code locates the requested object in the file system and specifies its MIME type.

Format

```
ObjectType request_template /path/file:function_name
```

request_template
Specifies a template for requests that further determine whether your application function is called. The specification can include the protocol, domain, and host; it can be preceded by a slash (/) and can use an asterisk (*) as a wildcard. For example, `/front_page.html`, `http://www.ics.raleigh.ibm.com`, `/pub*`, `/*`, and `*` are all valid.

/path/file
Specifies the fully qualified file name of the compiled program, including the extension.

function_name
Specifies the name of the application function within your program.

Example

```
ObjectType /index.html /api/bin/icsextpgm.so:obj_type
```

Default

None

OutputTimeout — Specify the output timeout

Use this directive to set the maximum time allowed for the server to send output to a client. The time limit applies to requests for local files and requests for which the server is acting as a proxy. The time limit does not apply for requests that start a local CGI program.

If the server does not send the complete response within the time limit specified on this directive, the server drops the connection. Specify the time value in any combination of hours, minutes (or mins), and seconds (or secs).

Format

```
OutputTimeout time
```

Default

`OutputTimeout 30 minutes`

PacFilePath — Specify the directory containing the PAC files

Use this directive to specify the directory containing the proxy autoconfiguration files generated by using the remote config PAC file form.

Format

`PacFilePath` *`directory_path`*

Defaults

- **Windows:** `PacFilePath c:\Program Files\IBM\edge\cp\\HTML\pacfiles`
- **Linux and UNIX:** `PacFilePath /opt/ibm/edge/cp/server_root/pub/pacfiles`

Pass — Specify the template for accepting requests

Use this directive to specify a template for requests that you want to accept and respond to with a file from your server. After a request matches a template on a Pass directive, the request is not compared to request templates on any subsequent directives.

Format

`Pass` *`request_template`* [*`file_path`* [*`server_IP_address`* | *`host_name`*]]

`request_template`
Specifies a template for requests that you want the server to accept and respond to with a file.

You can use an asterisk (*) as a wildcard in the template. The tilde character (~) just after a slash (/) must be explicitly matched; a wildcard cannot be used to match it.

[*`file_path`*]
Specifies the path to the file that the server is to return. The *`file_path`* can contain a wildcard if the *`request_template`* has one. The part of the request that matches the *`request_template`* wildcard is inserted in place of the wildcard in *`file_path`*.

This parameter is optional. If you do not specify a path, the request itself is used as the path.

[*`server_IP_address`* | *`host_name`*]
If you are using multiple IP addresses or virtual hosts, use this parameter to specify an IP address or a host name. The server uses the directive only for requests that come to the server on this IP address or for this host. For an IP address, this is the address of the server's network connection, not the address of the requesting client.

You can specify an IP address (for example, `240.146.167.72`) or you can specify a host name (for example, `hostA.raleigh.ibm.com`).

This parameter is optional. Without this parameter, the server uses the directive for all requests regardless of the IP address the requests come in on or the host name in the URL.

A wildcard character cannot be specified for a server's IP address.

Examples

- In the following examples, the server responds to a request starting `/updates/parts/` with a file from the listed path, depending on the operating system. Anything that follows `/updates/parts/` is also used to specify the file.

  Linux and UNIX systems:
  ```
  Pass /updates/parts/*
  /opt/ibm/edge/cp/server_root/pub/*
  ```

  Windows systems:
  ```
  Pass /updates/parts/*  c:\Program
  Files\IBM\edge\cp\pub\*
  ```

- In the following example, the server responds to a request starting with `/gooddoc/` with a file from the directory /gooddoc. So the server responds to the request `/gooddoc/volume1/issue2/newsletter4.html` with the document in file /gooddoc/volume1/issue2/newsletter4.html.

  ```
  Pass /gooddoc/*
  ```

- The following examples use the optional IP address parameter. If the server receives requests that begin with `/parts/`, it returns a file from a different directory based on the IP address of the network connection on which the request comes in. For requests coming in on 240.146.167.72, the server returns a file from /customerA/catalog/. For requests coming in on any connection with an address 0.83.100.45, the server returns a file from /customerB/catalog/.

  ```
  Pass   /parts/*   /customerA/catalog/*   240.146.167.72
  Pass   /parts/*   /customerB/catalog/*   0.83.100.45
  ```

- The following examples use the optional host name parameter. If the server receives requests that begin with `/parts/`, it returns a file from a different directory based on the host name in the URL. For requests coming in for hostA, the server returns a file from /customerA/catalog/. For requests coming in for hostB, the server returns a file from /customerB/catalog/.

  AIX systems

  ```
  Pass /Admin/*    /usr/lpp/internet/server_root/Admin/*
  Pass /Docs/*     /usr/lpp/internet/server_root/Docs/*
  Pass /errorpages/*  /usr/lpp/internet/server_root/pub/errorpages/*
  Pass /*          /usr/lpp/internet/server_root/pub/*
  ```

  Solaris, HP-UX, and Linux systems

  ```
  Pass /Admin/*    /opt/ibm/edge/cp/server_root/Admin/*
  Pass /Docs/*     /opt/ibm/edge/cp/server_root/Docs/*
  Pass /errorpages/*  /opt/ibm/edge/cp/server_root/pub/errorpages/*
  Pass /*          /opt/ibm/edge/cp/server_root/pub/*
  ```

Defaults

AIX systems

```
Pass    /Admin/*    /usr/lpp/internet/server_root/Admin/*
Pass    /Docs/*     /usr/lpp/internet/server_root/Docs/*
Pass    /errorpages/*  /usr/lpp/internet/server_root/pub/errporpages/*
Pass    /*          /usr/lpp/internet/server_root/pub/*
```

HP-UX, Linux, and Solaris systems

```
Pass    /Admin/*    /opt/ibm/edge/cp/server_root/Admin/*
Pass    /Docs/*     /opt/ibm/edge/cp/server_root/Docs/*
Pass    /errorpages/*  /opt/ibm/edge/cp/server_root/pub/errorpages/*
Pass    /*          /opt/ibm/edge/cp/server_root/pub/*
```

Windows systems

```
Pass    /icons/*    C:\Program Files\IBM\edge\cp\icons\*
Pass    /Admin/*    C:\Program Files\IBM\edge\cp\Admin\*
Pass    /Docs/*     C:\Program Files\IBM\edge\cp\Docs\*
Pass    /erropages/*  C:\Program Files\IBM\edge\cp\pub\errorpages\*
Pass    /*          C:\Program Files\IBM\edge\cp\pub\*
```

PersistTimeout — Specify the time to wait for the client to send another request

Use this directive to specify the length of time that the server waits between client requests before canceling a persistent connection. The time can be specified in any valid time increment, but it is usually specified in seconds or minutes.

The server uses a different timeout directive, the InputTimeout, to determine how long to wait for the client to send the first request after the connection is established. For more information about the input timeout, see "InputTimeout — Specify the input timeout" on page 210.

After the server sends its first response, it uses the value set for the PersistTimeout directive to determine how long to wait for each subsequent request before canceling the persistent connection.

Format

```
PersistTimeout  time
```

Default

```
PersistTimeout 4 seconds
```

PICSDBLookup — Customize the PICS label retrieval step

Use this directive to specify a customized application function that the server calls to retrieve PICS labels for a specified URL. The function can either dynamically create a PICS label for the requested file or search for a PICS label in an alternative file or database.

Format

```
PICSDBLookup  /path/file:function_name
```

/path/file
: Specifies the fully qualified file name of the compiled program, including the extension.

function_name
: Specifies the name of application function within your program.

Example

```
PICSDBLookup  /api/bin/icsext05.so:get_pics
```

Default

None

PidFile (Linux and UNIX only) — Specify the file in which to store the process ID of Caching Proxy

Linux and UNIX only. Use this directive to specify the location of the file that contains the process ID of Caching Proxy. When the server process starts, it records its process ID (PID) in a file. If multiple instances of the server are running on a single system, each instance must have its own PidFile directive.

Format

```
PidFile path_to_pid_file_info
```

Example

```
PidFile /usr/pidinfo
```

Defaults

- If a ServerRoot directive is specified: `PidFile` *`server_root`* `/ibmproxy-pid`
- If no ServerRoot directive is specified: `PidFile /tmp/ibmproxy-pid`

Plugin module directives

The directives listed below have been added to the Caching Proxy ibmproxy.conf file to enable new features and plugins. Configuration and Administration forms are not available for editing most of these directives. A standard text editor, such as vi or emacs, must be used to manually edit them. Further information on each of these new directives appears in this chapter, in alphabetical order.

- "ExternalCacheManager — Configure the Caching Proxy for dynamic caching from IBM WebSphere Application Server" on page 200
- "ICP_Address — Specify IP address for ICP queries" on page 206
- "ICP_Port — Specify port number for ICP queries" on page 207
- "ICP_Timeout — Specify maximum wait time for ICP queries" on page 207
- "Occupier — Specify a member of an ICP cluster" on page 207
- "ICP_MaxThreads — Specify maximum threads for ICP queries" on page 206
- "SignificantURLTerminator — Specify a terminating code for URL requests" on page 252
- "SSLCertificate — Specify key labels for certificates" on page 254
- "SSLOnly — Disable listener threads for HTTP requests" on page 255

In the ibmproxy.conf file, directives used to configure Caching Proxy plugin modules must be entered in the following format:

```
<MODULEBEGIN> plugin name
subdirective1
subdirective2

<MODULEEND>
```

Each plugin program parses the ibmproxy.conf file and reads only its own block of subdirectives. The Caching Proxy parser disregards everything between `<MODULEBEGIN>` and `<MODULEEND>`.

Caching Proxy plugin modules and some new features require that API directives be added to the ibmproxy.conf file. Because the proxy server interacts with the plugin modules in the order in which they are listed, take care when ordering the directives within the proxy configuration file. Prototype directives (in the form of comments) have been added to the API section of the ibmproxy.conf file. These API directives are in a purposeful order. When adding API directives to enable new features and plugin modules, order the directives as shown in the prototype section of the configuration file. Alternatively, uncomment and edit, if necessary, API directives to include support for each desired function or plugin. Add user-generated plugin modules after those supplied with the product.

Port — Specify the port on which the server listens for requests

Use this directive to specify the number of the port on which the server listens for requests. The standard port number for HTTP is 80. Other port numbers lower than 1024 are reserved for other TCP/IP applications and must not be used. Common ports used for proxy Web servers are 8080 and 8008.

When a port other than 80 is used, clients are required to include a specific port number on requests to the server. The port number is preceded by a colon (:) and placed after the host name on the URL. For example, from the browser, the URL `http://www.turfco.com:8008/` requests the default welcome page from a host named www.turfco.com that is listening on port 8008.

You can use the **-p** option on the **ibmproxy** command to override this setting when starting the server.

Format

`Port number`

If you change this directive, you must manually stop and then start your server again for the change to take effect. The server does not recognize the change if you only restart it. (See Chapter 5, "Starting and stopping the Caching Proxy," on page 15.)

Default

`Port 80`

PostAuth — Customize the PostAuth step

Use this directive to specify a customized application function that the server calls during the PostAuth step. This code is executed regardless of the return codes from previous steps or other PostAuth handlers. It allows you to clean up any resources allocated to process the request.

Format

`PostAuth /path/file:function_name`

/path/file
Specifies the fully qualified file name of the compiled program, including the extension.

function_name
Specifies the name of the application function within the program.

Example

```
AuthExit /ics/api/bin/icsext05.so:post_exit
```

Default

None

PostExit — Customize the PostExit step

Use this directive to specify a customized application function that the server calls during the PostExit step. This code is executed regardless of the return codes from previous steps or other PostExit handlers. It allows you to clean up any resources allocated to process the request.

Format

`PostExit /path/file:function_name`

/path/file
Specifies the fully qualified file name of the compiled program, including the extension.

function_name
Specifies the name of the application function within the program.

Example

```
PostExit  /ics/api/bin/icsext05.so:post_exit
```

Default

None

PreExit — Customize the PreExit step

Use this directive to specify a customized application function that the server calls during the PreExit step. This code is executed after a client request is read but before any other processing occurs. You can call the GoServe module during this step.

Format

`PreExit` *`/path/file:function_name`*

`/path/file`
Specifies the fully qualified file name of the compiled DLL, including the extension.

`function_name`
Specifies the name of the application function within the program.

Example

```
PreExit  /ics/api/bin/icsext05.so:pre_exit
```

Default

None

Protect — Activate a protection setup for requests that match a template

Use this directive to activate protection setup rules for requests that match a template.

Note: For protection to work properly, the DefProt and Protect directives must be placed before any Pass, Exec, or Proxy directives in the configuration file.

A protection setup is defined with protection subdirectives. The format of the Protect directive depends on whether you want to point to a label or file containing the protection subdirectives or to include the protection subdirectives inline as part of the Protect directive.

Format

This parameter can take the following forms:

- The Protect directive can be specified as a full path and file name for a separate file that contains the protection subdirectives. It can also be specified by a protection setup label name that matches a name defined previously on a Protection directive; the Protection directive contains the protection subdirectives. Use this format:

  ```
  Protect request_template [setup_file | label[
       [FOR Server_IP_address | host_name]
  ```

  Note: The directive must be typed on one line, even though it is shown here on two lines.

- You can specify the actual protection subdirectives inline on the Protect directive. The subdirectives must be enclosed in braces ({}). The left brace character must

be the last character on the same line as the Protect directive. Each subdirective follows on its own line. The right brace character must be on its own line following the last subdirective line. No comment lines can appear between the braces. To include the protection subdirectives inline as part of the Protect directive, the format is as follows:

```
Protect request_template [FOR Server_IP_address | hhost_name]
    subdirective  value
    subdirective  value
    .
    .
    .
  }
```

The following parameters are used:

request_template
Specifies a template for requests for which to activate protection. The server compares incoming client requests to the template and activates protection if there is a match.

[*setup_file* | *label***]**
If you are pointing to a label or file containing the protection subdirectives, this parameter specifies the protection setup to activate for requests that match *request_template*.

This parameter is optional. If it is omitted, the protection setup is defined by the most recent DefProt directive that contains a matching template.

[FOR *server_IP_address* | *host_name***]**
If you are using multiple IP addresses or virtual hosts, use this parameter to specify an IP address or a host name. The server uses the directive only for requests that come to the server on this IP address or for this host. For an IP address, this is the address of the server's network connection, not the address of the requesting client. If you protect an IP address, both the IP address and the fully qualified hostname are protected. However, if the server is called from within its network using a name other than the fully qualified hostname, such as by using an entry in a hostname file, it is not protected.

Example:

```
Protect http://x.x.x.x  PROT-ADMIN
```

Within a Web browser:

- http://x.x.x.x is protected
- http://hostname.example.com is protected
- http://hostname is not protected

Example:

```
Protect http://hostname.example.com  PROT-ADMIN
```

Within a Web browser:

- http://x.x.x.x is not protected
- http://hostname.example.com is protected
- http://hostname is not protected

You can specify an IP address (for example, `FOR 240.146.167.72`), or you can specify a host name (for example, `FOR hostA.bcd.com`).

Wildcard characters cannot be used to specify server IP addresses.

This parameter is optional. Without this parameter, the server uses the directive for all requests regardless of the IP address the requests come in on or the host name in the URL.

Note: The [*server_IP_address* | *host_name*] parameter is used with either the [*setup_file* | *label*] parameter or the *subdirective value* parameter.

- To use [*server_IP_address* | *host_name*] with [*setup_file* | *label*], you must put FOR, or some other character string (without blanks), between the [*setup_file* | *label*] parameter and the [*server_IP_address* | *host_name*] parameters.
- To use [*server_IP_address* | *host_name*] with *subdirective value* parameters, do *not* include FOR before the *IP_address* or *host_name*.

subdirective value

To include the protection subdirectives as part of the Protect directive, use this parameter. For descriptions of the protection subdirectives, see the following:

- "AuthType — Specify the authentication type" on page 236
- "DeleteMask — Specify the user names, groups, and addresses that are allowed to delete files" on page 236
- "GetMask — Specify the user names, groups, and addresses allowed to get files" on page 236
- "GroupFile — Specify the location of the associated group file" on page 237
- "Mask — Specify the user names, groups, and addresses allowed to make HTTP requests" on page 237
- "PasswdFile — Specify the location of the associated password file" on page 237
- "PostMask — Specify the user names, groups, and addresses allowed to post files" on page 237
- "PutMask — Specify the users names, groups, and addresses allowed to put files" on page 238
- "ServerID — Specify a name to associate with the password file" on page 238

Examples

- In the following example, the server activates protection as follows:
  - Requests that start with /secret/scoop/ activate protection. The protection setup is defined in the /server/protect/setup1.acc protection setup file. Because the Protect directive does not specify a protection setup, the protection setup on the previously matching DefProt directive is used.
  - Requests beginning with /secret/business/ activate protection. The protection setup is defined on the Protection directive that has a label of BUS-PROT.
  - Requests beginning with /topsecret/ activate protection. The protection setup is included directly on the Protect directive.

  These examples use IP addresses. If the server receives requests that begin with /secret/ or /topsecret/, it activates a different protection setup for the request, based on the IP address of the network connection that request comes in on.
  - For /secret/ requests coming in on 0.67.106.79, the server activates the protection setup defined on a Protection directive with a label of

CustomerA-PROT. For /topsecret/ requests coming in on 0.67.106.79, the server activates the protection setup defined inline on the Protect directive for /topsecret/.

– For /secret/ requests coming in on 0.83.100.45, the server activates the protection setup defined on a Protection directive with a label of CustomerB-PROT. For /topsecret/ requests coming in on 0.83.100.45, the server activates the protection setup defined inline on the Protect directive for /topsecret/.

```
Protection BUS-PROT {
   UserID    busybody
   GroupID   webgroup
   AuthType Basic
   ServerID restricted
   PasswdFile  /docs/WWW/restrict.pwd
   GroupFile   /docs/WWW/restrict.grp
   GetMask authors
   PutMask authors
}
DefProt  /secret/*  /server/protect/setup1.acc
Protect  /secret/scoop/*
Protect  /secret/business/*   BUS-PROT
Protect  /topsecret/*  {
   AuthType Basic
   ServerID restricted
   PasswdFile /docs/WWW/restrict.pwd
   GroupFile  /docs/WWW/restrict.grp
   GetMask topbrass
   PutMask topbrass
}
Pass   /secret/scoop/*       /WWW/restricted/*
Pass   /secret/business/*    /WWW/confidential/*
Pass   /topsecret/*          /WWW/topsecret/*

Protect   /secret/*   CustomerA-PROT   FOR  0.67.106.79
Protect   /secret/*   CustomerB-PROT   FOR  0.83.100.45
Protect  /topsecret/*   0.67.106.79  {
   AuthType Basic
   ServerID restricted
   PasswdFile /docs/WWW/customer-A.pwd
   GroupFile  /docs/WWW/customer-A.grp
   GetMask A-brass
   PutMask A-brass
}
Protect  /topsecret/*   0.83.100.45   {
   AuthType Basic
   ServerID restricted
   PasswdFile /docs/WWW/customer-B.pwd
   GroupFile  /docs/WWW/customer-B.grp
   GetMask B-brass
   PutMask B-brass
}
```

- The following examples use virtual hosts. If the server receives requests that begin with /secret/ or /topsecret/, it activates a different protection setup for the request based on the host name in the URL.

  – For /secret/ requests coming in for hostA.bcd.com, the server activates the protection setup defined on a Protection directive with a label of CustomerA-PROT. For /topsecret/ requests coming in for hostA.bcd.com, the server activates the protection setup defined inline on the Protect directive for /topsecret/.

  – For /secret/ requests coming in for hostB.bcd.com, the server activates the protection setup defined on a Protection directive with a label of

CustomerB-PROT. For /topsecret/ requests coming in for hostB.bcd.com, the server activates the protection setup defined inline on the Protect directive for /topsecret/.

– For proxied requests, the server activates the protection setup defined on a Protection directive with a label of proxy-prot. For example:

```
Protect http://host1/* proxy-prot

Protect   /secret/*   CustomerA-PROT   FOR  hostA.bcd.com
Protect   /secret/*   CustomerB-PROT   FOR  hostB.bcd.com
Protect  /topsecret/*   hostA.bcd.com  {
   AuthType Basic
   ServerID restricted
   PasswdFile /docs/WWW/customer-A.pwd
   GroupFile  /docs/WWW/customer-A.grp
   GetMask A-brass
   PutMask A-brass
}
Protect  /topsecret/*   hostB.bcd.com {
   AuthType Basic
   ServerID restricted
   PasswdFile /docs/WWW/customer-B.pwd
   GroupFile  /docs/WWW/customer-B.grp
   GetMask B-brass
   PutMask B-brass
   }
```

Default

By default, protection is provided for the Configuration and Administration forms by a Protect directive with a request template of /admin-bin/*.

Protection — Define a named protection setup within the configuration file

Use this directive to define a protection setup within the configuration file. You give the protection setup a name and define the type of protection by using protection subdirectives.

Notes:

1. In the configuration file, place Protection directives before any DefProt or Protect directives that point to them.
2. To use domain names in your protection rules, set the DNS-Lookup directive to on.

Format

```
Protection label_name  {
    subdirective  value
    subdirective  value
    .
    .
    .
  }
```

label_name
Specifies the name to associate with this protection setup. The name can then be used by subsequent DefProt and Protect directives to point to this protection setup.

subdirective value
The subdirectives are enclosed in braces ({ }). The left brace character must be the last character on the same line as the *label_name*. Each subdirective follows on its own line. The right brace character must be on its own line following the last subdirective line.. No comment lines can appear between the braces.

See “Protection subdirectives — Specify how a set of resources is protected” for descriptions of the protection subdirectives.

Example

```
Protection NAME-ME   {
   AuthType Basic
   ServerID restricted
   PasswdFile  /WWW/password.pwd
   GroupFile   /WWW/group.grp
   GetMask groupname
   PutMask groupname
}
```

Default

```
Protect /admin-bin/* {
  ServerId     Private_Authorization
  AuthType     Basic
  GetMask      All@(*)
  PutMask      All@(*)
  PostMask     All@(*)
  Mask         All@(*)
  PasswdFile   /opt/ibm/edge/cp/server_root/protect/webadmin.passwd
}
```

Protection subdirectives — Specify how a set of resources is protected

The following are descriptions of the protection subdirectives that can be used in a protection setup. The subdirectives are in alphabetical order.

The protection setups can be either in separate files or included in the configuration file as part of DefProt, Protect, or Protection directives.

AuthType — Specify the authentication type

Use this Protection subdirective when limiting access based on user names and passwords. Specify the type of authentication to use when the client sends a password to the server. With basic authentication (`AuthType Basic`), passwords are sent to the server as plain text. They are encoded, but not encrypted.

Default:

```
AuthType Basic
```

DeleteMask — Specify the user names, groups, and addresses that are allowed to delete files

Use this Protection subdirective to specify user names, groups, and address templates authorized to make DELETE requests to a protected directory.

Example:

```
DeleteMask authors,(niceguy,goodie)@45.96.3.1,128.0.*.*
```

GetMask — Specify the user names, groups, and addresses allowed to get files

Use this Protection subdirective to specify user names, groups, and address templates authorized to make GET requests to a protected directory.

Example:

```
GetMask authors,(niceguy,goodie)@45.96.3.1,128.0.*.*
```

Default:

```
GetMask  All@(*)
```

GroupFile — Specify the location of the associated group file

Use this Protection subdirective to specify the path and file name of the server group file that the protection setup uses. The groups defined within the server group file can then be used by the following:

- Any mask subdirectives that are part of the protection setup. (The mask subdirectives are DeleteMask, GetMask, Mask, PostMask, and PutMask.)
- Any ACL file on a directory that is protected by the protection setup.

Example:

```
GroupFile /docs/etc/WWW/restrict.group
```

Mask — Specify the user names, groups, and addresses allowed to make HTTP requests

Use this subdirective to specify user names, groups, and address templates authorized to make HTTP requests that are not covered by other mask subdirectives.

Examples:

```
Mask authors,(niceguy,goodie)@45.96.3.1,128.0.*.*
```

Note: When you use the Mask directive, it is important to note that Masks are case sensitive. The following is an example of a Mask protection specified on a user ID:

```
MASK WEBADM,webadm
```

PasswdFile — Specify the location of the associated password file

Use this Protection subdirective when limiting access based on user names and passwords. Specify the path and name of the password file that this protection setup is to use.

Because some browsers cache User IDs and passwords by security realm (ServerID) within a host, follow these guidelines when specifying ServerID and password files:

- For protection setups that use the same password file, use the same ServerID.
- For protection setups that use different password files, use different ServerIDs.

Example:

```
PasswdFile /docs/etc/WWW/restrict.password
```

Note: If the path or file name of the password file contains embedded blanks, then the entire path and file name must be enclosed in quotation marks ("".)

```
PasswdFile "c:\test this\admin.pwd"
```

PostMask — Specify the user names, groups, and addresses allowed to post files

For a secure server, use this Protection subdirective to specify users, groups, and address templates authorized to make POST requests to a protected directory.

Example:

```
PostMask authors,(niceguy,goodie)@45.96.3.1,128.0.*.*
```

PutMask — Specify the users names, groups, and addresses allowed to put files

Use this Protection subdirective to specify users, groups, and address templates authorized to make PUT requests to a protected directory.

Example:

```
PutMask authors,(niceguy,goodie)@45.96.3.1,128.0.*.*
```

ServerID — Specify a name to associate with the password file

Use this Protection subdirective when limiting access based on user names and passwords. Specify a name that you want to associate with the password file being used. The name does not need to be a real machine name.

The name is used as an identifier to the requester. Because different protection setups can use different password files, associating a name with the protection setup can help the client decide which password to send. Most clients display this name when prompting for a user name and password.

Because some browsers cache user IDs and passwords by security realm (ServerID) within a host, follow these guidelines when specifying ServerID and password files:

- For protection setups that use the same password file, use the same ServerID.
- For protection setups that use different password files, use different ServerIDs.

Example:

```
ServerID restricted
```

Proxy — Specify proxy protocols or reverse proxy

Use this directive to indicate which protocols the Caching Proxy is to process and map a request to a server. Valid protocols are `http`, `ftp`, and `gopher`.

The UseSession parameter option for the Proxy directive helps maintain a persistent channel on the server side if the requests from clients are forwarded to the same server and from the same client-side TCP channel. UseSession overrides the directive ServerConnPool, which allows requests from different clients to share a persistent connection to the back-end server.

The JunctionPrefix parameter option for the Proxy directive is used in conjunction with the junction rewriting plugin. For more information, refer to “Enable junction rewrite (optional)” on page 41 and “Define the junction with the JunctionPrefix option (recommended method)” on page 42.

Format

```
Proxy request_template target_server_path [[ip]:port]
      [UseSession] [JunctionPrefix:url_prefix]
```

This directive passes the request to a remote server. For example, the following directive causes all requests to be forwarded to the designated URL:

```
Proxy /*   http://proxy.server.name/*
```

For a secure reverse proxy server, use the following directive:

```
Proxy /*  https://proxy.server.name/*
```

If you want your proxy server to be less restrictive, uncomment the following directives from your configuration file. (However, these directives may introduce a security problem when proxy is configured as a reverse proxy.)

```
Proxy http:*
Proxy ftp:*
Proxy gopher:*
```

The following is an example of the UseSession option for the Proxy directive:

```
Proxy  /abc/*   http://server1/default/abc/*  :80  UseSession
```

When the incoming client request comes from port 80, and if the URL on the client request matches the pattern /abc/*, then the URL is mapped to http://server1/default/abc/* .

The following is the format for the JunctionPrefix option for the Proxy directive:

```
Proxy url_patern1 url_pattern2 JunctionPrefix:url_prefix
```

For more information on use of the JunctionPrefix parameter option, refer to "Enable junction rewrite (optional)" on page 41 and "Define the junction with the JunctionPrefix option (recommended method)" on page 42.

Defaults

None.

ProxyAccessLog — Name the path for the proxy access log file

Use this directive to specify the path and name for the file where you want the server to log access statistics for proxy requests. By default, the server writes an entry to this log each time it acts as a proxy for a client request. You can use the NoLog directive if you do not want to log requests from certain clients.

The server starts a new log file each day at midnight if it is running. Otherwise, the server starts a new log file the first time you start it on any day. When creating the file, the server uses the file name you specify and appends a date suffix or extension. The date suffix or extension is in the format *Mmmddyyyy*, where *Mmm* is the first three letters of the month; *dd* is the day of the month; and *yyyy* is the year.

It is a good idea to remove old log files, because they can consume a significant amount of space on your hard drive.

Format

```
ProxyAccessLog path/file
```

Defaults

- **Linux and UNIX systems:** `ProxyAccessLog /opt/ibm/edge/cp/server_root/logs/proxy`
- **Windows systems:** `ProxyAccessLog drive:\Program Files\IBM\edge\cp\logs\proxy`

ProxyAdvisor — Customize the servicing of proxy requests

Use this directive to specify a customized application you want the server to call during the Proxy Advisor step. This code will service the request.

Format

ProxyAdvisor */path/file*:*function_name*

/path/file
Specifies the fully qualified file name of the compiled program.

function_name
Specifies the name of the application function within the program.

Example:

```
ProxyAdvisor /api/bin/customadvise.so:proxyadv
```

Default

None

ProxyForwardLabels — Specify PICS filtering

Use the ProxyForwardLabels directive to specify PICS filtering at the proxy server and at the client, or at two proxies in a proxy hierarchy.

If ProxyForwardLabels is set to `on`, the the proxy server generates `PICS-Label:` HTTP headers for all PICS labels found, including labels from the origin server, label bureaus, the Caching Proxy's label cache, and label-supplier plugins.

If ProxyForwardLabels is set to `Off`, `PICS-Label:` HTTP headers are not generated.

Format

```
ProxyForwardLabels {on | off}
```

Default

```
ProxyForwardLabels Off
```

ProxyFrom — Specify a client with a From: header

Use this directive to generate a `From:` header. This is typically used to give an e-mail address of the proxy administrator.

Format

ProxyFrom *e-mail_address*

Example

The setting `ProxyFrom webmaster@proxy.ibm.com` results in the following header change:

Original header

```
Location: http://www.ibm.com/
Last Modified: Tue 5 Nov 1997 10:05:39
GMT
Pragma: no-cache
```

Modified Header

```
Location: http://www.ibm.com/
Last Modified: Tue 5 Nov 1997 10:05:39
GMT
From: webmaster@proxy.ibm.com
Pragma: no-cache
```

Default

None

ProxyIgnoreNoCache — Ignore a reload request

Use this directive to specify how the server reacts when users click **Reload** on the browser. If the ProxyIgnoreNoCache directive is set to `on`, during periods of high

load, the server does not request the page from the destination server and supplies the cached copy of the file if it is available. The server essentially disregards the `Pragma: no-cache` header sent from the browser.

Format

```
ProxyIgnoreNoCache  {on | off}
```

Default

```
ProxyIgnoreNoCache off
```

ProxyPersistence — Allow persistent connections

Use this directive to specify whether a persistent connection is maintained with the client. A persistent connection reduces waiting time for users and reduces the CPU load on the proxy server, but it requires more resources. More threads, and therefore more memory on the proxy server, are required for a persistent connection.

Persistent connections must not be used on a multilevel proxy server setup if any of the proxies is not HTTP 1.1 compliant.

Format

```
ProxyPersistence {on | off}
```

Default

```
ProxyPersistence on
```

ProxySendClientAddress — Generate the Client IP Address: header

Use this directive to specify whether the proxy forwards the IP address of the client to the destination server.

Format

```
ProxySendClientAddress  {Client_IP: | OFF}
```

Example

The directive `ProxySendClientAddress` *`Client-IP`*`:` results in the following header change:

Original header

```
Location: http://www.ibm.com/
Last Modified: Tue 5 Nov 1997 10:05:39
GMT
Pragma: no-cache
```

Modified Header

```
Location: http://www.ibm.com
Last Modified: Tue 5 Nov 1997 10:05:39
GMT
Client-IP: 0.67.199.5
Pragma: no-cache
```

Default

None

ProxyUserAgent — Modify User Agent string

Use this directive to specify a `User Agent` string that replaces the string that the client sends. This allows greater anonymity while visiting Web sites. However, some sites have customized pages based on the `User Agent` string. Using the ProxyUserAgent directive prevents these custom pages from being displayed.

Format

ProxyUserAgent *product_name/version*

Example

The directive ProxyUserAgent Caching Proxy/6.0 results in the following header change:

Original header	Modified header
Location: http://www.ibm.com/	Location: http://www.ibm.com
Last Modified: Tue 5 Nov 1997 10:05:39 GMT	Last Modified: Tue 5 Nov 1997 10:05:39 GMT
User Agent: Mozilla/ 2.02 OS2	**User Agent: Caching Proxy/6.0**
Pragma: no-cache	Pragma: no-cache

Default

None

ProxyVia — Specify format of HTTP header

Use this directive to control the format of the HTTP header. There are four possible values for this directive. If ProxyVia is set to Full, the Caching Proxy adds a Via header into the request or reply; if a Via header is already in the stream, the Caching Proxy adds host information at the end. If set to Set, the Caching Proxy sets the Via header to host information; if a Via header is already in the stream, the Caching Proxy removes it. If set to Pass, the Caching Proxy forwards any header information as is. If set to Block, the Caching Proxy does not forward any Via header.

Format

ProxyVia {Full | Set | Pass | Block}

Example

ProxyVia Pass

Default

ProxyVia Full

ProxyWAS — Specify that requests are sent to WebSphere Application Server

The ProxyWAS mapping directive works identically to the Proxy directive, but also indicates to the Caching Proxy that matching requests are directed to a WebSphere Application Server. For examples on using this directive, see “Proxy — Specify proxy protocols or reverse proxy” on page 238.

Format

ProxyWAS *request_template target_server_path* [UseSession]
[JunctionPrefix:*url_prefix*]

Default

None

PureProxy — Disable a dedicated proxy

Use this directive to specify whether the server is acting as a proxy server or as a proxy and content server. It is recommended that you use the Caching Proxy as a proxy only.

Format

```
PureProxy {on | off}
```

Default

```
PureProxy on
```

PurgeAge — Specify the age limit for a log

Use this directive to specify the age of a log, in days, before it is purged. If PurgeAge is set to `0`, the log is never deleted.

Note: The plugin never deletes the log for the current day or the preceding day.

Format

```
PurgeAge number
```

Default

```
PurgeAge 7
```

Related directives

- "CompressAge — Specify when to compress logs" on page 185
- "CompressDeleteAge — Specify when to delete logs" on page 186
- "CompressCommand — Specify the compression command and parameters" on page 185
- "Midnight — Specify the API plugin used to archive logs" on page 221
- "LogArchive — Specify the behavior of log archiving" on page 215
- "PurgeSize — Specify the limit for the size of the log archive"

PurgeSize — Specify the limit for the size of the log archive

Use this directive to specify how large, in megabytes, the log files can grow before the log archive is purged. If the PurgeSize directive is set to `0`, there is no size limit and no files are deleted.

The setting for PurgeSize refers to *all* the logs of a log type. For instance, if you are logging errors (that is, if an ErrorLog entry has been made in the configuration file) and PurgeSize is defined as 10 MB, the Caching Proxy calculates the sizes of all the error logs, adds them together, then deletes logs until the total size is less than 10 MB.

Note: The plugin never deletes the log for the current day or the preceding day. When log files are deleted, the oldest logs are deleted first, until the size of each log type's log files is less than or equal to the value defined by PurgeSize (in megabytes).

Format

```
PurgeSize number_of_MB
```

Default

```
PurgeSize 0
```

Related directives

- "CompressAge — Specify when to compress logs" on page 185
- "CompressDeleteAge — Specify when to delete logs" on page 186

- "CompressCommand — Specify the compression command and parameters" on page 185
- "LogArchive — Specify the behavior of log archiving" on page 215
- "Midnight — Specify the API plugin used to archive logs" on page 221
- "PurgeAge — Specify the age limit for a log" on page 243

RCAConfigFile — Specify an alias for ConfigFile

Use this directive to specify the name and location of the Remote Cache Access configuration file.

Note: The RCA configuration file has been merged into the file ibmproxy.conf. For backward compatibility, RCAConfigFile is supported as an alias for ConfigFile.

Format

```
RCAConfigFile /etc/file_name
```

Example

```
RCAConfigFile /etc/user2rca.conf
```

Default

```
RCAConfigFile /etc/rca.conf
```

RCAThreads — Specify the number of threads per port

Use this directive to specify the number of threads working on an RCA port.

Format

```
RCAThreads number_of_threads
```

Example

```
RCAThreads 50
```

Default

```
MaxActiveThreads x [(ArraySize -1) / (2 x ArraySize -1)]
```

ReadTimeout — Specify the time limit for a connection

Use this directive to specify the time limit allowed without network activity before a connection is canceled.

Format

```
ReadTimeout time
```

Default

```
ReadTimeout 5 minutes
```

Redirect — Specify a template for requests sent to another server

Use this directive to specify a template for requests that you want to accept and send to another server. After a request matches a template on a Redirect directive, the request is not compared to templates on any other directives in the configuration file.

Format

```
Redirect request_template URL  [server_IP_address | host_name]
```

request_template
Specifies a template for requests that you want your server to send to another server.

You can use an asterisk (*) as a wildcard in the template. The tilde character (~) just after a slash (/) must be explicitly matched; a wildcard cannot be used to match it.

URL
Specifies the URL request that the server sends to another server. The response to this request goes to the original requester without any indication that it did not come from your server.

URL must contain a protocol specification and the name of the server to which the request is sent. It can also contain a path or file name. If *request_template* uses a wildcard, the path or file name on *URL* can also use a wildcard. The part of the original request that matches the wildcard on *request_template* is inserted in place of the wildcard on *URL*.

[*server_IP_address* | *host_name*]
If you are using multiple IP addresses or virtual hosts, use this parameter to specify an IP address or a host name. The server uses the directive only for requests that come to the server on this IP address or for this host. For an IP address, this is the address of the server's network connection, not the address of the requesting client.

You can specify an IP address (for example, `240.146.167.72`) or you can specify a host name (for example, `hostA.bcd.com`).

This parameter is optional. Without this parameter, the server uses the directive for all requests regardless of the IP address the requests come in on or the host name in the URL.

A wildcard character cannot be specified for a server's IP address.

Examples

- In the following example, the server sends any requests beginning with `/chief/stuff/` to the wahoo directory of the www.other.org server.

```
Redirect  /chief/stuff/*   http://www.other.org/wahoo/*
```

- The following examples use the optional IP address parameter. If your server receives requests that begin with `/stuff/`, it redirects the request to different servers based on the IP address of the network connection on which the request comes in. For requests coming in on 240.146.167.72, the server sends the request to the wahoo directory of the www.chief.org server. For requests coming in on any connection with an address of 0.83.100.45, the server sends the request to the pound directory of the www.dawg.com server.

```
Redirect  /stuff/*   http://www.chief.org/wahoo/*   240.146.167.72
Redirect  /stuff/*   http://www.dawg.com/pound/*    0.83.100.45
```

- The following examples use the optional IP address parameter. If your server receives requests that begin with `/stuff/`, it redirects the request to different servers based on the host name in the URL. For requests coming in for hostA, the server sends the request to the wahoo directory of the www.chief.org server. For requests coming in for hostB, the server sends the request to the pound directory of the www.dawg.com server.

```
Redirect  /stuff/*   http://www.chief.org/wahoo/*   hostA.bcd.com
Redirect  /stuff/*   http://www.dawg.com/pound/*    hostB.bcd.com
```

Default

None

RegisterCacheIdTransformer — Cache more than one variant of a resource based on the Cookie header

Use this directive to allow Caching Proxy to cache more than one variant of a resource (URI) based on the Cookie header.

Note: If cookies are disabled on the client browsers, clients can access the same cached object.

For more information, see "SupportVaryHeader — Cache more than one variant of a resource based on the HTTP Vary header" on page 257.

Format

```
RegisterCacheIdTransformer Cookie cookie-name
```

The *cookie-name* is the name in the Cookie header in the client's request.

Example

```
RegisterCacheIdTransformer Cookie Usergroup
```

For an example of using this directive in conjunction with SupportVaryHeader, see"SupportVaryHeader — Cache more than one variant of a resource based on the HTTP Vary header" on page 257.

Default

None

ReversePass — Intercept automatically redirected requests

The ReversePass mapping directive examines the server response stream to detect requests that are rewritten as a result of automatic redirection. Typically, when a server returns an HTTP code in the 3*xx* class (for example, 301, moved permanently, or 303, see other), the server sends a message with the reply that instructs the requesting client to direct future requests to the correct URL and IP address. In the case of a reverse proxy setup, a redirection message from the origin server can cause client browsers to bypass the proxy server for subsequent requests. To avoid having clients directly contact the origin server, use the ReversePass directive to intercept requests that are made specifically to the origin server.

Unlike other mapping directives, which process the request stream, ReversePass matches its template to the response stream. The response stream is the reply that the proxy server obtains from the origin server and sends to the client.

Format

```
ReversePass rewritten_URL proxy_URL [host:port]
```

The *host*:*port* option allows the proxy to apply a different ReversePass rule based on the backend server hostname and port.

Examples

- The following example statement prevents direct requests to the origin server:

  ```
  ReversePass  http://backend.company.com:9080/*  http://edge.company.com/*
  ```

  The port 9080 is the default port for Application Service at the Edge. This type of request might be generated if the origin application server returned a 3*xx* code to the client.

- The following example statement catches requests that were redirected by a 301 code from the edge application: server.

```
ReversePass  http://edge.company.com:9080/*  http://edge.company.com/*
```

Note: The contents of the *proxy_URL* pattern, up to the wild card (*), must match exactly what the back-end server sends in the location header or the directive fails.

Default
None

RewriteSetCookieDomain — Specify the domain pattern that needs to be rewritten

Use this directive to specify the domain pattern that needs to be rewritten. The directive will transform the domain from *domain_pattern1* to *domain_pattern2*.

Format
```
RewriteSetCookieDomain domain_pattern1 domain_pattern2
```

Example
```
RewriteSetCookieDomain .internal.com .external.com
```

Default
None

Related directives
- "JunctionRewriteSetCookiePath — Rewrite the path option in the Set-Cookie header, when used with JunctionRewrite plugin" on page 211

RTSPEnable — Enable RTSP redirection

This directive enables or disables RTSP redirection. Options are `on` or `off`.

Format
```
RTSPEnable {on | off}
```

Example
```
RTSPEnable  on
```

Default
None

rtsp_proxy_server - Specify servers for redirection

This directive is used to specify RTSP proxy servers to receive redirected requests. Different servers can be specified for different types of streams. The format of this directive is:

```
rtsp_proxy_server server dns address[:port] default rank [list of mime types]
```

Example
```
rtsp_proxy_server    rproxy.mycompany.com:554     1
rtsp_proxy_server    fw1.mycompany.com:554        2
rtsp_proxy_server    fw1.mycompany.com:555        3
rtsp_proxy_server    fw2.mycompany.com:557        4
```

Default
None

rtsp_proxy_threshold — Specify number of requests before redirection to a cache

This directive specifies how many requests are received before an RTSP request is redirected to a proxy server rather than to the origin server. RealNetworks proxies cache streams on the first request, and caching initially incurs at double the bandwidth of receiving a stream. Specifying a threshold greater than one prevents requests made only once from being cached. The format of this directive is:

```
rtsp_proxy_threshold number_of_hits
```

Example

```
rtsp_proxy_threshold 5
```

Default

None

rtsp_url_list_size — Specify number of URLs in proxy memory

This directive specifies the number of unique URLs that are kept in memory for redirection. The proxy refers to this list to determine whether a given URL has been encountered before. Larger list sizes improve the proxy server's ability to send a subsequent request to the same proxy server that received the previous request, but each list entry consumes approximately 16 bytes of memory.

Format

```
rtsp_url_list_size size_of_list
```

Example

```
rtsp_url_list_size 8192
```

Default

None

ScriptTimeout – Specify the timeout setting for scripts

Use this directive to set the time allowed for a CGI program started by the server to finish. When the time expires, the server ends the program. On Linux and UNIX platforms, this is done with the KILL signal.

Enter the time value using any combination of hours, minutes (or mins), and seconds (or secs).

Format

```
ScriptTimeout timeout
```

Default

```
ScriptTimeout 5 minutes
```

SendHTTP10Outbound — Specify the protocol version for proxied requests

Use this directive to specify that requests sent from the Caching Proxy to a downstream server are to use the HTTP Version 1.0 protocol. (A *downstream* server is another proxy server in a chain of proxies or an origin server that processes the request.)

If this directive is used, the Caching Proxy identifies HTTP 1.0 as the protocol in the request line. Only HTTP 1.0–specific functionality and certain HTTP 1.1

functions, such as cache-control headers, which are supported by most HTTP 1.0 servers, are sent to the downstream server. Use this directive if you have a downstream server that does not handle HTTP 1.1 requests correctly.

If the SendHTTP10Outbound directive is *not* specified, the Caching Proxy identifies HTTP 1.1 as the protocol in the request line. HTTP 1.1 functionality, such as persistent connections, can also be used in the request.

Format

```
SendHTTP10Outbound url_pattern
```

Examples

This directive can be specified multiple times, for example:

```
SendHTTP10Outbound http://www.hosta.com/*
SendHTTP10Outbound http://www.hostb.com/*
```

For backward compatibility, the previous syntax of SendHTTP10Outbound is handled as follows:

- `SendHTTP10Outbound on` is treated as if `SendHTTP10Outbound *` was specified.
- `SendHTTP10Outbound off` is ignored.

Note: If both `SendHTTP10Outbound off` and `SendHTTP10Outbound url_pattern` are specified, then `SendHTTP10Outbound off` is ignored, but a warning message is issued.

Default

None

SendRevProxyName — Specify the Caching Proxy host name in the HOST header

When functioning as a reverse proxy, the Caching Proxy receives HTTP requests from a client and sends the requests to the origin server. By default, the Caching Proxy writes the origin server's host name in the HOST header of the request that it sends to the origin server. If the SendRevProxyName directive is set to yes, the Caching Proxy writes its own host name in the HOST header instead. This directive can be used to enable special configuration for back-end servers because it allows the request to the origin server to always appear to come from the proxy server, even in the case where the request is redirected from one back-end server to another.

This directive differs from the ReversePass mapping directive as follows: the ReversePass directive intercepts requests with a specified syntax and substitutes different request content that you specify. The SendRevProxyName directive can be set only to substitute the Caching Proxy host name for the origin server host name. This directive is not useful in configuring Application Service at the Edge.

Format

```
SendRevProxyName  {yes | no}
```

ServerConnGCRun — Specify the interval at which to run garbage collection thread

This directive sets the interval at which the garbage collection thread check for server connections that have timed out (set with the ServerConnTimeout directive). Use this directive only if the ServerConnPool directive is set to `on`.

Format

ServerConnGCRun *time_interval*

Example

```
ServerConnGCRun 2 minutes
```

Default

```
ServerConnGCRun 2 minutes
```

ServerConnPool — Specify the pooling of connections to origin servers

This directive allows the proxy to pool together its outgoing connections to origin servers. Setting this directive to on enhances performance and takes better advantage of origin servers that allow persistent connections. You can also specify how long to maintain an unused connection via the ServerConnTimeout directive.

Note: This directive is best enabled in a controlled environment; it can degrade performance in a forward proxy situation or one where the origin servers are not HTTP 1.1 compliant.

Format

```
ServerConnPool {on | off}
```

Default

```
ServerConnPool off
```

ServerConnTimeout — Specify maximum inactive period

Use this directive to limit the time allowed without network activity before cancelling the connection. Use this directive only if the ServerConnPool directive is set to on.

Format

ServerConnTimeout *time-spec*

Example

```
ServerConnTimeout 30 seconds
```

Default

```
ServerConnTimeout 10 seconds
```

ServerInit — Customize the Server Initialization step

Use this directive to specify a customized application function that the server calls during its initialization routines. This code is executed before any client requests are read and whenever the server is restarted.

If you are using the GoServe modules in the PreExit or Service steps, you need to call the gosclone module here.

Format

ServerInit */path/file*:*function_name* [*initialization_string*]

/path/file
Specifies the fully qualified file name of the compiled program, including the extension.

function_name
Specifies the name of the application function within your program.

initialization_string
Optional; specifies a text string that is passed to the application function.

Example
```
ServerInit   /ics/api/bin/icsext05.so:svr_init
```

Default
None

ServerRoot — Specify the directory where the server program is installed

Use this directive to specify the directory where the server program is installed (the current working directory of the server). Logging directives use this current working directory as the default root when relative path names are used.

On Windows systems, the directory is identified during installation.

Format
```
ServerRoot  directory_path
```

Defaults
- **Linux and UNIX systems:** `ServerRoot /opt/ibm/edge/cp/server_root/`
- **Windows systems:** `C:\Program Files\IBM\edge\cp\bin\`

Note: You can change the default, but it has no effect on how the server processes requests.

Note: PASS and EXEC rules can be independent of this directory.

ServerTerm — Customize the Server Termination step

Use this directive to specify a customized application function that the server calls during the Server Termination step. This code is executed when an orderly shutdown occurs and whenever the server is restarted. It allows you to release resources allocated by a PreExit application function.

Format
```
ServerTerm  /path/file:function_name
```

/path/file
Specifies the fully qualified file name of the compiled program, including the extension.

function_name
Specifies the name of the application function within your program.

Example
```
ServerTerm  /ics/api/bin/icsext05.so:shut_down
```

Default
None

Service — Customize the Service step

Use this directive to specify a customized application function that the server calls during the Service step. This code services the client request. For example, it sends the file or runs the CGI program.

There is no default for this directive. If the request matches a Service rule (that is, if an application function specified on a Service directive is executed), but the function returns HTTP_NOACTION, the server generates an error and the request fails.

Format

```
Service request_template/path/file:function_name
[server_IP_address | host_name]
```

request_template
Specifies a template for requests that further determine whether the application function is called. The specification can include the protocol, domain, and host; it can be preceded by a slash (/) and can use an asterisk (*) as a wildcard. For example, `/front_page.html`, `http://www.ics.raleigh.ibm.com`, `/pub*`, `/*`, and `*` are all valid.

/path/file
Specifies the fully qualified file name of the compiled program, including the extension.

function_name
Specifies the name of the application function within your program.

[*Server_IP_address* | *host_name*]
If you use multiple IP addresses or virtual hosts, this parameter determines whether your application function is called only for requests coming in on a specific IP address or for a specific host.

A wildcard character cannot be specified for a server's IP address.

Examples

```
Service  /index.html /ics/api/bin/icsext05.so:serve_req
Service  /cgi-bin/hexcalc* /ics/api/calculator:HEXcalc*
```

Note: If you want full path translation, including *query_string*, you must have an asterisk (*) in both the *request_template* and in the *path/file:function_name*, as shown in the second example.

Default

None

SignificantURLTerminator — Specify a terminating code for URL requests

Use this directive to specify a terminating code for URL requests. Using the terminating code in a request causes the Caching Proxy to evaluate only characters before the terminating code when processing the request and when evaluating whether the result is already cached. When more than one terminator code is defined, the Caching Proxy compares incoming URLs against the terminator codes in the order in which they are defined in the ibmproxy.conf file.

Format

```
SignificantURLTerminator terminating_string
```

Example

```
SignificantURLTerminator  &.
```

In this example, the two requests that follow are treated as identical.

```
http://www.exampleURL.com/tx.asp?id=0200&.;x=004;y=001
http://www.exampleURL.com/tx.asp?id=0200&.;x=127;y=034
```

Default

None

SMTPServer (Windows only)— set an SMTP server for the sendmail routine

Use this directive to set the SMTP server used by the internal sendmail routine within the Caching Proxy for Windows. The following two directives must also be set for this routine: "WebMasterEMail — Set an e-mail address to receive select server reports" on page 262 and "WebMasterSocksServer (Windows only)— set a socks server for the sendmail routine" on page 263.

Format

```
SMTPServer IP address or hostname of SMTP server
```

Example

```
SMTPServer mybox.com
```

Default

```
None
```

SNMP — Enable and disable SNMP support

Use this directive to enable or disable SNMP support.

Format

```
SNMP   {on | off}
```

Default

```
SNMP off
```

SNMPCommunity — Provide a security password for SNMP

Use this directive to define the password between the Web server Distributed Protocol Interface (DPI) subagent and the SNMP agent. The SNMP community name authorizes a user to view the performance variables monitored by SNMP for a specified community of servers. The system administrator defines which variables from which servers can be viewed when a password is entered. If you change the SNMP community name, be sure to also change the community name specified in the file /etc/snmpd.conf.

Format

```
SNMPCommunity name
```

Default

```
SNMPCommunity public
```

SSLCaching — Enable caching for a secure request

Use this directive to cache content on a secure request when a reverse proxy is used. This directive configures caching for all connections to the proxy server, both client connections and connections with a back-end content server.

Note: The SSL directives are not supported on SUSE Linux.

Format

```
SSLCaching {on | off}
```

Default

```
SSLCaching off
```

SSLCertificate — Specify key labels for certificates

Use this directive to specify key labels that allow the proxy to determine which certificate to send to the client when the Caching Proxy is acting as a single reverse proxy for multiple domains that offer their own SSL certificates and to instruct the proxy server to either retrieve or not retrieve a client-side PKI certificate for client authentication..

Format

```
SSLCertificate serverIP/hostname CertificateLabel
     [NoClientAuth | ClientAuthRequired]
```

serverIP/hostname
You can specify an IP address (for example, 204.146.167.72) or you can specify a host name (for example, hostA.raleigh.ibm.com) for the server to which the SSL request is directed.

CertificateLabel
The name of the certificate which is to be used if client authentication is required for SSL requests directed to the designated IP address or host name.

[NoClientAuth | ClientAuthRequired]
Instructions to the proxy server to either retrieve or not retrieve a client-side PKI certificate.

Examples

```
SSLCertificate  www.abc.com      ABCCert
SSLCertificate  204.146.167.72   intABCCert
SSLCertificate  www.xyz.com      XYZCert
SSLCertificate  www.xyz.com      XYZCert      ClientAuthRequired
```

Default

None

SSLCryptoCard — Specify the installed cryptographic card

Use this directive to tell the proxy server that there is a cryptographic card installed and to specify the card.

Format

```
SSLCryptoCard {rainbowcs | nciphernfast} {on | off}
```

Example

```
SSLCryptoCard rainbowcs on
```

Default
None

SSLEnable — Specify listening on port 443 for secure requests

Use this directive to specify that the Caching Proxy listens on port 443 for secure requests.

Note: The SSL directives are not supported on SUSE Linux.

Format
```
SSLEnable {on | off}
```

Default
```
SSLEnable off
```

SSLForwardPort — Specify which port to address for HTTP SSL upgrades

Use this directive to specify the port to address for HTTP requests that the Caching Proxy upgrades to HTTPS requests by implementing SSL. Specify a port other than the main HTTP port 80 or the main SSL port 443.

Format
```
SSLForwardPort port number
```

Example
```
SSLForwardPort 8888
```

Default
None

SSLOnly — Disable listener threads for HTTP requests

Use this directive to disable listener threads for standard HTTP requests (typically ports 80 and 8080) when SSL (typically port 443) is enabled.

Format
```
SSLOnly    {on | off}
```

Default
```
SSLOnly   off
```

SSLPort — Specify HTTPS listening port other than default

Use this directive to specify the HTTPS listening port other than ibmproxy's default HTTPS port 443.

Note: The ibmproxy supports one HTTPS port for each instance, so the directive should NOT be used to specify multiple HTTPS ports. To support multiple HTTPS ports, you must start multiple ibmproxy instances with different `ibmproxy.conf` files.

Format
```
SSLPort port value
```

Where the *port value* is an integer value greater than 0. Also the *port value* should be allowed by the operating system and should not be used by any other application.

Example
```
SSLPort 8443
```

Default
443

SSLTunneling — Enable SSL tunneling

Use this directive to allow SSL tunneling to any port on the destination host. Setting this directive on allows SSL tunneling to any port on the destination server. Setting this directive off allows SSL tunneling only to ports given in Proxy rules. If there are no Proxy rules for SSL tunneling, and the SSLTunneling directive is set to `off`, then SSL tunneling is not allowed. If the SSLTunneling directive is set to `on`, you must also enable the method CONNECT, using the directive Enable.

When using Caching Proxy as a reverse proxy, disabling this directive (default) protects against SSL tunneling vulnerability attacks.

Note: Use the Proxy directive to enable SSL tunneling to a specific port on the destination host.

Format
```
SSLTunneling {on | off}
```

Default
```
SSLTunneling off
```

SSLVersion — Specify the version of SSL

Use this directive to specify the version of SSL to use: V2, V3, or all versions. Set this directive to `V2` if you are using servers that cannot support SSL Version 3.

Note: The SSL directives are not supported on SUSE Linux.

Format
```
SSLVersion  {SSLV2 | SSLV3 | all}
```

Default
```
SSLVersion SSLV3
```

SSLV2Timeout — Specify the time to wait before a SSLV2 session expires

Use this directive to specify in seconds how long an SSL version 2 session waits with no activity before the session expires.

Note: The SSL directives are not supported on SUSE Linux.

Format
`SSLV2Timeout` *`seconds`*

where *seconds* represents a value between `0` and `100`.

Default

```
SSLV2Timeout 100
```

SSLV3Timeout — Specify the time to wait before a SSLV3 session expires

Use this directive to specify in seconds how long an SSL version 3 session waits with no activity before it expires.

Note: The SSL directives are not supported on SUSE Linux.

Format

```
SSLV3Timeout seconds
```

where *seconds* represents a value between 1 and 86400 seconds (which is 1 day in seconds).

Default

```
SSLV3Timeout 100
```

SuffixCaseSense — Specify whether suffix definitions are case sensitive

Use this directive to specify whether you want the server to distinguish between uppercase and lowercase letters when file suffixes are compared to the suffix patterns on the AddClient, AddCharSet, AddType, AddEncoding, and AddLanguage directives. By default, the server does not distinguish between cases.

Format

```
SuffixCaseSense  {on | Off}
```

Default

```
SuffixCaseSense  Off
```

SupportVaryHeader — Cache more than one variant of a resource based on the HTTP Vary header

Use this directive to allow Caching Proxy to cache more than one variant of a resource (URI) based on the HTTP Vary header.

When the SupportVaryHeader directive is enabled, the proxy forms a cache ID based on the URI and the selected header values in the client request.

The names of the selected headers are specified in the Vary header sent in a prior response from the server. If the server changes the set of selected header names for a resource, then all the previous cached objects for the resource are removed from the proxy's cache.

This directive can be used with the RegisterCacheIdTransformer directive ("RegisterCacheIdTransformer — Cache more than one variant of a resource based on the Cookie header" on page 246).

When both directives are used, the proxy creates an internal Cache ID transformer based on the Vary header from the server and client's request header. In this way, the proxy can generate unique cache identifiers for different request and response pairs, even if the requested URIs are the same.

The cached objects of the same URI have their own default life time in the cache, depending on the Expire and Cache-Control headers in the requests/responses, or other configuration settings. If the Dynacache plug-in is used, all the multiple presentations associated to the same URI become not valid together in the proxy's cache.

Format

```
SupportVaryHeader {on | off}
```

Example

For this example, the following directives are enabled and configured in ibmproxy.conf as follows:

```
SupportVaryHeader on
RegisterCacheIdTransformer Cookie UserGroup
```

The client Guest accesses the proxy server with

```
URI [<code>] http://www.dot.com/group.jpg [</code>]
```

and the following request/response:

```
GET /group.jpg HTTP/1.1
Host: www.dot.com
Cookie: UserGroup=Guest
Accept-Language: en_US

HTTP/1.1 200
Server: my-server
Vary: Accept-Language
.......
```

Next, client Admin accesses the proxy server with the same URI

```
http://www.dot.com/group.jpg
```

and the following request/response:

```
GET /group.jpg HTTP/1.1
Host: www.dot.com
Cookie: UserGroup=Admin
Accept-Language: fr_FR

HTTP/1.1 200
Server: my-server
Vary: Accept-Language
.......
```

As a result, if the responses are cacheable, the proxy server generates two different cache IDs:

```
1. CacheID(URI, "Guest", "en_US")
2. CacheID(URI, "Admin", "fr_FR")
```

The proxy server stores two different variants of the response from the server in the cache. Subsequently, when any client requests the resource (.../group.jpg), with either combination of language preference and user group values, the proxy server retrieves the appropriate variant of the resource from the cache and serves it.

Default

```
SupportVaryHeader off
```

TLSV1Enable — Enable Transport Layer Secure protocol

Use this directive to enable the TLS version 1 protocol in SSL connections. After this directive is turned on, the SSL connection checks the TLS protocol first, then the SSLv3 protocol, and the SSLv2 protocol last.

Note: This directive works with Internet Explorer and other browsers, but not with Netscape. (Netscape is not a recommended browser for use with Caching Proxy.)

Format

```
TLSV1Enable {on | off}
```

Example

```
TLSV1Enable on
```

Initial configuration setting

None

Transmogrifier — Customize the Data Manipulation step

Use this directive to specify a customized application function that the server calls during the Data Manipulation step. This code provides three application functions:

- An *open* function to perform any initialization prior to processing the data
- A *write* function to process the data
- A *close* function to perform any clean up activities
- An *error* function to provide notification of problems that occurred

You can have multiple Transmogrifiers active for each instance of the server.

Format

```
Transmogrifier /path/file:function_name:function_name:function_name
```

/path/file
Specifies the fully qualified file name of the compiled program, including the extension.

function_name
Specifies the name you give the application function within your program. You must supply the name of the open, write, and close functions.

Example

```
Transmogrifier /ics/bin/icsext05.so:open_data:write_data:close_data
```

Default

None

TransmogrifiedWarning — Send warning message to client

Use this directive to send a message to the client informing it that the data:

Format

```
transmogrifiedwaning {yes|no}
```

Default

Yes

TransparentProxy — Enable transparent proxy on Linux or AIX

Linux and AIX only. Use this directive to specify whether the server can run as a transparent proxy server.

Note: When the TransparentProxy directive is set to `On`, the BindSpecific directive is ignored and defaults to `Off`. Because most HTTP Traffic flows on Port 80, it is highly recommended that it is one of the configured ports.

Format

```
TransparentProxy  {on | off}
Port 80
```

Default

```
TransparentProxy  Off
```

Note: After starting transparent proxy, if you want to stop the Caching Proxy server, you must also issue the following command as root:

```
ibmproxy -unload
```

On Linux systems, this command removes the redirection Firewall rules, and on AIX systems, it unloads the Transparent Proxy Kernel Extension. If you do not issue this command after stopping the server, your machine will accept requests that are not destined for it.

UpdateProxy — Specify the cache destination

Use this directive to specify which proxy server the cache agent updates. This is required when the cache agent needs to update a proxy server other than the local proxy server on which the cache agent is running. Optionally, you can specify the port.

Note: On Linux and UNIX platforms, this directive is required for using the cache agent. If you are using only one machine for the proxy, specify the host name.

Although the cache agent can update the cache on another server, it cannot retrieve the cache access log from that machine. Therefore, if the UpdateProxy directive specifies a host other than the local host, the LoadTopCached directive is ignored.

Format

```
UpdateProxy fully_qualified_host_name_of_proxy_server
```

Example

```
UpdateProxy proxy15.ibm.com:1080
```

Default

None

UserId — Specify the default user ID

Use this directive to specify the user name or number to which the server changes before accessing files.

If you change this directive, you must manually stop your server and then start it again for the change to take effect. The server does not recognize the change if you only restart it. (See Chapter 5, “Starting and stopping the Caching Proxy,” on page 15.)

Note: If you change the server defaults for the user ID, group ID, or log directory paths, create the new directories and update the permissions and ownership of the directories. To enable the server to write information to a user-defined log directory, set the permission for that directory as 755, and set the user-defined server user ID as the owner. For example, if you change the user ID of the server from the default to jdoe, and the default logs directory to server_root/account, then the server_root/account directory must have the permission 755 and be owned by jdoe.

Format

```
UserId  {ID_name | number}
```

Default

AIX, Linux, Solaris: `UserId nobody`

HP-UX: `UserId www`

V2CipherSpecs — List the supported cipher specifications for SSL Version 2

This directive lists the available cipher specification for SSL Version 2.

Note: The SSL directives are not supported on SUSE Linux.

Format

```
V2CipherSpecs specification
```

Acceptable values are any combination of the following. None can be used twice.

- `1` — RC4 US
- `2` — RC4 Export
- `3` — RC2 US
- `4` — RC2 Export
- `6` — DES 56-bit
- `7` — Triple DES US
- `NULL` — Default cipher specifications are used

Examples

- For the U.S.: `V2CipherSpecs '137624'`
- For export: `V2 Cipherspecs '246'`

Default

None (SSL is disabled by default.)

V3CipherSpecs — List the supported cipher specifications for SSL Version 3

This directive lists available cipher specifications for SSL Version 3.

Note: The SSL directives are not supported on SUSE Linux.

If the FIPSenable directive is set ″on″, the V3CipherSpecs directive will be ignored. For more information, see “FIPSEnable — Enable Federal Information Processing Standard (FIPS) approved ciphers for SSLV3 and TLS” on page 201.

Format

```
V3CipherSpecs specification
```

Acceptable values include the following:

- `00` — NULL NULL
- `01` — NULL MD5
- `02` — NULL SHA
- `03` — RC4 MD5 Export
- `04` — RC4 MD5 US
- `05` — RC4 SHA US
- `06` — RC2 MD5 Export
- `09` — DES SHA Export
- `0A` — Triple DS SHA US
- `62` — 56-bit DES CBC SHA
- `64` — 56-bit RC4 SHA
- `NULL` — Default cipher specs are used.

Examples

- For the U.S.: `V3CipherSpecs '0A09060564620403020100'`
- For export: `V3Cipherspecs '0906646203020100'`

Default

None (SSL is disabled by default.)

WebMasterEMail — Set an e-mail address to receive select server reports

Use this directive to set an e-mail address at which to receive select Caching Proxy reports, such as a notice 30 days prior to the expiration of an SSL certificate. On Linux and UNIX systems, a sendmail process must be running. For Windows systems, the sendmail process is built into the Caching Proxy, so no external mail server is required; however, two additional directives must be set: “WebMasterSocksServer (Windows only)— set a socks server for the sendmail routine” on page 263 and “SMTPServer (Windows only)— set an SMTP server for the sendmail routine” on page 253.

Note: This e-mail address is also used as the anonymous FTP password.

Format

```
WebMasterEMail webmastermailaddress
```

Example

```
WebMasterEmail webmaster@computer.com
```

Default

```
WebMasterEmail webmaster
```

WebMasterSocksServer (Windows only)— set a socks server for the sendmail routine

Use this directive to set the socks server used by the internal sendmail routine within the Caching Proxy for Windows. The following two directives must also be set for this routine: "WebMasterEMail — Set an e-mail address to receive select server reports" on page 262 and "SMTPServer (Windows only)— set an SMTP server for the sendmail routine" on page 253.

Format

```
WebMasterSocksServer IP address or hostname of socks server
```

Example

```
WebMasterSocksServer socks.mybox.com
```

Default

```
None
```

Welcome — Specify the names of welcome files

Use this directive to specify the name of a welcome file that the server looks for to respond to requests that do not contain a specific file name. You can build a list of welcome files by putting multiple occurrences of this directive in the configuration file.

For requests that do not contain a file name or a directory name, the server always looks in the file root directory for a file that matches a name specified on a Welcome directive. If a match is found, the file is returned to the requester.

For requests that contain a directory name but not a file name, the AlwaysWelcome directive controls whether the server looks in the directory for a welcome file to return. By default, AlwaysWelcome is set to a value of `On`. This means the server always looks in the requested directory for a file matching a name specified on a Welcome directive. If a match is found, the file is returned to the requester.

If the server finds more than one match between files in a directory and file names on Welcome directives, the order of the Welcome directives determines which file is returned. The server uses the Welcome directive closest to the beginning of the configuration file.

Format

```
Welcome file_name [server_IP_address | host_name]
```

file_name
Specifies the name of a file that you want to define as a welcome file.

[*server_IP_address* | *host_name*]
If you are using multiple IP addresses or virtual hosts, use this parameter to specify an IP address or a host name. The server uses the directive only for requests that come to the server on this IP address or for this host. For an IP address, this is the address of the server's network connection, not the address of the requesting client.

You can specify an IP address (for example, `240.146.167.72`), or you can specify a host name (for example, `hostA.bcd.com`).

This parameter is optional. Without this parameter, the server uses the directive for all requests, regardless of the IP address on which the requests come in or the host name in the URLs.

A wildcard character cannot be specified for a server's IP address.

Examples

- The following example defines two welcome pages and assumes the AlwaysWelcome directive is set to its default of `On`. For requests that do not contain a file name, the server tries to return a welcome file from the directory specified on the request (or the file root directory if the request does not specify a file name or a directory). The server first looks for a file named letsgo.html. If there is no file with that name in the directory, the server looks for a file named Welcome.html.

  ```
  Welcome letsgo.html
  Welcome Welcome.html
  ```

- In the following example, the server looks for different welcome files based on the IP address of the network connection the request comes in on. For requests coming in on 0.67.106.79, the server looks for welcome files named CustomerA.html. For requests coming in on 0.83.100.45, the server looks for welcome files named CustomerB.html. If the request comes in on a different IP address, the server looks for the default address.

  ```
  Welcome CustomerA.html  0.67.106.79
  Welcome CustomerB.html  0.83.100.45
  ```

- In the following example, the server looks for different welcome files based on the host name in the URL. For requests coming in for hostA, the server looks for welcome files named CustomerA.html. For requests coming in for hostB, the server looks for welcome files named CustomerB.html. If the request comes in for a different host, the server looks for the default host name.

  ```
  Welcome CustomerA.html  hostA.bcd.com
  Welcome CustomerB.html  hostB.bcd.com
  ```

Defaults

These defaults are in the order used by the default configuration:

```
Welcome Welcome.html
Welcome welcome.html
Welcome index.html
Welcome Frntpage.html
```

Notices

Third edition (June 2005)

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 give you any license to these patents. You can send license inquiries, in writing, to:

IBM Corporation
Attn.: G71A/503
P.O. box 12195
3039 Cornwallis Rd.
Research Triangle Park, N.C. 27709-2195
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, Japan

The following paragraph does not apply to the United Kingdom or any country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR CONDITIONS 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 document. 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
ATTN: Software Licensing
11 Stanwix Street
Pittsburgh, PA 15222-9183
U.S.A.

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

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples may 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.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

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

- AIX®
- IBM
- Netfinity®
- RS/6000®

- SecureWay®
- Tivoli
- ViaVoice®
- WebSphere

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

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

Intel™, Intel Inside (logos), MMX™ and Pentium® are trademarks of Intel Corporation in the United States, other countries, or both.

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

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

Printed in USA

GC31-6857-02

Spine information:

WebSphere Application Server Caching Proxy Administration Guide Version 6.0.2 GC31-6857-02