WebSphere® IBM HTTP Server, Version 6

User's Guide

Note
Before using this information, be sure to read the general information under "Notices" on page 125.

Compilation date: December 7, 2004

Contents

How to send your comments

Your feedback is important in helping to provide the most accurate and highest quality information.

- To send comments on articles in the WebSphere Application Server Information Center
  1. Display the article in your Web browser and scroll to the end of the article.
  2. Click on the **Feedback** link at the bottom of the article, and a separate window containing an e-mail form appears.
  3. Fill out the e-mail form as instructed, and click on **Submit feedback** .
- To send comments on PDF books, you can e-mail your comments to: **wasdoc@us.ibm.com** or fax them to 919-254-0206.

  Be sure to include the document name and number, the WebSphere Application Server version you are using, and, if applicable, the specific page, table, or figure number on which you are commenting.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Chapter 1. Welcome to the product documentation for IBM HTTP Server

This information applies to Version 6, and to all subsequent releases and modifications until otherwise indicated in new editions.

Chapter 2. Overview

This section describes what is new and changed in IBM HTTP Server V6.0.

What's new in this release

New functions in this release include:

1. IBM HTTP Server can now be remotely administered and configured using the WebSphere administrative console:
   - You can start and stop the IBM HTTP Server.
   - You can monitor the IBM HTTP Server status and display the IBM HTTP Server error and access logs.
   - You can display and edit the IBM HTTP Server configuration file.
   - You can generate the `plugin-cfg.xml` file for a particular instance of IBM HTTP Server. This file can be propagated to the machine where the instance of the IBM HTTP Server and `plugin-cfg.cml` file reside.
2. Support for IPv6 on HP-UX and Solaris.

Key differences from the Apache HTTP Server

This section takes a high-level look at the main differences between IBM HTTP Server and the Apache HTTP Server.

IBM HTTP Server is based on the Apache HTTP Server (httpd.apache.org), developed by the Apache Software Foundation. IBM HTTP Server includes the following additional features not available in the Apache HTTP Server:

- Support for the WebSphere administrative console.
- IBM support for Secure Sockets Layer (SSL) connections.
- InstallShield for multiple platforms enables consistent installation of the IBM HTTP Server on different platforms.
- Fast Response Cache Accelerator (FRCA) is available for AIX 5.x and certain Windows operating systems. It significantly improves HTTP Server performance when serving static content such as HTML files or image files.
- Dynamic content generation with FastCGI.
- Installation of IBM HTTP Server in multiple languages on all platforms.
- Web server-based Lightweight Directory Access Protocol (LDAP) authentication protection through an LDAP module.

Using third-party modules with IBM HTTP Server

There are many third-party modules written for Apache 2.0 that can be used with IBM HTTP Server.

The use of third-party modules does not prevent IBM HTTP Server from being supported, but IBM cannot support the third-party module itself. If a problem occurs when the third-party module is loaded, IBM support may ask for the problem to be reproduced without the third-party module loaded, in order to determine if problem is specific to configurations with the third-party module. If a problem is specific to configurations with the third-party module, the provider of

that module may need to help determine the cause. IBM cannot resolve such problems without the involvement of the provider of the module, as this requires understanding of the implementation of the module, particularly with regards to its use of the Apache APIs.

Upgrading from previous releases of IBM HTTP Server

This section provides information on what to look for when you are upgrading IBM HTTP Server V6.0 from a previous release.

If you are upgrading from a previous version of IBM HTTP Server, the IBM HTTP Server installer will perform the upgrade for you.

Upgrading from IBM HTTP Server V2.0. The configuration directives are compatible; however, on some platforms, the Application Programming Interface (API) for third-party modules has changed as follows:

- IPv6 support is now provided on HP-UX and Solaris platforms. Because of this change, third-party modules that use fields in the `apr_sockaddr_info_t` structure may need to be recompiled. Providers of such modules need to look at how the definition of `apr_sockaddr_info_t` varies when IPv6 is supported in order to determine if their module has to be recompiled.

Upgrading from IBM HTTP Server V1.3. The main task of upgrading from IBM HTTP Server V1.3 is obtaining versions of third-party modules which are compatible with IBM HTTP Server V6.0 or Apache HTTP Server V2.0.

Configuration changes will be required as well. These are described in the section ″Run-time configuration changes″ at http://publib.boulder.ibm.com/httpserv/manual60/upgrading.html.

IBM HTTP Server modules retain the same configuration directives from IBM HTTP Server V1.3 to IBM HTTP Server V6.0.

The following features in IBM HTTP Server V1.3 are not available in IBM HTTP Server V6.0:

- Disk caching proxy
- Digest authentication
- Simple Network Management Protocol (SNMP) agent
- Windows performance monitor is no longer supported.

IBM HTTP Server modules retain the same configuration directives from IBM HTTP Server V1.3 to IBM HTTP Server V6.0, with the exception of Fast Response Cache Accelerator (FRCA). Using the **Port** directive to define the port number FRCA will listen on is replaced by the **AfpaPort** directive.

Chapter 3. Installing and uninstalling

This section focuses on installing and uninstalling IBM HTTP Server.

Installing IBM HTTP Server

1. Prepare your operating platform for installing IBM HTTP Server as you would for installing any of the installable components on the product disc. See Preparing the operating system for product installation.
2. Insert the product disc and mount the disc if necessary. See "Mounting CD-ROMS on Linux and UNIX operating systems" on page 6 for information about mounting the product disc, if you are installing IBM HTTP Server on a Linux or UNIX platform.
3. Start the installation with the **launchpad.sh** command on Linux and UNIX platforms or the **launchpad.bat** on Windows platforms. You can also start the installation using the **/IHS/install** command, where `IHS` is the installable component directory on the product disc:
   - On Linux and UNIX platforms - `/IHS/install`
   - On Windows operating systems - `\IHS\install`

   When using the launchpad, launch the Installation wizard for IBM HTTP Server.

   After launching the Installation wizard from the launchpad or from the command line, the ISMP wizard initializes and then presents the Welcome panel.

   Separate installation procedures for the WebSphere Application Server product, the IBM HTTP Server product, and the Web server plug-ins let you install only what you need on a particular machine. Installation of the product code is simplified and faster than in Version 5. The installation program installs the system files (shared binary files), which you do not update until you install a service fix. Creation and configuration of Application Servers do not change the system files.
4. Click **Next** to display the License agreement panel.
5. Accept the license agreement and click **Next** to display the installation root directory panel.
6. Specify the root directory information and click **Next** to display the feature type selection panel.

   The panel lets you bypass features selection by accepting typical features. Selecting Custom lets you select features in the Features selection panel. If the installation directory already contains an existing IBM HTTP Server installation you will be prompted with an update install panel. If you proceed with updating the existing installation, ensure that the server is stopped. If you choose to install to a different directory, select **Back** and you will return to the installation directory panel.

   Note: The installer will not update IBM HTTP Server 1.3.x versions.
7. Click **Custom** to select features and click **Next** to display the Features selection panel. Selecting a custom installation enables you to modify the IBM HTTP Server port configurations. The default port values are 80 for IBM HTTP Server and 8008 for the IBM HTTP administration module. If the

default ports are already in use by another application on the IBM HTTP Server installation, the Port panel should be visited.

8. Select features to install and click **Next** to display the Windows service authorization panel, when installing IBM HTTP Server on a Windows platform. This panel gives the option to create a Windows service for IBM HTTP Server and the IBM HTTP administration server. You can configure the services to run as LocalSystem or a user ID that you specify. The user ID requires the following advanced user rights: Act as part of the operating system and Log on as a service.
9. Specify your user ID and password information and click **Next** when installing IBM HTTP Server on a Windows platform.
10. Review the confirmation panel to verify your selections. Click **Back** to change any of your specifications. Click **Next** to begin installing IBM HTTP Server.
11. Review the confirmation panel to verify your selections. Click **Back** to change any of your specifications. Click **Next** to begin installing IBM HTTP Server.

    After displaying installation status, the wizard displays the Completion status panel that indicates a successful installation.
12. Click Next to display the Web server plug-ins prompt panel.
13. Click Next to launch the Plug-ins installation wizard.

    If the `plugin` directory does not exist at the same level as the `IHS` directory, the prompt panel for selecting the plug-ins installer does not display and the installation is finished. In that case, launch the Plug-ins installation wizard using the launchpad.

You can get started easily with Secure Sockets Layer (SSL) connections, by making only a few configuration changes, as described in "Setting up the Secure Sockets Layer protocol" on page 19. If you run the IBM HTTP Server on a Windows platform, you can configure the Fast Response Cache Accelerator to boost performance. You can also make many other configuration changes with Apache directives.

Mounting CD-ROMS on Linux and UNIX operating systems

This section describes how to mount the CD-ROM for IBM HTTP Server on Linux and UNIX operating systems.

After inserting a CD-ROM into a drive, some Linux and UNIX operating systems require you to mount the drive.

Use these procedures to mount the product discs for IBM HTTP Server.

- Mounting the CD-ROM on AIX To mount the CD-ROM on AIX using the System Management Interface Tool (SMIT), perform the following steps:
  1. Log in as a user with root authority.
  2. Insert the CD-ROM in the drive.
  3. Create a CD-ROM mount point by entering the `mkdir -p /cdrom` command, where `cdrom` represents the CD-ROM mount point directory.
  4. Allocate a CD-ROM file system using SMIT by entering the **smit storage** command.
  5. After SMIT starts, click **File Systems > Add / Change / Show / Delete File Systems > CDROM File Systems > Add CDROM File System**.
  6. In the Add a File System window:

- Enter a device name for your CD-ROM file system in the **DEVICE Name** field. Device names for CD-ROM file systems must be unique. If there is a duplicate device name, you may need to delete a previously-defined CD-ROM file system or use another name for your directory. The example uses `/dev/cd0` as the device name.
- Enter the CD-ROM mount point directory in the **MOUNT POINT** window. In our example, the mount point directory is `/cdrom`.
- In the **Mount AUTOMATICALLY at system restart** field, select `yes` to enable automatic mounting of the file system.
- Click **OK** to close the window, then click **Cancel** three times to exit SMIT.

7. Next, mount the CD-ROM file system by entering the **smit mountfs** command.
8. In the Mount a File System window:
   - Enter the device name for this CD-ROM file system in the **FILE SYSTEM name** field. In our example, the device name is `/dev/cd0`.
   - Enter the CD-ROM mount point in the **Directory over which to mount** field. In our example, the mount point is /cdrom.
   - Enter `cdrfs` in the **Type of Filesystem** field. To view the other kinds of file systems you can mount, click List.
   - In the **Mount as READ-ONLY system** field, select `yes`.
   - Accept the remaining default values and click **OK** to close the window.

   Your CD-ROM file system is now mounted. To view the contents of the CD-ROM, place the disk in the drive and enter the **cd /cdrom** command where **cdrom** is the CD-ROM mount point directory.

- Mounting the CD-ROM on HP-UX Because WebSphere Application Server contains several files with long file names, the mount command can fail. The following steps let you mount successfully your WebSphere Application Server product CD-ROM on the HP-UX platform:
  1. Log in as a user with root authority.
  2. In the `/etc` directory, add the following line to the `pfs_fstab` file:

     ```
     /dev/dsk/c0t2d0 mount_point pfs-rrip ro,hard
     ```

     where *mount_point* represents the mount point of the CD-ROM.
  3. Start the *pfs* daemon by entering the following commands (if they are not already running):

     ```
     /usr/sbin/pfs_mountd &
     /usr/sbin/pfsd 4 &
     ```
  4. Insert the CD-ROM in the drive and enter the following commands:

     ```
     mkdir /cdrom
     /usr/sbin/pfs_mount /cdrom
     ```

     The */cdrom* variable represents the mount point of the CD-ROM.
  5. Log out.
- Mounting the CD-ROM on Linux To mount the CD-ROM on Linux:
  1. Log in as a user with root authority.
  2. Insert the CD-ROM in the drive and enter the following command:

     ```
     mount -t iso9660 -o ro /dev/cdrom /cdrom
     ```

     The */cdrom* variable represents the mount point of the CD-ROM.
  3. Log out.

Some window managers can automatically mount your CD-ROM for you. Consult your system documentation for more information.

- Mounting the CD-ROM on Solaris To mount the CD-ROM on Solaris:
  1. Log in as a user with root authority.
  2. Insert the CD-ROM into the drive.
  3. If the Volume Manager is not running on your system, enter the following commands to mount the CD-ROM:

     ```
     mkdir -p /cdrom/unnamed_cdrom
     mount -F hsfs -o ro /dev/dsk/c0t6d0s2 /cdrom/unnamed_cdrom
     ```

     The */cdrom/unnamed_cdrom* variable represents the CD-ROM mount directory and the `/dev/dsk/c0t6d0s2` represents the CD-ROM drive device.

     If you are mounting the CD-ROM drive from a remote system using NFS, the CD-ROM file system on the remote machine must be exported with root access. You must also mount that file system with root access on the local machine.

     If the Volume Manager (vold) is running on your system, the CD-ROM is automatically mounted as:

     ```
     /cdrom/unnamed_cdrom
     ```
  4. Log out.

Return to the installation procedure to continue.

Installing silently

A silent installation uses the Installation wizard to install the product in silent mode, without the graphical user interface. Instead of displaying a wizard interface, the silent installation enables the installation program to read all of your responses from a file that you provide.

1. Log on as root on a Linux or UNIX operating system, or as a member of the administrator group on a Windows operating system. Considerations for Windows operating systems follow:
   - Some steps for installing silently require the administrator group user to have the following advanced user rights:
     - *Act as part of the operating system*
     - *Log on as a service*
   - The installation wizard grants your Windows user ID the advanced user rights, if the user ID belongs to the administrator group. The silent installation does not grant these rights. If you create a new user ID on a Windows platform to perform the silent installation, you must restart the system to activate the proper authorizations for the user ID, before you can perform a successful silent installation.
   - When installing the IBM HTTP Server as a Windows service, do not use a user ID that contains spaces. A user ID with spaces cannot be validated. Such a user ID is not allowed to continue the installation. To work around this problem, install with with the service configured to run as LocalSystem, and then modify the user ID after install.
2. Copy the `responsefile.txt` file to your disk drive and rename it, for example *`myoptionsfile.txt`*. You can now customize it. Accept the IBM HTTP Server license by setting `-W silentInstallLicenseAcceptance.value="true"` in your response file.

3. Issue the proper command to use your custom response file. For example, issue one of the following commands:
   - **On Linux and UNIX platforms** - *mnt_cdrom*/IHS/install -options *myoptionsfile.txt* -silent
   - **On Windows operating systems** - *"CD-ROM drive*:\IHS\install" -options *"myoptionsfile.txt"* -silent

   You can find the sample options response file in the IBM HTTP Server directory on the product CD.

Uninstalling the IBM HTTP Server

This section contains procedures for uninstalling the IBM HTTP Server. The uninstaller program is customized for each product installation, with specific disk locations and routines for removing installed features. The uninstaller program does not remove configuration and log files.

1. Stop IBM HTTP Server.
2. Go to the directory where you installed the IBM HTTP Server. Change to the _uninst directory, located in the root directory.
3. Double-click **uninstall** to launch the uninstallation program. You can also choose to do a silent uninstall by running the uninstall -silent command. The uninstall process on Linux and UNIX systems does not automatically uninstall the GSKit. You have to uninstall the GSKit manually by using the native uninstall method.
4. Click **Next** to begin uninstalling the product.The Uninstaller wizard displays a Confirmation panel that lists the product and features that you are uninstalling.
5. Click **Next** to continue uninstalling the product.The Uninstaller wizard deletes existing profiles first. After deleting profiles, the Uninstaller wizard deletes core product files by component.
6. Click **Finish** to close the wizard after the wizard removes the product.

The IBM HTTP Server uninstallation is now complete. The uninstallation is logged in the <*install_directory*>/ihsv6_uninstall.log file.

Chapter 4. Getting started

This section describes how to start and stop IBM HTTP Server.

Starting and stopping IBM HTTP Server with the WebSphere administrative console

You can use the WebSphere administrative console to start and stop IBM HTTP Server.

1. Launch the WebSphere administrative console.
2. Click **Servers > Web servers**.
3. Select your server by clicking the check box.
4. Click **Start**.

You can stop IBM HTTP Server by clicking **Stop**.

Starting IBM HTTP Server on Linux and UNIX platforms

This section provides information on getting started with IBM HTTP Server on Linux and UNIX platforms.

The `apachectl` command is used to start and stop IBM HTTP Server. The `apachectl` command is located in the `bin` subdirectory within the IBM HTTP Server installation directory. If that directory is not in your PATH, the full path should be given on the command line.

Starting IBM HTTP Server. To start IBM HTTP Server using the default `httpd.conf` configuration file, run the `apachectl start` command .

Stopping IBM HTTP Server. To stop IBM HTTP Server using the default `httpd.conf` configuration file, run the `apachectl stop` command.

Issue the commands from the default directories, based on your operating system:

- AIX: `/usr/IBMIHS/bin/apachectl start|stop`
- HP-UX: `/opt/IBMIHS/bin/apachectl start|stop`
- Linux: `/opt/IBMIHS/bin/apachectl start|stop`
- Solaris: `/opt/IBMIHS/bin/apachectl start|stop`

To start IBM HTTP Server using an alternate configuration file, run the `apachectl -k start -f` *`path_to_configuration_file`* command . To stop IBM HTTP Server using an alternate configuration file, run the `apachectl -k stop -f` *`path_to_configuration_file`* command .

The `apachectl` command is not in your PATH, the IBM HTTP Server installation directory is `/usr/IBMIHS`, and the default configuration file is used as follows:

```
# /usr/IBMIHS/bin/apachectl start
# /usr/IBMIHS/bin/apachectl stop
```

The `apachectl` command is not in your PATH, the IBM HTTP Server installation directory is `/opt/IBMIHS`, and an alternate configuration file, `/opt/IBMIHS/conf/nodeb.conf`, is used as follows:

```
# /opt/IBMIHS/bin/apachectl -k start -f /opt/IBMIHS/conf/nodeb.conf
# /opt/IBMIHS/bin/apachectl -k stop -f /opt/IBMIHS/conf/nodeb.conf
```

To confirm that IBM HTTP Server started successfully, open a browser and type in your server name in the URL box.

If you are going to run Application Response Measurement (ARM) agents, make sure you have the authority to run ARM agents when you start IBM HTTP Server.

If the IBM HTTP Server started successfully, you can configure your server for SSL, LDAP and FRCA.

Starting IBM HTTP Server on Windows operating systems

This section provides information on getting started with IBM HTTP Server on Windows operating systems.

To start IBM HTTP Server as a Windows service:

1. Click **Start > Programs > IBM HTTP Server 6.0 > Start Server**. A message box is displayed that indicates the server has started.
2. To confirm that IBM HTTP Server started successfully, open a browser and type in your server name in the URL box.

   If you use the developer installation option, then the IBM HTTP Server does not install as a service. You have to run the `apache.exe` file from a command line.

If IBM HTTP Server does not start:

1. Go to **Services** in the Control Panel.
2. Double-click **IBM HTTP Server 6.0** to start the server.
3. To confirm that IBM HTTP Server started successfully, open a browser and type in your server name in the URL box.

If you are going to run Application Response Measurement (ARM) agents, make sure you have the authority to run ARM agents when you start IBM HTTP Server.

If IBM HTTP Server started successfully, you can configure your server for SSL, LDAP and FRCA .

Related information

Using Apache with Microsoft Windows

Chapter 5. Configuring IBM HTTP Server

This section describes topics on how to configure secure sockets layer (SSL), lightweight directory access protocol (LDAP), and fast response cache accelerator (FRCA).

Configuration files

This section describes topics on how to configure your IBM HTTP Server.

Default and sample configuration files

Locate the `httpd.conf` configuration file in the `conf` directory of your server installation.

There is also an `httpd.conf.default` file, in case you need to use another copy of the original file.

The product provides a sample configuration file called `httpd.conf.sample`, illustrating basic IBM module directives and advanced security options.

Special considerations for IBM HTTP Server

The following items should be known when using IBM HTTP Server:

- The IBM HTTP Server and administration server configuration files, `httpd.conf` and `admin.conf` respectively, support only single-byte characters (SBCS). This restriction applies to all operating system platforms.
- On the Windows platform, the forward slash character (/) should be used as a path separator in the configuration file, instead of the backward slash character (\).

  Related information

  Customizing Apache for Windows operating systems

Apache Web server features

This section describes topics on the Apache Web server.

Related information

Apache directives

Apache concepts and tasks

This section describes topics on the Apache Web server.

Related information

Set up listening sockets

Virtual hosts

Log files

Configuration files

Configuration sections

Authentication, authorization, and access control

Environment variables

URL rewriting guide

Other topics

Apache modules supported by IBM HTTP Server

This section provides information on the Apache modules that are supported by IBM HTTP Server.

The following Apache modules are supported:

Module	Description	URL
core	Core Apache HTTP Server features	
mpm_winnt	MPM for Windows	
worker	MPM for UNIX	
mod_access	Provides access control based on client host name, IP address, or other characteristics of the client request.	
mod_actions	Provides for executing CGI scripts, based on media type or request method.	
mod_alias	Provides for mapping different parts of the host filesystem in the document tree and for URL redirection.	
mod_asis	Sends files that contain their own HTTP headers.	
mod_auth	User authentication using text files.	
mod_auth_anon	Allows ″anonymous″ user access to authenticated areas.	
mod_auth_dbm	Provides for user authentication using DBM files.	
mod_autoindex	Generates directory indexes automatically, similar to the UNIX `ls` command or the Win32 `dir` shell command.	
mod_cache	Content cache keyed to URIs.	
mod_cern_meta	CERN httpd metafile semantics.	
mod_cgi (Windows platforms only)	Execution of CGI scripts.	
mod_cgid (non-Windows platforms)	Execution of CGI scripts using an external CGI daemon.	
mod_dav	Distributed Authoring and Versioning (WebDAV) functionality.	
mod_dav_fs	File system provider for mod_dav.	

mod_deflate	Compress content before it is delivered to the client.	
mod_dir	Provides for ″trailing slash″ redirects and serving directory index files.	
mod_env	Content cache storage manager keyed to URIs.	
mod_expires	Generation of Expires HTTP headers according to user-specified criteria.	
mod_ext_filter	Pass the response body through an external program before delivery to the client.	
mod_file_cache	Caches a static list of files in memory.	
mod_headers	Customization of HTTP request and response headers.	
mod_imap	Server-side image map processing.	
mod_include	Server-parsed html documents (Server Side Includes).	
mod_info	Provides a comprehensive overview of the server configuration.	
mod_isapi (Windows platforms only)	ISAPI extensions within Apache for Windows platforms.	
mod_log_config	Logging of the requests made to the server.	
mod_mem_cache	Content cache keyed to URIs.	
mod_mime	Associates the requested extensions for a filename with the behavior of the file (handlers and filters), and content (mime-type, language, character set and encoding).	
mod_mime_magic	Determines the MIME type of a file by looking at a few bytes of its contents.	
mod_negotiation	Provides for content negotiation.	
mod_proxy	HTTP/1.1 proxy/gateway server	
mod_proxy_connect	mod_proxy extension for CONNECT request handling.	
mod_proxy_ftp	FTP support module for mod_proxy.	
mod_proxy_http	HTTP support module for mod_proxy.	

mod_rewrite	Provides a rule-based rewriting engine to rewrite requested URLs on the fly.	
mod_setenvif	Enables the setting of environment variables based on characteristics of the request.	
mod_so	Loading of executable code and modules into the server at start-up or restart time.	
mod_speling	Attempts to correct mistaken URLs that users might have entered by ignoring capitalization and by allowing up to one misspelling.	
mod_status	Provides information on server activity and performance.	
mod_unique_id	Provides an environment variable with a unique identifier for each request.	
mod_userdir	User-specific directories.	
mod_usertrack	Clickstream logging of user activity on a site.	
mod_vhost_alias	Provides for dynamically configured mass virtual hosting.	

SSL and IKEYMAN

This section describes topics on how to secure IBM HTTP Server using the Secure Sockets Layer protocol, along with information for using the Key Management Utility (IKEYMAN).

Security concepts and tasks

This section describes topics on how to secure IBM HTTP Server using the Secure Sockets Layer protocol, along with information for using the Key Management Utility (IKEYMAN).

Secure Sockets Layer protocol

The Secure Sockets Layer (SSL) protocol was developed by Netscape Communications Corporation.

SSL ensures the data that is transferred between a client and a server remains private. This protocol enables the client to authenticate the identity of the server. SSL Version 3, requires authentication of the client identity.

When your server has a digital certificate, SSL-enabled browsers like Netscape Navigator and Microsoft Internet Explorer can communicate securely with your server, using SSL. With SSL, you can easily establish a security-enabled Web site on the Internet, or on your private intranet. A browser that does not support HTTP

over SSL cannot request URLs using HTTPS. The non-SSL browsers do not allow submission of forms that require secure communications.

SSL uses a *security handshake* to initiate a secure connection between the client and the server. During the handshake, the client and server agree on the security keys to use for the session and the algorithms to use for encryption. The client authenticates the server; optionally, the server can request the client certificate. After the handshake, SSL encrypts and decrypts all the information in both the HTTPS request and the server response, including:

- The URL requested by the client
- The contents of any submitted form
- Access authorization information, like user names and passwords
- All data sent between the client and the server

HTTPS represents a unique protocol that combines SSL and HTTP. Specify `https://` as an anchor in HTML documents that link to SSL-protected documents. A client user can also open a URL by specifying `https://` to request an SSL-protected document.

Because HTTPS (HTTP + SSL) and HTTP are different protocols and use different ports (443 and 80, respectively), you can run both SSL and non-SSL requests simultaneously. This capability enables you to provide information to users without security, while providing specific information only to browsers making secure requests. With this functionality, a retail company on the Internet can support users looking through their company merchandise without security, but then fill out order forms and send their credit card numbers using security.

Secure network characteristics:

The rapid growth of electronic commerce over the Internet has led to an increasing demand for secure network communications. In addition, intra-company communications over private networks often contain confidential information that needs protection.

A secure network communication has the following characteristics:

Access control. Only authorized parties protect and access resources. Restricting access on the basis of passwords, IP address, host names, or secure sockets layer (SSL) client authentication ensures access control.

Authenticity. You know who you are talking to and that you can trust that person. Authentication, using digital signature and digital certificates, ensures authenticity.

Information integrity. Messages do not get altered during transmission. Without information integrity, you have no guarantee that the message you sent matches the message that you received. Digital signatures, which are encrypted messages, ensure integrity.

Privacy and confidentiality. Information conveyed from one party to another during a transaction remains private and confidential, even if it gets into the wrong hands. Encryption ensures privacy and confidentiality.

Encryption:

Encryption in its simplest form involves scrambling a message so that no one can read the message until it is unscrambled by the receiver.

The sender uses an algorithmic pattern, or a key to scramble, or encrypt the message. The receiver has the decryption key. Encryption ensures privacy and confidentiality in transmissions sent over the Internet.

Use two different kinds of keys for encryption:

Asymmetric keys. You create a key pair with asymmetric keys. The key pair consists of a public key and a private key, which differ from each other. The private key holds more of the secret encryption pattern than the public key. Do not share your private key with anyone.

The server uses its private key to sign messages to clients. The server sends its public key to clients so that they can encrypt messages to the server, which the server decrypts with its private key. Only you can decrypt a message that is encrypted with your public key because only you have the private key. Key pairs are stored in a key database that is protected by a password.

Symmetric keys. Symmetric keys follow an older model of the sender and receiver sharing some kind of pattern. The sender uses this same pattern to encrypt the message and the receiver uses this pattern to decrypt the message. The risk involved with symmetric keys centers around finding a safe transportation method to use, when sharing your secret key with the people to which you want to communicate.

The Secure Sockets Layer (SSL) protocol uses both asymmetric and symmetric key exchange. Use asymmetric keys for the *SSL handshake*. During the handshake, the master key, encrypted with the receiver public key passes from the client to the server. The client and server make their own session keys using the master key. The session keys encrypt and decrypt data for the remainder of the session. Symmetric key exchange occurs during the exchange of the cipher specification, or encryption level.

The server needs a *digital certificate*, which is an encrypted message that authenticates Web content, to send its public key to clients. A certificate authority (CA), which signs all certificates that it issues with a private key, issues this certificate and verifies the identity of the server.

Authentication:

Authentication verifies identity.

The server uses authentication in two ways:

- **Digital signature**. A digital signature represents a unique mathematically computed signature that ensures accountability. Think of a digital signature as similar to a credit card, on which your photo displays. To verify the identity of the person that is sending you a message, look at the digital certificate of the sender.
- **Digital certificate**. A digital certificate, or digital ID, is similar to having a credit card with a picture of the bank president with his arm around you. A merchant trusts you more because not only do you look like the picture on the credit card, the bank president trusts you, too.

You base your trust of the sender authenticity on whether you trust the third party, a person, or agency that certified the sender. The third party issuing digital certificates is called a certificate authority (CA) or *certificate signer*.

A digital certificate contains:

- The public key of the person getting certified
- The name and address of the person or organization getting certified, also known as the *distinguished name*
- The digital signature of the CA
- The issue date of the certificate
- The expiration date of the certificate

You enter your distinguished name as part of a certificate request. The digitally signed certificate includes your distinguished name and the distinguished name of the CA.

You can request one of the following certificates:

- A server certificate to do commercial business on the Internet from VeriSign or some other CA. For a list of supported CAs, see Buying a certificate from an external CA provider.
- A server certificate that you create for your own private Web network.

CAs broadcast their public key and distinguished name bundled together so that people add them to their Web servers and browsers, as a trusted CA certificate. When you designate the public key and certificate from a CA to become a trusted CA certificate, your server trusts anyone who has a certificate from that CA. You can have many trusted CAs as part of your server. The HTTP Server includes several default trusted CA certificates. You can add, or remove trusted CAs, using the IBM Key Management utility that is included with your server.

To communicate securely, the receiver in a transmission must trust the CA who issued the sender certificate. This situation remains true whether the receiver is a Web server or a browser. When a sender signs a message, the receiver must have the corresponding CA-signed certificate and public key designated as a trusted CA certificate.

Quick start: Setting up Security Sockets Layer

This section describes topics on how to secure the IBM HTTP Server.

Setting up the Secure Sockets Layer protocol:

This section provides information to help you set up Secure Sockets Layer (SSL), using the default `http.conf` configuration file.

1. Specify the SSLEnable directive in the configuration file, to enable SSL.
2. Specify a Keyfile directive.
3. Restart the server.

Setting advanced security options:

After setting up secure connections, follow these instructions to enable advanced security options:

1. Enable client authentication. If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.
2. Set and view cipher specifications.
3. Define Secure Sockets Layer (SSL) for multiple-IP virtual hosts.

Overview: SSL certificates

This section describes topics on SSL certificates.

Introduction: Certificates:

This section provides information on Secure Sockets Layer certificates.

An SSL certificate uniquely identifies your Web server to Web browsers. When a Web browser establishes an SSL session with your Web server, it requests information from the certificate in order to determine if your Web server is really who it claims to be. In order to be useful, SSL certificates need to be electronically 'signed' by a trusted third-party that can vouch for the authenticity of your Web server. The ″signer″ of a certificate is known as a certificate authority (CA). The signer of your Web server certificate must be trusted by the clients (Web browsers) that connect to your Web server.

Production Web servers must use signed certificates purchased from a Certificate Authority that supports IBM HTTP Server such as VeriSign or Thawte. You can use the IKEYMAN Key Management utility provided with IBM HTTP Server to create self-signed certificates. Self-signed certificates are useful for test purposes but should not be used in a production Web server.

For your convenience, IBM HTTP Server includes several default signer certificates. Be aware that these default signer certificates have expiration dates. It is important to verify the expiration dates of all your certificates and manage them appropriately. When you purchase a signed certificate from a CA, they will provide you access to their most recent signer certificates.

Identifying a Public Key Infrastructure:

A Public Key Infrastructure (PKI) represents a system of digital certificates, certificate authorities, registration authorities, a certificate management service, and X.500 directories.

A PKI verifies the identity and the authority of each party that is involved in an Internet transaction, either financial or operational, with requirements for identity verification. Examples of these transactions include confirming the origin of proposal bids, or the author of e-mail messages.

A PKI supports the use of *certificate revocation lists* (CRLs). A CRL is a list of revoked certificates. CRLs provide a more global method for authenticating client identity by certificate, and can verify the validity of trusted CA certificates.

An X.500 directory server stores and retrieves CRLs and trusted CA certificates. The protocols used for storing and retrieving information from an X.500 directory server include Directory Access Protocol (DAP) and Lightweight Directory Access Protocol (LDAP). The IBM HTTP Server supports LDAP.

You can distribute information on multiple directory servers over the Internet and intranets, enabling an organization to manage certificates, trust policy, and CRLs from either a central location, or in a distributed manner. This capability makes the trust policy more dynamic because you can add or delete trusted CAs from a network of secure servers, without having to reconfigure each of the servers.

Related reference

Locating glossary terms

Certificate authorities:

This section describes topics on how to secure IBM HTTP Server using certificate authorities.

Associating your public key with certificate authorities:

This section contains information regarding trusted certificate authorities (CAs) on the IBM HTTP Server.

Associate your public key 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 provider. The default certificate request file name is `certreq.arm`. The certificate request file is a PKCS 10 file, in Base64-encoded format. The IBM HTTP Server supports the following external CAs:

- Thawte
- Verisign

Related concepts

"SSL certificate revocation list" on page 23
This section provides information on identifying directives for certificate revocation list (CRL) and those supported in global servers and virtual hosts.

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

List of trusted certificate authorities on the IBM HTTP Server: The default key store contains the following list of designated trusted certificate authorities (CAs) on the IBM HTTP Server:

- Verisign test signer CA cert
- VeriSign Class 2 OnSite Individual CA
- VeriSign International Server CA - Class 3
- Verisign Class 1 Public Primary Certification Authority - G2
- Verisign Class 2 Public Primary Certification Authority - G2
- Verisign Class 3 Public Primary Certification Authority - G2
- Verisign Class 1 Public Primary Certification Authority
- Verisign Class 2 Public Primary Certification Authority
- Verisign Class 3 Public Primary Certification Authority
- VeriSign Class 1 CA Individual Subscriber-Persona Not Validated
- Thawte Personal Premium CA
- Thawte Personal Freemail CA
- Thawte Personal Basic CA
- Thawte Premium Server CA
- Thawte Server CA
- RSA Secure Server Certification Authority

If you are using a personal certificate and the signer is not in the list, you must obtain a signer certificate from the associated trusted certificate authority. See "Supported certificate authority software" on page 22 for a list of certificate authorities that IBM HTTP Server supports.

Related concepts

"SSL certificate revocation list" on page 23
This section provides information on identifying directives for certificate revocation list (CRL) and those supported in global servers and virtual hosts.

Related tasks

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Creating a self-signed certificate on the Linux for S/390 platform" on page 52
It usually takes two to three weeks to get a certificate from a well-known CA. While waiting for an issued certificate, use IKEYMAN to create a self-signed server certificate to enable SSL sessions between clients and the server.

Supported certificate authority software: The IBM HTTP Server supports the following certificate authority (CA) software:

- Any X.509-compliant certificate authority
- Entrust
- Netscape Certificate Server
- Tivoli PKI
- XCert

Obtaining certificates:

This section provides information to help you get started with secure connections on the Web server. Obtaining certificates is the first step in securing your Web server.

When you set up secure connections, associate your public key with a digitally-signed certificate from a certificate authority (CA) that is designated as a trusted CA on your server.

- **Buy a certificate from an external certificate authority provider.** You can buy a signed certificate by submitting a certificate request to a CA provider. The IBM HTTP Server supports several external certificate authorities. By default, many CAs exist as trusted CAs on the IBM HTTP Server. See "List of trusted certificate authorities on the IBM HTTP Server" on page 21.

  Use the key management utility (IKEYMAN) to create a new key pair and certificate request to send to an external CA, then define SSL settings in the `http.conf` file.
- **Create a self-signed certificate.** You can use the key management utility (IKEYMAN), or you can purchase certificate authority software from a CA provider.

  Related concepts

  "Associating your public key with certificate authorities" on page 21
  This section contains information regarding trusted certificate authorities (CAs) on the IBM HTTP Server.

  "Understanding Secure Sockets Layer environment variables" on page 30
  SSL-specific environment variables get exposed to common gateway interface (CGI) applications and server-side includes (SSI) processed pages.

  Related tasks

  "Defining SSL for multiple-IP virtual hosts" on page 24
  You can define different Secure Sockets Layer (SSL) options for various virtual hosts, or multiple servers running on one machine. In the configuration file,

define each SSL directive in the stanza for the virtual host to which the directive applies. When you do not define an SSL directive on a virtual host, the server uses the directive default.

SSL certificate revocation list:

This section provides information on identifying directives for certificate revocation list (CRL) and those supported in global servers and virtual hosts.

Certificate revocation provides the ability to revoke a client certificate given to the IHS server by the browser when the key becomes compromised or when access permission to the key gets revoked. CRL represents a database which contains a list of certificates revoked before their scheduled expiration date.

If you want to enable certificate revocation in the IBM HTTP Server, publish the CRL on a Lightweight Directory Access Protocol (LDAP) server. Once the CRL is published to an LDAP server, you can access the CRL using the IBM HTTP Server configuration file. The CRL determines the access permission status of the requested client certificate.

Identifying directives needed to set up a certificate revocation list. The SSLClientAuth directive can include two options at once:

- SSLClientAuth 2 crl
- SSLClientAuth1 crl

The CRL option turns CRL on and off inside an SSL virtual host. If you specify CRL as an option, then you elect to turn CRL on. If you do not specify CRL as an option, then CRL remains off. If the first option for SSLClientAuth equals 0/none, then you cannot use the second option, CRL. If you do not have client authentication on, then CRL processing does not take place.

Identifying directives supported in global or server and virtual host. Global server and virtual host support the following directives:

- SSLCRLHostname: The IP Address and host of the LDAP server, where the CRL database resides.
- SSLCRLPort: The port of the LDAP server where the CRL database resides; the default equals 389.
- SSLCRLUserID: The user ID to send to the LDAP server where the CRL database resides; defaults to anonymous if you do not specify the bind.
- SSLStashfile: The fully qualified path to file where the password for the user name on the LDAP server resides. This directive is not required for an anonymous bind. Use when you specify a user ID. Use the **sslstash** command, located in the bin directory of IBM HTTP Server, to create your CRL password stash file. The password you specify using the **sslstash** command should equal the one you use to log in to your LDAP server.

  Usage: `sslstash [-c] <directory_to_password_file_and_file_name> <function_name> <password>`

where:

- **-c**: Creates a new stash file. If not specified, an existing file updates.
- **File**: Represents the fully qualified name of the file to create, or update.
- **Function**: Indicates the function for which to use the password. Valid values include crl, or crypto.

- **Password**: Represents the password to stash.

How to view certificate expiration dates: You can display expiration dates of certificates in your key database by viewing the certificate information with the IKEYMAN Key Management utility GUI or using the **gsk7cmd** command. The following is an example of how to use the **gsk7cmd** command to display the validity dates on all certificates in the `key.kdb` certificate key file that will expire within 1825 days (5 years):

```
gsk7cmd  -cert -list all -expiry 1825 -db key.kdb -pw <password>

Certificates in database: key.kdb
VeriSign Class 1 CA Individual Subscriber-Persona Not Validated
Validity
Not Before: Mon May 11 20:00:00 EDT 1998
Not After: Mon May 12 19:59:59 EDT 2008
```

where *<password>* is the password you specified when creating the `key.kdb` key database file.

Defining SSL for multiple-IP virtual hosts

You can define different Secure Sockets Layer (SSL) options for various virtual hosts, or multiple servers running on one machine. In the configuration file, define each SSL directive in the stanza for the virtual host to which the directive applies. When you do not define an SSL directive on a virtual host, the server uses the directive default.

The default disables SSL for each virtual host. To enable SSL:

1. Specify the SSLEnable directive on the virtual host stanza in the configuration file, to enable SSL for a virtual host.
2. Specify a Keyfile directive and any SSL directives you want enabled for that particular virtual host. You can specify any directive, except the cache directives inside a virtual host.
3. Restart the server.

Cryptographic devices for Secure Sockets Layer

Managing cryptographic keys and storing them on cryptographic hardware provides a highly secure architecture for secure online transactions. This capability greatly increases performance and security in a Web server using Secure Sockets Layer (SSL).

Get started with cryptographic hardware for Secure Sockets Layer:

The IBM 4758 requires the PKCS11 support software for the host machine and internal firmware.

You will need the manual that explains software installation and card coprocessor microcode loading. The support software and manual do not come with the IBM 4758 card, but you can download them from http://www-3.ibm.com/security/cryptocards/index.shtml. From the download site, obtain the PKCS#11 Model 002/023 software and the PKCS#11 Installation manual.

1. After installing the support software on your machine and loading the microcode on the IBM 4758, initialize the card.
2. Configure IBM HTTP Server to pass the module for the PKCS11 device, the token label, the key label of the key created by the PKCS11 device, and the user PIN password of the token to the GSKit for access to the key for the PKCS11 device by modifying the configuration file. The PKCS11 module differs for each

platform and PKCS11 device. For the IBM hardware cryptographic devices - IBM 4758 card, available on AIX and Windows operating systems, and IBM e-business Cryptographic Accelerator, the PKCS11 module ships with the `bos.pkcs11` package on AIX.

3. Install the `devices.pci.14109f00` device for the IBM 4758 and the `devices.pci.1410e601` device for the IBM e-business Cryptographic Accelerator. AIX V4.3.3 maintenance level09 is recommended when using the IBM e-business Cryptographic Accelerator. For the IBM 4758 on Windows, the PKCS11 module comes with the PKCS11 software available for download from: http://www.ibm.com/security/cryptocards/html/ordersoftware.shtml. For nCipher, the PKCS11 module ships with nCipher software and is located in the `$NFAST_HOME/toolkits/pkcs11` directory.

   The default locations of the PKCS11 modules for each PKCS11 device follow:

   - nCipher:
     - AIX - `/opt/nfast/toolkits/pkcs11/libcknfast.so`
     - HP-UX - `/opt/nfast/toolkits/pkcs11/libcknfast.sl`
     - Linux - `/opt/nfast/toolkits/pkcs11/libcknfast.so`
     - SUN - `/opt/nfast//toolkits/pkcs11/libcknfast.so`
     - Windows - `C:\nfast\toolkits\pkcs11\cknfast.dll`
   - IBM 4758:
     - AIX -`/usr/lib/pkcs11/PKCS11_API.so`
     - Windows - `$PKCS11_HOME\bin\nt\cryptoki.dll`
   - IBM e-business Cryptographic Accelerator:
     - AIX - `/usr/lib/pkcs11/PKCS11_API.so`

Cryptographic hardware for Secure Sockets Layer: The following cryptographic devices have been tested with IBM HTTP Server. However, since device drivers for these devices are frequently upgraded by the hardware vendors to correct customer-reported problems or to provide support for new operating system platforms, check with the hardware vendors for specific applications of these devices.

Device	Key Storage	Acceleration Support	Notes
Rainbow Cryptoswift PCI with BSAFE Interface Model	No	Yes	Use with SSLAcceleratorDisable directive only. Supported on HP, Solaris, and the Windows operating systems.
nCipher nFast Accelerator with BHAPI plug-in under BSAFE 4.0	No	Pure accelerator	Requires either a SCSI or PCI-based nForce unit; use with SSLAcceleratorDisable directive only. Supported on Solaris and Windows operating systems.
nCipher nForce Accelerator, *accelerator mode*	No	Yes	Uses the BHAPI and BSAFE interface. Supported on Solaris and Windows operating systems.

nCipher nForce Accelerator, Key stored accelerator mode	Yes	Yes	Uses the PKCS#11 interface. Requires either a SCSI, or PCI-based nForce unit. Move to nCipher nForce Accelerator V4.0 or later for better performance. Supported on AIX, HP, Linux, Solaris, and Windows operating systems.
IBM 4758 Model 002/023 with PKCS#11 Interface	Yes	No	Uses the PKCS11 interface. Supported on AIX and Windows operating systems.

AIX operating systems. Support for the following adapters has been tested with WebSphere Application Server V4.0.2 or later:

Device	Key Storage	Acceleration Support	Notes
Rainbow Cryptoswift PCI with BSAFE Interface Model CS/200 and CS/600	No	Yes	Supported on the AIX operating system.
IBM e-business Cryptographic Accelerator	No	Yes	Uses the PKCS11 interface. Because this device uses the PKCS11 interface, the SSLAcceleratorDisable directive does not apply to this device. Supported on the AIX operating system.

Use the Rainbow Cryptoswift, IBM e-business Cryptographic Accelerator, nCipher nFast Accelerator and nCipher nForce Accelerator, for public key operations, and RSA key decryption. These devices store keys on your hard drive. Accelerator devices speed up the public key cryptographic functions of SSL, freeing up your server processor, which increases server throughput and shortens wait time. The Rainbow Cryptoswift, IBM e-business Cryptographic Accelerator, and nCipher accelerators incorporate faster performance and more concurrent secure transactions.

The PKCS#11 protocol either stores RSA keys on cryptographic hardware, or encrypts keys using cryptographic hardware to ensure protection. The nCipher nForce Accelerator can either perform acceleration, or it can perform both acceleration and key storage with PKCS#11 support. The IBM 4758 and nCipher nForce Accelerator with PKCS#11 support ensures inaccessible keys to the outside world. This support never reveals keys in an unencrypted form because the key is either encrypted by the hardware, or stored on the hardware.

nCipher nForce Accelerator V4.0 and later using PKCS11 key storage, has a nonremovable option which can noticeably improve performance. Contact nCipher Technical Support for instructions to turn on this feature.

Initializing IBM cryptographic hardware (IBM 4758 and IBM e-business Cryptographic Accelerator) on the AIX operating system:

1. To initialize the IBM cryptographic hardware (IBM 4758 and IBM e-business Cryptographic Accelerator) on AIX, obtain and install the `bos.pkcs11` software. Obtain the most recent bos.pkcs11 package from: Download AIX fixes. For Version 4.3, click AIX 4.3 OS, Java, compilers > Download selective fixes. For Version 5.1, click AIX 5.1 OS, Java, compilers > Download selective fixes. This package installs the PKCS11 module needed for the SSLPKCSDriver directive discussed below. You also need the devices.pci.1410e601 device for the IBM e-business Cryptographic Accelerator and the devices.pci.14109f00 and devices.pci.14109f00 for the IBM 4758.
2. Initialize your token. After you install the PKCS11 software, initialize your device. You can access the Manage the PKCS11 subsystem panel from Smitty to initialize your PKCS11 device.
   a. Select Initialize your token.
   b. Set a security officer and User PIN, if not already set.
   c. Initialize your user PIN. See Chapter 5: Token Initialization from the PKCS11 manual for more detailed information.

Initializing IBM tokens on Windows operating systems:

To initialize the IBM 4758 card on Windows operating systems, you will need the PKCS11 software.

The PKCS11 software is available at http://www-3.ibm.com/security/cryptocards/html/ordersoftware.shtml.

You can use the `TOKUTIL.EXE` utility that installs with the PKCS11 software to initialize your card on Windows operating systems.

Refer to Chapter 5: Token Initialization from the PKCS11 for more details.

Make sure you have the `cryptoki.dll` module in your path.

Using IKEYMAN to store keys on a PKCS11 device:

To create keys for your PKCS11 device, provide an `ikmuser.properties` file for IKEYMAN.

To provide this file, complete the following steps:

1. Copy the `ikmuser.sample` file that ships with the IBM HTTP Server and GSKit to a file called `ikmuser.properties` in the classes directory. Typically, you can find the `ikmuser.sample` file in the following directories:
   - AIX - `/usr/opt/ibm/gskta/classes`
   - HP - `/opt/ibm/gsk7/classes`
   - Linux - `/usr/local/ibm/gsk7/classes`
   - Solaris - `/opt/ibm/gsk7/classes`
   - Windows - `C:\Program Files\ibm\gsk7\classes`

Cryptographic token may not work if the `ikmuser.properties` file is not in the classes directory.

2. Edit the `ikmuser.properties` file to set the DEFAULT_CRYPTOGRAPHIC_MODULE property to the name of the module managing your PKCS11 device. For example:
   `DEFAULT_CRYPTOGRAPHIC_MODULE=C:\pkcs11\bin\NT\cryptoki.dll`
   - nCipher:
     - AIX -`/opt/nfast/toolkits/pkcs11/libcknfast.so`
     - HP-UX -`/opt/nfast/toolkits/pkcs11/libcknfast.sl`
     - Linux - `/opt/nfast/toolkits/pkcs11/libcknfast.so`
     - SUN -`/opt/nfast/toolkits/pkcs11/libcknfast.so`
     - Windows - `C:\nfast\toolkits\pkcs11\cknfast.dll`
   - IBM 4758
     - AIX - `/usr/lib/pkcs11/PKCS11_API.so`
     - Windows - `$PKCS11_HOME\bin\NT\cryptoki.dll`
   - IBM e-business Cryptographic Accelerator
     - AIX - /usr/lib/pkcs11/PKCS11_API.so

   The module normally installs on your system when you install the software for your PKCS11 device.
3. Save the `ikmuser.properties` file. The cryptographic token is no longer a separate item on the IKEYMAN GUI menu. It is treated as one of the keystore types. You can specify the PKCS11 module name by specifying the property of DEFAULT_CRYPTOGRAPHIC_MODULE in the `ikmuser.properties` file. However, IKEYMAN will no longer try to load the DLL/LIB at startup time to decide whether to support the cryptographic token. The value of DEFAULT_CRYPTOGRAPHIC_MODULE will be used only for the default value shown on the user interface.

   When you open the Cryptographic Token, IKEYMAN will first retrieve the value of DEFAULT_CRYPTOGRAPHIC_MODULE in the `ikmuser.properties` file and pre-fill the value in the **File Name** and **Location** fields in the **Key Database File > Open** dialog box of IKEYMAN user interface. You can modify the value in the **File Name** and **Location** fields or click the **Browse** button. If the specified DLL/LIB cannot load, an error message will display.

   When the specified DLL/LIB successfully loads, you can use IKEYMAN. After opening a cryptographic token successfully, IKEYMAN will display the certificates stored in the cryptographic token.

   When IKEYMAN launches, the IBM Key Management window has an additional menu item called *cryptographic token*.
   - Click **Cryptographic Token** from your IBM Key Management window.
   - Click **Open**. The Open Cryptographic Token window displays. Select your cryptographic token label and enter the user pin and password you specified when initializing your token with the configuration utility.
   - If you want to open an existing, secondary key database, select **Open an existing secondary key database file**, and specify a file name and location. If not, disable this function by clearing the check mark. If you want to create a new, secondary key database file, select **Create new secondary key database file**, specify a conversational monitor system (CMS) key database file, the file name, and location. If not, make sure to clear the check mark by this option. Click **OK**.

- Proceed with the steps as if you had opened a key database. You can continue with the same steps to create a self-signed certificate, or add a new digital-signed certificate. Instead of using **Key Database > Open**, use **Cryptographic Token > Open**.

  With the IBM HTTP Server, you must specify a key file to perform encryption. If you use PKCS11 devices, this key file should hold your signer certificates for your personal certificate that you created using a PKCS11 device.

Configuring IBM HTTP Server to use nCipher and Rainbow accelerator devices and PKCS11 devices:

The IBM HTTP Server enables nCipher and Rainbow accelerator devices by default. To disable your accelerator device, add the **SSLAcceleratorDisable** directive to your configuration file.

When using the IBM e-business Cryptographic Accelerator, or the IBM 4758, the user ID under which the Web server runs must be a member of the PKCS11 group. You can create the PKCS11 group by installing the `bos.pkcs11` package or its updates. Change the **Group** directive in the configuration file to `group pkcs11`.

If you want the IBM HTTP Server to use the PKCS11 interface, configure the following:

1. Stash your password to the PKCS11 device, or optionally enable password prompting: Syntax: `sslstash [-c] <file> <function> <password>` where:
   - -c: Creates a new stash file. If not specified, an existing stash file is updated.
   - file: Represents a fully-qualified name of the file to create or update.
   - function: Represents the function for which the server uses the password. Valid values include `crl` or `crypto`.
   - password: Indicates the password to stash.
2. Place the following directives in your configuration file:
   - `SSLPKCSDriver <fully qualified name of the PKCS11 driver used to access PKCS11 device>`

     See SSLPKCSDriver directive for the default locations of the PKCS11 module, for each PKCS11 device.
   - `SSLServerCert <token label: key label of certificate on PKCS11 device>`
   - `SSLStashfile <fully qualified path to the file containing the password for the PKCS11 device>`
   - `Keyfile <fully qualified path to key file with signer certificates>`

Enabling session ID caching

Cached session IDs enable a client and server to communicate with a shortened handshake.

Enable session ID caching on Windows platforms by completing the following steps:

1. Set the timeout value that applies to the session ID cache to a value greater than 0. Specify the SSLV2Timeout directive with valid values between 0 and 100, and the SSLV3Timeout directive with valid values between 0 and 86400. These values appear in seconds.
2. Save the configuration file and restart the server.

Enable session ID caching on UNIX platforms, by completing the following steps:

1. Accept the default, or specify the SSLCacheEnable directive in the configuration file outside of a virtual host stanza.
2. Assign a name to the port for the session ID cache, by specifying the SSLCachePortFilename, if the default name in the <*server root*>/logs directory is unacceptable.
3. Set the timeout value that applies to the session ID cache. Specify the SSLV2Timeout directive with valid values between 0 and 100, and the SSLV3Timeout directive with valid values between 0 and 86400. These values appear in seconds.
4. Decide whether to log caching errors. To enable logging of errors that can occur during session ID caching, or retrieval from the cache, specify the SSLCacheErrorLog directive in the configuration file outside of a virtual host stanza.
5. Save the configuration file and restart the server.

Understanding Secure Sockets Layer environment variables

SSL-specific environment variables get exposed to common gateway interface (CGI) applications and server-side includes (SSI) processed pages.

You can categorize these variables into three types:

- Variables for information regarding the SSL handshake
- Variables for exposing the server certificate information
- Variables for exposing client certificate information, if you enable client authentication.

When making a valid SSL request, the SSL handshake environment variables and the server certificate environment variables are set. Setting client authentication to either optional or require, results in the client certificate environment variables setting.

Secure Sockets Layer handshake environment variables:

When making a valid SSL request, the SSL handshake environment variables and the server certificate environment variables are set. Setting client authentication to either optional or require, results in the client certificate environment variables setting.

A list of SSL handshake environment variables, with their descriptions and values follows:

SSL handshake environment variable	Description	Value
HTTPS	Indicates SSL connection	String contains either ON, for an SSL connection, or OFF, if not.
HTTPS_CIPHER	Contains the cipher used in the SSL handshake.	See the table below.
HTTPS_KEYSIZE	Indicates the size of the key.	See the table below.
HTTPS_SECRETKEYSIZE	Indicates the strength of the key.	See the table below.
SSL_PROTOCOL_VERSION	Contains the protocol version.	String contains either SSLV2, SSLV3, or TLSV1.

The following list contains the values for HTTPS_KEYSIZE and HTTPS_SECRETKEYSIZE:

Secure Sockets Layer V3 and Transport Layer Security V1

Cipher suite	Key size	Secret key size
SSL_RSA_WITH_NULL_MD5	0	0
SSL_RSA_WITH_NULL_SHA	0	0
SSL_RSA_EXPORT_WITH_RC4_40_MD5	128	40
SSL_RSA_WITH_RC4_128_MD5	128	128
SSL_RSA_WITH_RC4_128_SHA	128	128
SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5	128	40
SSL_RSA_WITH_DES_CBC_SHA	64	56
SSL_RSA_WITH_3DES_EDE_CBC_SHA	192	168
SSL_NULL_WITH_NULL_NULL	0	0
TLS_RSA_EXPORT1024_WITH_RC4_56_SHA	56	20
TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA	56	20

Secure Sockets Layer V2

Cipher suite	Key size	Secret key size
RC4_128_WITH_MD5	128	128
RC4_128_EXPORT40_WITH_MD5	128	40
RC2_128_CBC_WITH_MD5	128	128
RC2_128_CBC_EXPORT40_WITH_MD5	128	40
DES_64_CBC_WITH_MD5	64	56
DES_192_EDE3_CBC_WITH_MD5	192	168

Related tasks

"Server certificate environment variables"

"Client certificate environment variables" on page 32

Server certificate environment variables: A list of server certificate environment variables with their associated descriptions and values follows:

Server certificate environment variable	Description	Value
SSL_SERVER_C	Contains the country attribute of the server certificate	String or empty
SSL_SERVER_CN	Contains the common name attribute of the server certificate	String or empty
SSL_SERVER_DN	Contains the distinguished name of the server certificate used in the IP-based virtual host which received the request	String or empty

SSL_SERVER_EMAIL	Contains the e-mail attribute of the server certificate	String or empty
SSL_SERVER_L	Contains the locality attribute of the server certificate	String or empty
SSL_SERVER_O	Contains the organization attribute of the server certifiate	String or empty
SSL_SERVER_OU	Contains the organizational unit attribute of the server certificate	String or empty
SSL_SERVER_ST	Contains the state or province attribute of the server certificate	String or empty

Related tasks

"Secure Sockets Layer handshake environment variables" on page 30
When making a valid SSL request, the SSL handshake environment variables and the server certificate environment variables are set. Setting client authentication to either optional or require, results in the client certificate environment variables setting.

"Client certificate environment variables"

Client certificate environment variables:

A list of the client certificate environment variables, with their associated descriptions and values follows:

SSL client certificate environment variable	Description	Value
SSL_CLIENT_C	Contains the client certificate country	String or empty
SSL_CLIENT_CERTBODY	Contains the client certificate	String containing the complete client certificate as a string
SSL_CLIENT_CERTBODYLEN	Contains the length of the client certificate	Integer
SSL_CLIENT_CN	Contains the client certificate common name	String or empty
SSL_CLIENT_DN	Contains the distinguished name from the client certificate	String or empty
SSL_CLIENT_EMAIL	Contains the client certificate e-mail	String or empty
SSL_CLIENT_IC	Contains the country name of the client certificate issuer	String or empty
SSL_CLIENT_ICN	Contains the common name of the client certificate issuer	String or empty
SSL_CLIENT_IDN	Contains the distinguished name of the client certificate issuer	String or empty

SSL_CLIENT_EMAIL	Contains the e-mail address of the client certificate issuer	String or empty
SSL_CLIENT_IL	Contains the locality of the client certificate issuer	String or empty
SSL_CLIENT_IO	Contains the organization name of the client certificate issuer	String or empty
SSL_CLIENT_IOU	Contains the organizational unit name of the client certificate issuer	String or empty
SSL_CLIENT_IPC	Contains the postal code of the client certificate issuer	String or empty
SSL_CLIENT_IST	Contains the state or province of the client certificate issuer	String or empty
STRING_CLIENT_L	Contains the client certificate locality	String or empty
SSL_CLIENT_NEWSESSIONID	Indicates whether this session ID is new	String containing ″TRUE″ or ″FALSE″
SSL_CLIENT_O	Contains the client certificate organization	String or emtpy
SSL_CLIENT_OU	Contains the client certificate organizational unit	String or empty
SSL_CLIENT_PC	Contains the client certificate postal code	String or empty
SSL_CLIENT_SERIALNUM	Contains the client certificate serial number	String or empty
SSL_CLIENT_SESSIONID	Contains the session ID	String or empty
SSL_CLIENT_ST	Contains the client certificate state or province	String or emtpy

Related tasks

“Secure Sockets Layer handshake environment variables” on page 30
When making a valid SSL request, the SSL handshake environment variables and the server certificate environment variables are set. Setting client authentication to either optional or require, results in the client certificate environment variables setting.

“Server certificate environment variables” on page 31

SSL client authentication

This section describes topics on how to configure secure sockets layer.

Choosing the level of client authentication:

If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

By default, the IBM HTTP Server enables the cache accelerator.

For each virtual host, choose the level of client authentication:

1. Specify one of the following values in the configuration file on the SSLClientAuth directive, for each virtual host stanza . A virtual host stanza

represents a section of the configuration file that applies to one virtual host.

None	The server requests no client certificate from the client.
Optional	The server requests, but does not require, a client certificate. If presented, the client certificate must prove valid.
Required	The server requires a valid certificate from all clients.

For example, `SSLClientAuth required`.

If you want to use a certificate revocation list (CRL), add `crl`, as a second argument for SSLClientAuth. For example: `SSLClientAuth required crl`.

2. Save the configuration file and restart the server.

Choosing the type of client authentication protection:

If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

By default, the IBM HTTP Server enables the cache accelerator.

For each virtual host, choose the type of client authentication:

1. Specify one of the following directives in the configuration file, for each virtual host stanza:
   a. SSLClientAuthRequire. Refer to the description of SSLClientAuthRequire (need link). For example, `SSLClientAuthRequire CommonName=Richard`
   b. SSLFakeBasicAuth. Refer to the description of SSLFakeBasicAuth (need link). If you specify SSLFakeBasicAuth, verify that the mod_ibm_ssl module is displayedlast in the module list.
2. Save the configuration file and restart the server.

Related tasks

"Setting cipher specifications"
This section describes setting cipher specifications for secure transactions.

Overview: SSL cipher specifications

This section describes topics on how to configure secure sockets layer cipher specifications.

Setting cipher specifications:

This section describes setting cipher specifications for secure transactions.

For each virtual host, set the cipher specification to use during secure transactions. A virtual host allows you to maintain more then one server on a single machine. This is done by having different host names. The specified cipher specifications validate against the level of the Global Security Kit (GSK) toolkit that is installed on your system. Invalid cipher specifications cause an error to log in the error log. If the client issuing the request does not support the ciphers specified, the request fails and the connection closes to the client.

1. Specify a value for each virtual host stanza in the configuration file, that are on the "SSLCipherSpec directive" on page 59, as in the following examples: `SSLCipherSpec`*short_name* or `SSLCipherSpec` *long_name*, where *short_name* and

long_name represent the name of "Version 2 cipher specifications" or "SSL Version 3 and TLS Version 1 cipher specifications."

2. Save the configuration file and restart the server.

Cipher specifications are now being used when connecting from a client to a server.

Viewing cipher specifications:

This section describes viewing cipher specifications for secure transactions.

To see which cipher specifications the server uses for secure transactions, look at the informational messages in the error log.

1. Specify to include informational messages in the error log by using the LogLevel directive in the configuration file:`LogLevel info`. The error log is specified by the ErrorLog directive in the http configuration file. The location is set by the ErrorLog directive, which can be configured.
2. Look in the error log for messages in this format: *`TimeStamp info_message`* `mod_ibm_ssl: Using Version` *`2/3`* `Cipher:` *longname*|*shortname*

The order that the cipher specifications are displayed in the error log from top to bottom represents the attempted order of the cipher specifications.

Version 2 cipher specifications:

Short name	Long name	Description
27	SSL_DES_192_EDE3_CBC_WITH_MD5	Triple-DES (168-bit)
21	SSL_RC4_128_WITH_MD5	RC4 (128-bit)
23	SSL_RC2_CBC_128_CBC_WITH_MD5	RC2 (128-bit)
26	SSL_DES_64_CBC_WITH_MD5	DES (56-bit)
22	SSL_RC4_128_EXPORT40_WITH_MD5	RC4 (40-bit)
24	SSL_RC2_CBC_128_CBC_EXPORT40_WITH_MD5	RC2 (40-bit)

SSL Version 3 and TLS Version 1 cipher specifications:

Short name	Long name	Description
3A	SSL_RSA_WITH_3DES_EDE_CBC_SHA	Triple-DES SHA (168-bit)
33	SSL_RSA_EXPORT_WITH_RC4_40_MD5	RC4 SHA (40-bit)
34	SSL_RSA_WITH_RC4_128_MD5	RC4 MD5 (128-bit)
39	SSL_RSA_WITH_DES_CBC_SHA	DES SHA (56-bit)
35	SSL_RSA_WITH_RC4_128_SHA	RC4 SHA (128-bit)
36	SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5 Cipher specification 36 requires Netscape Navigator V4.07; it does not work on earlier versions of Netscape browsers.	RC2 MD5 (40-bit)
32	SSL_RSA_WITH_NULL_SHA	
31	SSL_RSA_WITH_NULL_MD5	
30	SSL_NULL_WITH_NULL_NULL	

62	TLS_RSA_EXPORT1024_WITH_RC4_56_SHA	RC4 SHA Export 1024 (56-bit)
64	TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA	DES SHA Export 1024 (56-bit)

IKEYMAN

This section describes topics on how to set up and use the Key Management utility (IKEYMAN) with IBM HTTP Server.

The Key Management utility:

To have a secure network connection, create a key for secure network communications and receive a certificate from a certificate authority (CA) that is designated as a trusted CA on your server.

Use IKEYMAN to create key databases, public and private key pairs and certificate requests. If you act as your own CA, you can use IKEYMAN to create self-signed certificates. If you act as your own CA for a private Web network, you have the option to use the server CA utility to generate and issue signed certificates to clients and servers in your private network.

Use IKEYMAN for configuration tasks that are related to public and private key creation and management. You cannot use IKEYMAN for configuration options that update the `httpd.conf` configuration file.

Linux for S/390 users. Use the IKEYCMD command line interface to perform similar functions to IKEYMAN.

Setting your system environment:

This section provides detailed information on tasks that you can perform using the IBM Key Management utility (IKEYMAN). This information does not explain how to configure security options that require updates to the server configuration file.

The IKEYMAN user interface is Java-based and needs either an IBM Developer Kit for the Java platform, or a Java Runtime Environment (JRE) to run. Verify that you have Developer Kit level V1.4 or later for IKEYMAN support.

On Windows and Solaris operating systems, the GSKit libraries that are installed as part of the Secure Sockets Layer (SSL) component include a JRE. No additional environment setup requirements exist on these platforms. To run on AIX, HP, or Linux operating systems, or to use another IBM Developer Kit for the Java platform on the Solaris operating system, set your system environment using the following guidelines:

- Set the variables for the IBM Developer Kit for the Java platform. These variables vary, depending on the version. You can verify these variables by reading the documentation included with the Developer Kit.
- For the IBM Developer Kit for the Java platform, Version 1.4.x, update the PATH variable:

  ```
  EXPORT PATH = <the IBM Developer Kit for the Java platform
  home directory full path name> /jre/sh: <the IBM Developer
  Kit for Java platform home directory full path name>/sh: $PATH
  ```

- On Linux and UNIX platforms: Run <*install_root*>/bin/ikeyman to open the IKEYMAN user interface. If you want to run IKEYMAN from any directory, add the path where IKEYMAN installs to your PATH environment variable: EXPORT PATH=/bin:$PATH

Linux users. See Using the IKEYCMD command-line interface for more detailed information regarding IKEYCMD. To run IKEYMAN on the Linux for S/390 operating system, set up environment variables to use the IKEYCMD command line interface as follows:

- Set your PATH to where your Java or JRE executable resides: EXPORT PATH=/opt/IBMJava/bin:$PATH
- Set the following CLASSPATH environment variable:

  ```
  EXPORT CLASSPATH=/usr/local/ibm/gsk/classes/cfwk.zip:
  /usr/local/IBM/gsk/classes/gsk7cls.jar:$CLASSPATH
  ```

When completed, IKEYCMD runs from any directory. To run an IKEYCMD command, use the following syntax:

```
java com.ibm.gsk.ikeyman <command>
```

You can substitute jre for java, depending on whether you use a JRE executable, or the IBM Developer Kit for the Java platform. For example, jre com.ibm.gsk.ikeyman.ikeycmd <*command*>.

Each IKEYCMD, except create database, requires that you specify the key database and password for the key database because the database opens with each command. See Using the IKEYCMD command-line interface, for more detailed information on IKEYCMD.

If you are unable to open IKEYMAN, take the following actions:

1. Rename and move the $JAVA_HOME/jre/lib/ext/gskikm.jar file to a directory that is not visible to the JDK class path, extdirs, or bootclasspath. For example, on Linux platforms: mv $JAVA_HOME/jre/lib/ext/gskikm.jar to /gskfiles/gskikm.jar.org
2. Set the JAVA_HOME user variable to the location of the Java Developer Kit on the machine. If you are using WebSphere Application Server, set the JAVA_HOME user variableto the JRE that shipped with WebSphere Application Server: EXPORT JAVA_HOME=*the IBM Developer Kit for the Java platform home directory full path name*

   For example, on Linux operating systems: export JAVA_HOME=/opt/WebSphere/AppServer/java

   Verify that the C:Program Files\IBM\Java141\jre\lib\security\java.security file has the following providers for GSKit:

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

   If you plan to use cryptographic hardware for GSKit, add the following providers in this order:

   ```
   security.provider.1=com.ibm.spi.IBMCMSProvider
   security.provider.2=com.ibm.crypto.provider.IBMJCE
   security.provider.3=com.ibm.jsse.IBMJSSEProvider
   security.provider.4=com.ibm.crypto.pkcs11.provider.IBMPKCS11
   ```
3. If you are not using an IBM Developer Kit, or if the IBM Developer Kit files are older than the GSKit files, copy all of the Java archive (JAR) files from the ibm\gsk7\classes\jre\lib\ext directory to the Java141\jre\lib\ext directory.

Key Management Utility GUI interface:

This section describes topics on how to set up and use the Key Management utility (IKEYMAN) with IBM HTTP Server.

Starting the Key Management utility user interface:

This section describes how to get started and use Key Management (IKEYMAN) utility.

To start the IKEYMAN user interface, type `<install_root>/bin/ikeyman` on the command line, or change to the `<install_root>/bin` directory and type `ikeyman` on the command line.

Windows operating systems. Go to the start user interface and click **Start Key Management Utility**. If you start IKEYMAN to create a new key database file, the utility stores the file in the directory where you start IKEYMAN.

Creating a new key database:

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.

You can create multiple databases if you prefer to keep certificates in separate databases.

To create a new key database:

1. Start the IKEYMAN user interface. Refer to Starting the Key Management utility topic for platform-specific instructions.
2. **On Linux and UNIX platforms,** click **key database file** from the main user interface, then click **New**.

   On Windows operating systems, enter your key database name in the New dialog box, or click **key.kdb** if you use the default. Click **OK**.
3. Enter your correct password in the Password Prompt dialog box, and confirm the password. Click **OK**.

The new key database should appear in the IKEYMAN utility with default signer certificates. Ensure that there is a functional, non-expiring signer certificate for each of your personal certificates.

Setting the database password:

When you create a new key database, you specify a key database password. This password protects the private key. The private key is the only key that can sign documents or decrypt messages that are encrypted with the public key. Changing the key database password frequently is a good practice.

Use the following guidelines when specifying the password:

- The password must come from the U.S. English character set.
- The password must contain at least six characters and contain at least two nonconsecutive numbers. Make sure that the password does not consist of publicly obtainable information about you, such as the initials and birth date for you, your spouse, or children.

- Stash the password or enable secure sockets layer (SSL) password prompting. See Using SSL password prompting.

Keep track of expiration dates for the password. If the password expires, a message writes to the error log. The server starts, but a secure network connection does not exist if the password has expired.

Changing the database password:

This section describes how to change the password for a key database. A key database is used to store public keys that are used for secure connections.

To change the password:

1. Start the IKEYMAN user interface. Refer to Starting the Key Management utility for platform-specific instructions.
2. Click **Key Database File** from the main UI, then click **Open**.
3. Enter your key database name in the Open dialog box, or click the **key.kdb** file, if you use the default. Click **OK**.
4. Enter your password in the Password Prompt dialog box, and click **OK**.
5. Click **Key Database File** from the main UI, then click **Change Password**.
6. Enter a new password in the Password Prompt dialog box, and a new confirming password. Click **OK**.

Creating a new key pair and certificate request:

You find key pairs and certificate requests stored in a key database. This section provides information on how to create a key.

To create a public and private key pair and certificate request, complete the following steps:

1. If you have not created the key database, see Creating a new key database for instructions.
2. Start the IKEYMAN user interface. Refer to Starting the Key Management Utility for platform-specific instructions.
3. Click **Key Database File** from the main UI, and then click **Open**.
4. Enter your key database name in the Open dialog box, or click the **key.kdb** file, if you use the default. Click **OK**.
5. In the Password Prompt dialog box, enter your correct password and click **OK**.
6. Click **Create** from the main UI, then click **New Certificate Request**.
7. In the New Key and Certificate Request dialog box, enter the following information:
   - Key label: Enter a descriptive comment to identify the key and certificate in the database.
   - Key size: Choose your level of encryptions from the drop-down menu.
   - Organization Name: Enter your organization name.
   - Organization Unit
   - Locality
   - State/Province
   - Zip code
   - Country: Enter a country code. Specify at least two characters. Example: US Certificate request file name, or use the default name.

8. Click **OK**.
9. Click **OK** in the Information dialog box. A reminder to send the file to a certificate authority appears.

Exporting keys:

This article describes how to export your key into another database or to a PKCS12 file. PKCS12 is a standard for securely storing private keys and certificates.

To export keys from another database, complete the following steps:

1. Start the IKEYMAN user interface. Refer to Starting the Key Management utility for platform-specific instructions.
2. Click **Key Database File** from the main UI, then click **Open**.
3. Enter your key database name in the Password Prompt dialog box, or click **key.kdb** if you are using the default.
4. Enter your correct password in the Password Prompt dialog box, and click **OK**.
5. Click **Personal Certificates** in the Key database content frame, then click **Export/Import** on the label.
6. In the Export/Import Key window:
   a. Click **Export Key**.
   b. Click the target database type.
   c. Enter the file name, or use the Browse option.
   d. Enter the current location.
7. Click **OK**.
8. Click **OK** in the Password prompt dialog box, to export the selected key to another key database.

To export keys to a PKCS12 file, complete the following:

1. Enter `ikeyman` on a command line on the Linux or UNIX platforms, or start the Key Management utility in the IBM HTTP Server folder on the Windows operating system.
2. Click **Key Database File** from the main UI, then click **Open**.
3. Enter your key database name in the Open dialog box, or click **key.kdb** if you use the default. Click **OK**.
4. Enter your password in the Password Prompt dialog box, and click **OK**.
5. Click **Personal Certificates** in the Key Database content frame, then click **Export/Import** on the label.
6. In the Export/Import Key window:
   a. Click **ExportKeyM**.
   b. Click the PKCS12 database file type.
   c. Enter the file name, or use the Browse option.
   d. Enter the correct location.
7. Click **OK**.
8. Enter the correct password in the Password prompt dialog box, and enter the password again to confirm. Click **OK** to export the selected key to a PKCS12 file.

Importing keys:

This article describes how to import your key into another database or to a PKCS12 file. PKCS 12 is a standard for securely storing private keys and certificates.

To import keys from another database, complete the following steps:

1. Start the IKEYMAN user interface. Refer to Starting the Key Management utility for platform-specific instructions.
2. Click **Key Database File** from the main UI, then click **Open**.
3. Enter your key database name in the Password prompt dialog box, or click **key.kdb** if you are using the default.
4. Enter your correct password in the Password prompt dialog box, and click **OK**.
5. Click **Personal Certificates** in the Key Database content frame, then click **Export/Import** on the label.
6. In the Export/Import Key window:
   a. Click **Import Key**.
   b. Click the target database type.
   c. Enter the file name, or use the Browse option.
   d. Enter the current location.
7. Click **OK**.
8. Click **OK** in the Password prompt dialog box, to import the selected key to another key database.

To import keys to a PKCS12 file, complete the following steps:

1. Enter `ikeyman` on a command line on the Linux or UNIX platforms, or start the Key Management utility in the IBM HTTP Server folder on the Windows operating system.
2. Click **Key Database File** from the main UI, then click **Open**.
3. Enter your key database name in the Open dialog box, or click **key.kdb**, if you use the default. Click **OK**.
4. Enter your password in the Password prompt dialog box, and click **OK**.
5. Click **Personal Certificates** in the Key Database content frame, then click **Export/Import** on the label.
6. In the Export/Import Key window:
   a. Click **Import Key**.
   b. Click the PKCS12 database file type.
   c. Enter the file name, or use the Browse option.
   d. Enter the correct location.
7. Click **OK**.
8. Enter the correct password in the Password prompt dialog box, then click **OK**.

Listing certificate authorities:

This article describes how to view trusted certificate authorities within a key database.

A trusted certificate authority issues and manages public keys for data encryption. A key database is used to share public keys that are used for secure connections.

To display a list of trusted certificate authorities (CAs) in a key database, complete the following steps:

1. Start the IKEYMAN user interface. Refer to Starting the Key Management utility for platform-specific instructions.
2. Click **Key Database File** from the main UI, then click **Open**.
3. Enter your key database name in the Open dialog box, or click **key.kdb** if you are using the default.
4. Enter your correct password in the Password prompt dialog box, and click **OK**.
5. Click **Signer Certificates** in the Key database content frame.
6. Click **Signer Certificates, Personal Certificates,**or **Certificate Requests**, to view the list of CAs in the Key Information window.

Opening a key database:

This article describes how to open a key database. A key database stores the keys and certificates.

To open an existing key database:

1. Start the IKEYMAN user interface. Refer to Starting the Key Management Utility for platform-specific instructions.
2. Click **Key Database File** from the main UI, then click **Open**.
3. In the Open dialog box, enter your key database name, or click the **key.kdb** file, if you use the default. Click **OK**.
4. Enter your correct password in the Password Prompt dialog box, and click **OK**.
5. The key database name is displayed in the File Name text box.

You must use IKEYMAN to manage your key database when using IBM HTTP Server. When you use IKEYMAN to open a key database, you must first specify the name of the key database and the password.

Creating a self-signed certificate:

It usually takes two to three weeks to get a certificate from a well known certificate authority (CA). While waiting for a certificate to be issued, use IKEYMAN to create a self-signed server certificate to enable SSL sessions between clients and the server. Use this procedure if you act as your own CA for a private Web network.

Complete the following steps to create a self-signed certificate:

1. If you have not created the key database, see Creating a new key database for instructions.
2. Start the IKEYMAN user interface.
3. Click **Key Database File** from the main UI, and then click **Open**.
4. Enter your key database name in the Open dialog box, or click the **key.kdb** file, if you use the default. Click **OK**.
5. In the Password Prompt dialog box, enter your correct password and click **OK**.
6. Click **Personal Certificates** in the Key Database content frame, and click the **New Self-Signed** radio button.
7. Enter the following information in the Password Prompt dialog box:
   - Key label: Enter a descriptive comment to identify the key and certificate in the database.
   - Key size: Choose your level of encryptions from the drop-down menu.
   - Common Name: Enter the fully qualified host name of the Web server as the common name. Example: www.myserver.com.

- Organization Name: Enter your organization name.
- Organization Unit
- Locality
- State/Province
- Zip code
- Country: Enter a country code. Specify at least two characters. Example: US
  Certificate request file name, or use the default name.
- Validity Period

8. Click **OK**.

Receiving a signed certificate from a certificate authority:

This topic describes how to receive an electronically mailed certificate from a certificate authority (CA), that is designated as a trusted CA on your server. A certificate authority is a trusted third-party organization or company that issues digital certificates used to create digital signatures and public-private key pairs.

By default, the following CA certificates are stored in the key database and marked as trusted CA certificates:

- RSA Secure Server Certification Authority (from VeriSign)
- Thawte Personal Basic CA
- Thawte Personal Freemail CA
- Thawte Personal Premium CA
- Thawte Personal Server CA
- Thawte Server CA
- Verisign Class 1 CA Individual-Persona Not Validated
- Verisign Class 2 CA Individual-Persona Not Validated
- Verisign Class 3 CA Individual-Persona Not Validated
- Verisign Class 1 CA Public Primary Certification Authority
- Verisign Test CA Root Certificate

The certificate authority can send more than one certificate. In addition to the certificate for your server, the CA can also send additional signing certificates or intermediate CA certificates. For example, Verisign includes an intermediate CA certificate when sending a Global Server ID certificate. Before receiving the server certificate, receive any additional intermediate CA certificates. Follow the instructions in the Storing a CA certificate topic to receive intermediate CA certificates.

If the CA that issuing your CA-signed certificate is not a trusted CA in the key database, store the CA certificate first 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. For instructions, see Storing a CA certificate.

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

1. Start the IKEYMAN user interface. Refer to Starting the Key Management utility for platform-specific instructions.
2. Click **Key Database File** from the main UI, then click **Open**.
3. Enter your key database name in the Open dialog box, or click the **key.kdb** file, if you use the default. Click **OK**.

4. Enter your correct password in the Password Prompt dialog box, then click **OK**.
5. Click **Personal Certificates** in the Key database content frame, then click **Receive**.
6. Enter the name of a valid Base64-encoded file in the Certificate file name text field in the Receive certificate from a file dialog box. Click **OK**.

Displaying the default key in a key database:

To display the default key entry:

1. Start the IKEYMAN GUI.
2. Click **Key Database File** from the main UI, then click **Open**.
3. Enter your key database name in the Open dialog box, or click the **key.kdb**file, if using the default. Click **OK**.
4. Enter your password in the Password Prompt dialog box, then click **OK**.
5. Click **Personal Certificates** in the Key Database content frame, and click the **CA certificate** label name.
6. Click **View/Edit** and view the certificate default key information in the Key Information window.

Storing a certificate authority certificate:

To store a certificate from a certificate authority (CA) who is not a trusted CA:

1. Start the IKEYMAN GUI.
2. Click **Key Database File** from the main UI, then click **Open**.
3. Enter your key database name in the Open dialog box, or click the **key.kdb**file, if using the default. Click **OK**.
4. Enter your password in the Password Prompt dialog box, then click **OK**.
5. Click **Signer Certificates** in the Key Database content frame, then click **Add**.
6. In the Add CA Certificate from a File dialog box, click the **Base64-encoded ASCII data certificate file name**, or use the Browse option. Click **OK**.
7. In the Label dialog box, enter a label name and click **OK**.

Storing the encrypted database password in stash file:

For a secure network connection, you can store the encrypted database password in a stash file.

To store the password while a database creates:

1. Start the IKEYMAN GUI.
2. Click **Key Database File** from the main UI, then click **Open**.
3. Enter your key database name in the Open dialog box, or click the **key.kdb**file, if using the default. Click **OK**.
4. Enter your password in the Password Prompt dialog box, then enter again to confirm your password.
5. Select the stash box and click **OK**.
6. Click **Key Database File > Stash Password**.
7. Click **OK** in the information dialog box.

To store the password after creating a database:

1. Start the IKEYMAN GUI.
2. Click **Key Database File** from the main UI, then click **Open**.
3. Enter your key database name in the Open dialog box, or click the `key.kdb` file, if you use the default. Click **OK**.
4. Enter your correct password in the Password Prompt dialog box, and click **OK**.
5. Click **Key Database File**, then click **Stash Password**.
6. Click **OK** in the Information dialog box.

Key Management Utility command line interface on the Linux for S/390 platform:

On the Linux for S/390 platform, IKEYCMD, the Java command line interface to IKEYMAN, provides the necessary options to create and manage keys, certificates and certificate requests.

If you act as your own certificate authority (CA), you can use IKEYCMD to create self-signed certificates. If you act as your own CA for a private Web network, you have the option to use the server CA utility to generate and issue signed certificates to clients and servers in your private network.

Use IKEYCMD for configuration tasks related to public and private key creation and management. You cannot use IKEYCMD for configuration options that update the server configuration file, `httpd.conf`. For options that update the server configuration file, use the IBM HTTP Server administration server.

The IKEYCMD user interface uses Java and native command line invocation, enabling IKEYMAN task scripting.

Using the gsk7cmd command on the Linux for S/390 platform:

You can run IKEYMAN using the **gsk7cmd** command to invoke the Java virtual machine with the correct parameters. Alternatively, you can use the Java command to run the class directly.

- To run IKEYCMD using the **gsk7cmd** command, set up environmental variables. Set your PATH to the location of your Java or JRE executable as follows:

  ```
  export PATH=/opt/IBMJava/bin:/usr/local/ibm/gsk7/bin:$PATH
  ```

  The **gsk7cmd** command should run from any directory using the following syntax:

  ```
  gsk7cmd command
  ```

  where *command* is the required command name.
- To run IKEYMAN by specifying the class directly, set up environment variables to use the IKEYCMD command-line interface as follows:
  1. Set your PATH to the location of your Java or JRE executable:

     ```
     export PATH=/opt/IBMJava/bin:$PATH
     ```
  2. Set the following CLASSPATH environment variable, entered as one line:

     ```
     CLASSPATH=/usr/local/ibm/gsk7/classes/cfwk.zip:/usr
     /local/ibm/gsk/classes/gsk7cls.jar:$CLASSPATH
     ```

Once completed, IKEYCMD should run from any directory, using the following syntax:

```
java com.ibm.gsk.ikeyman.Ikeycmd command
```

Note: You can substitute `jre` for `java`, depending on whether you are you using JRE or JDK. For example:

```
jre com.ibm.gsk.ikeyman.Ikeycmd <command>
```

Key Management Utility command line options: The following table describes each action possible on a specified object:

Option	Description
dB	Fully qualified path name of a key database
-default_cert	Sets a certificate to use as the default certificate for client authentication (yes or no). Default is no.
-dn	X.500 distinguished name. Input as a quoted string of the following format (only CN, O, and C are required): `"CN=Jane Doe,O=IBM,OU=Java Development,L=Endicott, ST=NY,ZIP=13760,C=country"`
encryption	Strength of encryption used in certificate export command (strong or weak). Default is strong.
-expire	Expiration time of either a certificate or a database password (in days). Defaults are: 365 days for a certificate and 60 days for a database password.
-file	File name of a certificate or certificate request (depending on specified object).
-format	Format of a certificate (either ASCII for Base64_encoded ASCII or binary for Binary DER data). Default is ASCII.
-label	Label attached to a certificate or certificate request
-new_format	New format of key database
-new_pw	New database password
-old_format	Old format of key database
-pw	Password for the key database or PKCS#12 file. See Creating a new key database.
-size	Key size (512 or 1024). Default is 1024.
-stash	Indicator to stash the key database password to a file. If specified, the password will be stashed in a file.
-target	Destination file or database
-target_pw	Password for the key database if -target specifies a key database. See Creating a new key database.
-target_type	Type of database specified by -target operand (see -type)
-trust	Trust status of a CA certificate (enable or disable). Default is enable.

-type	Type of database. Allowable values are CMS (indicates a CMS key database), webdb (indicates a keyring), sslight (indicates an SSLight .class), or pkcs12 (indicates a PKCS#12 file).
-x509version	Version of X.509 certificate to create (1, 2 or 3). Default is 3.

Key Management Utility command line interface syntax: The syntax of the Java command line interface follows:

```
Java [-Dikeycmd.properties=<properties_file>] com.ibm.gsk.ikeyman.ikeycmd
<object> <action> [options]
```

where:

- -Dikeycmd.properties specifies the name of an optional properties file to use for this Java invocation. A default properties file, `ikeycmd.properties`, exists as a sample file that you can modify and use with any Java application.

Object includes one of the following:

- -keydb: Actions taken on the key database (either a CMS key database file, a WebDB key ring file, or SSLight class)
- -cert: Actions taken on a certificate
- -certreq: Actions taken on a certificate request
- -help: Displays help for the IKEYCMD invocations
- -version: Displays version information for IKEYCMD

Action represents the specific action to take on the object, and options represents the options, both required and optional, specified for the object and action pair.

The object and action keywords are positional and you must specify them in the selected order. However, options are not positional and you can specify them in any order, as an option and operand pair.

Key Management Utility command line parameters: The following table describes each action possible on a specified object:

Object	Actions	Description
-keydb	-changepw	Change the password for a key database
	-convert	Convert a key database from one format to another
	-create	Create a key database
	-delete	Delete the key database
	-stashpw	Stash the password of a key database into a file
-cert	-add	Add a CA certificate from a file into a key database
	-create	Create a self-signed certificate
	-delete	Delete a CA certificate

	-export	Export a personal certificate and its associated private key from a key database into a PKCS#12 file, or to another key database
	-extract	Extract a certificate from a key database
	-getdefault	Get the default personal certificate
	-import	Import a certificate from a key database or PKCS#12 file
	-list	List all certificates
	-modify	Modify a certificate. (Currently the only field you can modify is the **Certificate trust** field)
	-receive	Receive a certificate from a file into a key database
	-setdefault	Set the default personal certificate
	-sign	Sign a certificate stored in a file with a certificate stored in a key database and store the resulting signed certificate in a file
-certreq	-create	Create a certificate request
	-delete	Delete a certificate request from a certificate request database
	-details	List the detailed information of a specific certificate request
	-extract	Extract a certificate request from a certificate request database into a file
	-list	List all certificate requests in the certificate request database
	-recreate	Recreate a certificate request
-help		Display help information for the IKEYCMD command
-version		Display IKEYCMD version informatoin

Creating a new key database on the Linux for S/390 platform:

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.

You can create multiple databases if you prefer to keep certificates in separate databases.

To create a new key database using the IKEYCMD command-line interface, enter the following command (as one line):

```
gsk7cmd -keydb -create -db <filename> -pw <password> -type
<cms | jks | jceks | pks12> -expire <days> -stash
```

where:

- -db <filename> is the name of the database.
- -expire <days> is the number of days before password expires. This parameter is only valid for CMS key databases.
- -keydb Specifies the command is for the key database.
- -pw <password> is the password to access the key database.
- -type <cms | jks | jceks | pkcsk> is the database type. Note: IBM HTTP Server only handles a CMS key database.
- -stash stashes the password for the key database. When the -stash option is specified during the key database creation, the password is stashed in a file with a filename built as follows:

  ```
  <filename_of_key_database>.sth
  ```

  This parameter is only valid for CMS key databases. For example, if the database being created is named `keydb.kdb`, the stash filename is `keydb.sth`. Note: Stashing the password is required for IBM HTTP Server.

Setting the database password:

When you create a new key database, you specify a key database password. This password protects the private key. The private key is the only key that can sign documents or decrypt messages that are encrypted with the public key. Changing the key database password frequently is a good practice.

Use the following guidelines when specifying the password:

- The password must come from the U.S. English character set.
- The password must contain at least six characters and contain at least two nonconsecutive numbers. Make sure that the password does not consist of publicly obtainable information about you, such as the initials and birth date for you, your spouse, or children.
- Stash the password or enable secure sockets layer (SSL) password prompting. See Using SSL password prompting.

Keep track of expiration dates for the password. If the password expires, a message writes to the error log. The server starts, but a secure network connection does not exist if the password has expired.

Changing the database password on the Linux for S/390 platform:

This section describes how to change the password for a key database. A key database is used to store public keys that are used for secure connections.

To change the password, enter the following command (as one line):

```
gsk7cmd -keydb -changepw -db <filename>.kdb -pw <password>
-new_pw <new_password> -expire <days> -stash
```

where:

- `-db <filename>` is the name of the database.

- `-changepw` changes the password.
- `-keydb` specifies the command is for the key database.
- `-new_pw <new_password>` is the new key database password. This password must be different than the old password and cannot be a NULL string.
- `-pw <password>` is the password to access the key database.
- `-expire <days>` is the number of days before password expires. This parameter is only valid for CMS key databases.
- `-stash` stashes the password for the key database. This parameter is only valid for CMS key databases. Note: Stashing the password is required for the IBM HTTP Server.

Creating a new key pair and certificate request on the Linux for S/390 platform:

You find key pairs and certificate requests stored in a key database. This section provides information on how to create a key.

Complete the following steps to create a public and private key pair and certificate request:

1. Enter the following command (as one line):

   ```
   gsk7cmd -certreq -create -db <filename> -pw <password> -label <label>
   -dn <distinguished_name> -size <1024 | 512> -file <filename>
   ```

   where:

   - `-certreq` specifies a certificate request.
   - `-create` specifies a create action.
   - `-db <filename>` specifies the name of the database.
   - `-pw` is the password to access the key database.
   - `label` indicates the label attached to the certificate or certificate request.
   - `dn <distinguished_name>` indicates an X.500 distinguished name. Input as a quoted string of the following format (only CN, O, and C are required): CN=common_name, O=organization, OU=organization_unit, L=location, ST=state, province, C=country

     Note: For example, `"CN=weblinux.raleigh.ibm.com,O=IBM,OU=IBM HTTP Server,L=RTP,ST=NC,C=US"`
   - `-size <1024 | 512>` indicates a key size of 512 or 1024.
   - `-file <filename>` is the name of the file where the certificate request will be stored.
2. Verify that the certificate was successfully created:
   a. View the contents of the certificate request file you created.
   b. Make sure the key database recorded the certificate request:

      ```
      gsk7cmd -certreq -list -db <filename> -pw <password>
      ```

      You should see the label listed that you just created.
3. Send the newly-created file to a certificate authority.

Exporting keys on the Linux for S/390 platform:

This article describes how to export your key into another database or to a PKCS12 file. PKCS12 is a standard for securely storing private keys and certificates.

To export keys from another database, enter the following command:

```
gsk7cmd -cert -export -db <filename>
-pw <password> -label <label>
-type <cms | jks | jceks | pkcs12>
 -target <filename> - target_pw <password>
-target_type <cms | jks | jceks | pkcs12>
```

where:

- -cert specifies a personal certificate.
- -export specifies an export action.
- -db <filename> is the name of the database.
- -pw <password> is the password to access the key database.
- -label <label> is the label attached to the certificate.
- -target <filename> is the destination file or database. If the **target_type** is JKS, CMS, or JCEKS, the database specified here must exist.
- -target_pw is the password for the target key database.
- -target_type <cms | jks | jceks | pkcs12> is the type of database specified by the -target operand.
- -type <cms | jks | jceks | pkcs12> is the type of database key.

Importing keys on the Linux for S/390 platorm:

This article describes how to import keys using the IKEYCMD command line interface on the Linux for S/390 platform.

To import certificates from another key database, complete the following steps:

```
gsk7cmd -cert -import -db <filename> -pw <password> -label <label>
-type <cms | JKS | JCEKS| pkcs12> -new_label <label> -target <filename>
-target_pw <password> -target_type <cms | JKS |JCEKS | pkcs12>
```

where:

- -cert - specifies a certificate.
- -import - specifies an import action.
- -db <filename> - indicates the name of the database.
- -pw <password> - indicates the password to access the key database.
- -label <label> - indicates the label that is attached to the certificate.
- -new_label <label> - re-labels the certificate in the target key database.
- -type <cms | JKS | JCEKS | pkcs12> - specifies the type of database.
- -target <filename> - indicates the destination database.
- -target_pw <password> - indicates the password for the key database if -target specifies a key database
- -target_type <cms | JKS | JCEKS | pkcs12> - indicates the type of database that is specified by the -target opearnd.
- pfx - imported file in Microsoft .pfx file format.

Listing certificate authorities on the Linux for S/390 platform:

This article describes how to view trusted certificate authorities within a key database.

A trusted certificate authority issues and manages public keys for data encryption. A key database is used to share public keys that are used for secure connections.

To display a list of trusted CAs in a key database (command should be entered as one line):

```
gsk7cmd -cert -list CA -db < dbname > -pw <password> -type <cms | jks |jceks | pkcs12>
```

Creating a self-signed certificate on the Linux for S/390 platform:

It usually takes two to three weeks to get a certificate from a well-known CA. While waiting for an issued certificate, use IKEYMAN to create a self-signed server certificate to enable SSL sessions between clients and the server.

Use this procedure if you are acting as your own CA for a private Web network. To create a self-signed certificate, enter the following command (as one line):

```
gsk7cmd -cert -create -db <filename> -pw <password> -size <1024 | 512>
-dn <distinguished_name> -label label> -default_cert <yes | no> - expire <days>
```

where:

- `-cert` specifies a self-signed certificate.
- `-create` specifies a create action.
- `-db <filename>` is the name of the database.
- `-pw <password>` is the password to access the key database.
- `-dn <distinguished_name>` - indicates an X.500 distinguished name. Input as a quoted string of the following format (Only CN, O, and C are required): CN=common_name, O=organization, OU=organization_unit, L=location, ST=state, province, C=country

  For example, `"CN=weblinux.raleigh.ibm.com,O=IBM,OU=IBM HTTP Server,L=RTP,ST=NC,C=US"`
- `-label <label>` is a descriptive comment used to identify the key and certificate in the database.
- `-size` specifies the key size 512 or 1024.
- `-default_cert<yes | no>`specifies whether this is the default certificate in the key database.
- `-expire <days>` indicates the default validity period for new self-signed digital certificates is 365 days. The minimum is 1 day. The maximum is 7300 days (twenty years).

Receiving a signed certificate from a certificate authority on the Linux for S/390 platform:

This topic describes how to receive an electronically mailed certificate from a certificate authority (CA), that is designated as a trusted CA on your server. A certificate authority is a trusted third-party organization or company that issues digital certificates used to create digital signatures and public-private key pairs.

By default, the following CA certificates are stored in the key database and marked as trusted CA certificates:

- Verisign Class 2 OnSite Individual CA
- Verisign International Server CA -- Class 3
- VeriSign Class 1 Public Primary CA -- G2
- VeriSign Class 2 Public Primary CA -- G2
- VeriSign Class 3 Public Primary CA -- G2
- VeriSign Class 1 CA Individual Subscriber-Persona Not Validated
- VeriSign Class 2 CA Individual Subscriber-Persona Not Validated

- VeriSign Class 3 CA Individual Subscriber-Persona Not Validated
- RSA Secure Server CA (from RSA)
- Thawte Personal Basic CA
- Thawte Personal Freemail CA
- Thawte Personal Premium CA
- Thawte Premium Server CA
- Thawte Server CA

The certificate authority can send more than one certificate. In addition to the certificate for your server, the CA can also send additional signing certificates or intermediate CA certificates. For example, Verisign includes an intermediate CA certificate when sending a Global Server ID certificate. Before receiving the server certificate, receive any additional intermediate CA certificates. Follow the instructions in the Storing a CA certificate topic to receive intermediate CA certificates.

If the CA that issuing your CA-signed certificate is not a trusted CA in the key database, store the CA certificate first 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. For instructions, see Storing a CA certificate.

To receive the CA-signed certificate into a key database, enter the following command (as one line):

```
gsk7cmd -cert -receive -file <filename> -db <filename> -pw <password>
-format <ascii | binary> -label <label> -default_cert <yes | no>
```

where:

- `-cert` specifies a self-signed certificate.
- `-receive` specifies a receive action.
- `-file <filename>` is a file containing the CA certificate.
- `-db <filename>` is the name of the database.
- `-pw <password>` is the password to access the key database.
- `-format <ascii | binary>` specifies that the certificate authority might provide the CA certificate in either ASCII or binary format.
- `-default_cert <yes | no>` indicates whether this is the default certificate in the key database.
- `-label` specifies the label that is attached to a CA certificate.
- `-trust` indicates whether this CA can be trusted. Use enable options when receiving a CA certificate.

Displaying the default key in a key database on the Linux for S/390 platform:

To display a list of certificates in a key database and their expiration dates, enter the following:

```
gsk7cmd -cert -list -expiry < days > -db < filename >
-pw < paswsword > - type < type >
```

where:

- `-cert` indicates the operation applies to a certificate.
- `-list <all | personal | CA | site>` specifies a list action. The default is to list all certificates.

- `-expiry <days>` indicates that validity dates should be displayed. Specifying the number of days is optional, though when used will result in displaying all certificates that expire within that amount of days. To list certificates that have already expired, enter the value 0.
- `-db <filename>` is the name of the key database. It is used when you want to list a certificate for a specific key database.
- `-pw <password>` specifies the password to access the key database.
- `-type <cms | JKS | JCEKS | pkcs12>` specifies the type of database.

Storing a certificate authority certificate on the Linux for S/390 platform:

To store a certificate from a certificate authority (CA) who is not a trusted CA, enter the following:

```
gsk7cmd -cert -add -db <filename>.kdb -pw <password> -label <label>
-format <ascii | binary> -trust <enable | disable> -file <filename>
```

where:

- `-add` specifies an add action.
- `-cert` indicates the operation applies to a certificate.
- `-db <filename>` is the name of the database.
- `-file <filename>` specifies the file containing the CA certificate.
- `-format <ascii | binary>` indicates the certificate authorities might supply a binary or an ASCII file.
- `-label <label>` is the label attached to a certificate or certificate request.
- `-pw <password>` is the password to access the key database.
- `-trust <enable | disable>` indicates whether this CA can be trusted. Should be *yes*.

Storing the encrypted database password in stash file on the Linux for S/390 platform:

For a secure network connection, you can store the CMS encrypted database password in a stash file.

To store the password when creating a CMS database:

```
gsk7cmd -keydb -create -db <path_to_db>/<db_name> -pw <password> -type
cms -expire <days> -stash
```

To store the password after a CMS database has been created:

```
gsk7cmd -keydb -stashpw -db <db_name> -pw <password>
```

Secure Sockets Layer directives

This section describes the Secure Sockets Layer (SSL) directives.

Keyfile directive

The keyfile directive adjusts the verbosity of the messages recorded in the error logs.

When you specify a particular level, the server reports messages from all other levels of higher significance. For example, when you specify LogLevel info, the server reports messages with log levels of notice and warn. Specifying at least level crit is recommended.

Syntax `LogLevel error`

Scope Server configuration and virtual host
Default LogLevel error
Module `mod_ibm_ssl`
Multiple instances in the configuration file Permitted. Order of preference is top to bottom, first to last. If the client does not support cipher specifications, the connection closes.
Values The following available levels appear in order of decreasing significance:

Level	Description	Example
emerg	Emergencies: system rendered unusable.	″Child cannot open lock file. Exiting″
alert	Take immediate action.	″getpwuid: could not determine user name from uid″
crit	Critical conditions.	″socket: Failed to get a socket, exiting child″
error	Error conditions.	″Premature end of script headers″
warn	Warning conditions.	″child process 1234 did not exit, sending another SIGHUP″
notice	Normal, but significant condition.	″httpd: caught SIGBUS, attempting to dump core in ...″
info	Informational.	″Server seems busy, (you may need to increase StartServers, or Min/MaxSpareServers)...″
debug	Debug-level messages.	″Opening configuration file ...″

Related tasks

“Choosing the level of client authentication” on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

“Choosing the type of client authentication protection” on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

LogLevel directive

The LogLevel directive sets the key file to use.

Syntax `Keyfile [prompt]/`*`fully qualified path to key file`*`/keyfile.kdb`
Scope Global base and virtual host
Default None
Module `mod_ibm_ssl`
Multiple instances in the configuration file One instance per virtual host and global server

Values	File name of the key file. Use the prompt option to enable the HTTP server to prompt you for the Key file password during start up. See Using SSL Password Prompting. LogLevel

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

SSLAcceleratorDisable directive

The SSLAcceleratorDisable directive disables the accelerator device.

Syntax	`SSLAcceleratorDisable`
Scope	Virtual and global
Default	Accelerator device is enabled
Module	`mod_ibm_ssl`
Multiple instances in the configuration file	Not permitted.
Values	None. Place this directive anywhere inside of the configuration file, including inside a virtual host. During initialization, if the system determines that an accelerator device is installed on the machine, the system uses that accelerator to increase number of secure transactions. This directive does not take arguments.

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

SSLCacheDisable directive

The SSLCacheDisable directive disables the external SSL session ID cache on UNIX platforms.

Syntax	`SSLCacheDisable`
Scope	One per physical Apache server instance, allowed only outside of virtual host stanzas.
Default	None
Module	`mod_ibm_ssl`
Multiple instances in the configuration file	Not permitted.
Values	None. Valid on in UNIX environments.

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

SSLCacheEnable directive

The SSLCacheEnable directive enables the external SSL session ID cache on UNIX platforms.

Syntax	`SSLCacheEnable`
Scope	One per physical Apache server instance, allowed only outside of virtual host stanzas.
Default	None
Module	`mod_ibm_ssl`
Multiple instances in the configuration file	Not permitted.
Values	None. Valid on in UNIX environments.

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

SSLCacheErrorLog directive

The SSLCacheErrorLog directive sets the file name for session ID cache error logging on UNIX platforms.

Syntax	`SSLCacheErrorLog /usr/HTTPServer/log/sidd_logg`
Scope	One per physical Apache server instance, allowed only outside of virtual host stanzas.
Default	None
Module	`mod_ibm_ssl`
Multiple instances in the configuration file	Not permitted.
Values	Valid file name. Valid on in UNIX environments.

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

SSLCachePath directive

The SSLCachePath directive specifies the path to the session ID caching daemon executable on UNIX platforms.

Syntax	`SSLCachePath /usr/HTTPServer/log/sidd`
Scope	One per physical IBM HTTP Server.
Default	`<server-root>/bin/sidd`
Module	`mod_ibm_ssl`
Multiple instances in the configuration file	Not permitted.

Values	Valid path name. Valid on in UNIX environments.

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

SSLCachePortFilename directive

The SSLCachePortFilename directive sets the file name for the UNIX domain socket that is used for communication between the server instances and the session ID cache daemon.

Syntax	`SSLCachePath /usr/HTTPServer/log/sidd`
Scope	One per physical Apache server instance, permitted only outside of the virtual host stanzas.
Default	If this directive is not specified and the cache is enabled, the server attempts to use the `<server-root>/logs/siddport` file.
Module	`mod_ibm_ssl`
Multiple instances in the configuration file	Not permitted.
Values	Valid path name. The Web server deletes this file during startup; do not name. Valid on in UNIX environments.

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

SSLCacheTraceLog directive

The SSLCacheTraceLog directive specifies the trace log to the appropriate session ID trace messages log on UNIX platforms.

Syntax	`SSLCacheTraceLog /usr/HTTPServer/log/sidd-trace.log`
Scope	One per physical IBM HTTP Server.
Default	None.
Module	`mod_ibm_ssl`
Multiple instances in the configuration file	Not permitted.
Values	Valid path name. Valid on in UNIX environments.

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

SSLCipherBan directive

The SSLCipherBan directive denies access to an object if the client attempts the specified cipher.

Syntax	`SSLCipherBan <cipher_specification>`
Scope	Multiple instances per directory stanza.
Default	None.
Module	`mod_ibm_ssl`
Multiple instances in the configuration file	Permitted per directory stanza. Order of preference is top to bottom.
Values	See "Version 2 cipher specifications" on page 35 and "SSL Version 3 and TLS Version 1 cipher specifications" on page 35.

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

SSLCipherRequire directive

The SSLCipherRequire directive allows limited access to objects according to specified ciphers.

Syntax	`SSLCipherRequire <cipher_specification>`
Scope	Multiple instances per directory stanza.
Default	None.
Module	`mod_ibm_ssl`
Multiple instances in the configuration file	Permitted per directory stanza. Order of preference is top to bottom.
Values	See "Version 2 cipher specifications" on page 35 and "SSL Version 3 and TLS Version 1 cipher specifications" on page 35.

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

SSLCipherSpec directive

The SSLCipherSpec directive specifies a cipher specification that you can use in a secure transaction.

Syntax	`SSLCipherSpec short name` or `SSLCipherSpec long name`
Scope	Virtual host.

Default	If nothing is specified, the server uses all of the cipher specifications available from the installed GSK library.
Module	`mod_ibm_ssl`
Multiple instances in the configuration file	Permitted. Order of preference is top to bottom, first to last. If the client does not support the cipher specifications, the connection closes.
Values	See "Version 2 cipher specifications" on page 35 and "SSL Version 3 and TLS Version 1 cipher specifications" on page 35.

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

SSLClientAuth directive

The SSLClientAuth directive sets the mode of client authentication to use (none (0), optional (1), or required (2)).

Syntax	`SSLClientAuth <level required> [crl]`
Scope	Virtual host.
Default	`SSLClientAuth none`
Module	`mod_ibm_ssl`
Multiple instances in the configuration file	One instance per virtual host.
Values	• 0/None: No client certificate requested. • 1/Optional: Client certificate requested, but not required. • 2/Required: Valid client certificate required. • CRL: Turns crl on and off inside an SSL virtual host. If you use certificate revocation list (CRL), you need to specify `crl` as a second argument for SSLClientAuth. For example: `SSLClientAuth 2 crl`. If you do not specify `crl`, you cannot perform CRL in an SSL virtual host. If you specify the value 0/None, you cannot use the CRL option.

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

SSLClientAuthGroup directive

The SSLClientAuthGroup directive enables you to group client certificate attributes together for use in the SSLClientAuthRequire directive. .

Syntax	`<SSLClientAuth` *group name <logic string>*`>`
Scope	Multiple instances per directory stanza.
Default	None.
Module	`mod_ibm_ssl`
Multiple instances in the configuration file	Permitted. The function joins these directives by "AND".
Values	Logical expression consisting of attribute checks linked with AND, OR, NOT, and parentheses.

Description of valid logical expressions. The following section provides a description of examples with valid logical expressions. For example: `SSLClientAuthGroup (CommonName = "Fred Smith" OR CommonName = "John Deere") AND Org = IBM` means that the object is not served, unless the client certificate contains a common name of either Fred Smith or John Deere and the organization is IBM. The only valid comparisons for the attribute checks, are equal and not equal (= and !=). You can link each attribute check with AND, OR, or NOT (also &&, `||`, and !). Use parentheses to group comparisons. If the value of the attribute contains a nonalphanumeric character, you must delimit the value with quotes.

A listing of the valid attributes follows:

- CommonName
- Country
- Email
- Group
- IssuerCommonName
- IssuerCountry
- IssuerEmail
- IssuerLocality
- IssuerOrg
- IssuerOrgUnit
- IssuerStateOrProvince
- Locality
- Org
- OrgUnit
- StateOrProvince

A listing of valid short names follows:

- CN
- C
- E
- G
- ICN
- IC
- IE
- IL

- IO
- IOU
- IST
- L
- O
- OU
- ST

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

SSLClientAuthRequire directive

The SSLClientAuthRequire directive enables extensive validation of client certificate information before serving an object.

Syntax	`<SSLClientAuthRequire CommonName = Richard`
Scope	Multiple instances per directory stanza.
Default	None.
Module	`mod_ibm_ssl`
Multiple instances in the configuration file	Permitted. The function joins these directives by "AND".
Values	Logical expression consisting of attribute checks linked with AND, OR, NOT, and parentheses.

Description of valid logical expressions. The following section provides a description of examples with valid logical expressions. For example: `SSLClientAuthGroup (CommonName = "Fred Smith" OR CommonName = "John Deere") AND Org = IBM` means that the object is not served, unless the client certificate contains a common name of either Fred Smith or John Deere and the organization is IBM. The only valid comparisons for the attribute checks, are equal and not equal (= and !=). You can link each attribute check with AND, OR, or NOT (also &&, `||`, and !). Use parentheses to group comparisons. If the value of the attribute contains a nonalphanumeric character, you must delimit the value with quotes.

A listing of the valid attributes follows:

- CommonName
- Country
- Email
- Group
- IssuerCommonName
- IssuerCountry
- IssuerEmail
- IssuerLocality
- IssuerOrg
- IssuerOrgUnit

- IssuerStateOrProvince
- Locality
- Org
- OrgUnit
- StateOrProvince

A listing of valid short names follows:
- CN
- C
- E
- G
- ICN
- IC
- IE
- IL
- IO
- IOU
- IST
- L
- O
- OU
- ST

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

SSLCRLHostname directive

The SSLCRLHostname directive specifies the TCP/IP name or address of LDAP server where CRL database resides.

Syntax	`<SSLCRLHostName <TCP/IP name or address>`
Scope	Global server or virtual host.
Default	Disabled by default.
Module	`mod_ibm_ssl`
Multiple instances in the configuration file	One instance per virtual host and global server.
Values	TCP/IP name or address of the LDAP Server

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

SSLCRLPort directive

The SSLCRLPort directive specifies the port of the LDAP server where the CRL database resides.

Syntax	`SSLCRLPort <port number>`
Scope	Global server or virtual host.
Default	Disabled by default.
Module	`mod_ibm_ssl`
Multiple instances in the configuration file	One instance per virtual host and global server.
Values	Port of LDAP server; default = 389.

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

SSLCRLUserID directive

The SSLCRLUserID directive specifies the user ID to send to the LDAP server, where CRL database resides.

Syntax	`SSLCRLUserID <[prompt] <userid>`
Scope	Global server or virtual host.
Default	Defaults to anonymous if you do not specify a user ID.
Module	`mod_ibm_ssl`
Multiple instances in the configuration file	One instance per virtual host and global server.
Values	User ID of LDAP server. Use the prompt option to enable the HTTP server to prompt you for the password needed to access the LDAP server during start up. See Using SSL Password Prompting.

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

SSLDisable directive

The SSLDisable directive disables SSL for the virtual host.

Syntax	`SSLDisable`
Scope	Global server or virtual host.
Default	Disabled by default.
Module	`mod_ibm_ssl`
Multiple instances in the configuration file	One instance per virtual host and global server.
Values	None.

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

SSLEnable directive

The SSLEnable directive enables SSL for the virtual host.

Syntax	`SSLEnable`
Scope	Global server or virtual host.
Default	Disabled by default.
Module	`mod_ibm_ssl`
Multiple instances in the configuration file	One instance per virtual host and global server.
Values	None.

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

SSLFakeBasicAuth directive

The SSLFakeBasicAuth directive enables the fake basic authentication support.

This support enables the client certificate distinguished name to become the user portion of the user and password basic authentication pair. Use **password** for the password.

Syntax	`SSLFakeBasicAuth`
Scope	Within a directory stanza, used along with AuthName, AuthType, and require directives.
Default	None.
Module	`mod_ibm_ssl`
Multiple instances in the configuration file	One instance per virtual host and global server.
Values	None.

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

SSLFIPSDisable directive

The SSLFIPSDisable directive disables Federal Information Processing Standards (FIPS).

Syntax	`SSLFIPSDisable`
Scope	Virtual and global.
Default	Disabled by default.
Module	`mod_ibm_ssl`

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

SSLFIPSEnable directive

The SSLFIPSEnable directive enables Federal Information Processing Standards (FIPS).

Syntax	`SSLFIPSEnable`
Scope	Virtual and global.
Default	Disabled by default.
Module	`mod_ibm_ssl`

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

SSLPKCSDriver directive

The SSLPKCSDriver directive identifies the fully qualified name to the module, or driver used to access the PKCS11 device.

Syntax	*Fully qualified name to module used to access PKCS11 device*>. If the module exists in the user's path, then specify just the name of the module.
Scope	Global server or virtual host.
Default	None.
Module	`mod_ibm_ssl`
Multiple instances in the configuration file	One instance per virtual host and global server.
Values	Path and name of PKCS11 module or driver.

The default locations of the modules for each PKCS11 device follow, platform:

- nCipher
  - AIX: `/opt/nfast/toolkits/pkcs11/libcknfast.so`
  - HP: `/opt/nfast/toolkits/pkcs11/libcknfast.sl`
  - Solaris: `/opt/nfast/toolkits/pkcs11/libcknfast.so`
  - Windows: `c:\nfast\toolkits\pkcs11\cknfast.dll`
- IBM 4758
  - AIX: `/usr/lib/pkcs11/PKCS11_API.so`

- Windows: `$PKCS11_HOME\bin\nt\cryptoki.dll`

- IBM e-business Cryptographic Accelerator
  - AIX: `/usr/lib/pkcs11/PKCS11_API.so`

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

SSLServerCert directive

The SSLServerCert directive sets the server certificate to use for this virtual host.

Syntax	`SSLServerCert [prompt] *my_certificate_label;* on PKCS11 device - SSLServerCert *mytokenlabel:mykeylabel*`
Scope	IP-based virtual hosts.
Default	None.
Module	`mod_ibm_ssl`
Multiple instances in the configuration file	One instance per virtual host and global server.
Values	Certificate label. Use the /prompt option to enable the HTTP server to prompt you for the Crypto token password during start up. See Using SSL Password Prompting. Use no delimiters around the certificate label. Ensure that the label is contained on one line; leading and trailing white space is ignored.

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

SSLStashfile directive

The SSLStashfile directive indicates path to file with file name, containing the encrypted password for opening the PKCS11 device.

Syntax	`sslstash [-c] <file> <function> <password>`, where: • -c = Create a new stash file. If not specified, the server updates an existing stash file • File = Fully qualified name of the file to create or update • Function = Function with which to use the password Valid values include crl or crypto • Password = The password to stash • Usage - `sslstash -c conf\pkcs11.passwd crypto pkcs11`

Scope	Virtual host and global server.
Default	None.
Module	`mod_ibm_ssl`
Multiple instances in the configuration file	One instance per virtual host and global server.
Values	Path with file name. Locate an **sslstash** command in the `bin` directory of the IBM HTTP Server, for UNIX, and the server installation root for the Windows platform. Use this command to store the password for the PKCS11 device. The stash file created after using the **sslstash** command can hold two different passwords for two different functions: crl and cryptography.

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

SSLV2Timeout directive

The SSLV2Timeout directive sets the timeout for SSL Version 2 session IDs.

Syntax	`SSLV2Timeout 60`
Scope	Global base and virtual host.
Default	40
Module	`mod_ibm_ssl`
Multiple instances in the configuration file	One instance per virtual host and global server.
Values	0 to 100 seconds.

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

SSLV3Timeout directive

The SSLV3Timeout directive sets the timeout for SSL Version 3 session IDs.

Syntax	`SSLV3Timeout 1000`
Scope	Global base and virtual host.
Default	120
Module	`mod_ibm_ssl`
Multiple instances in the configuration file	One instance per virtual host and global server.
Values	0 to 86400 seconds.

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

SSLVersion directive

The SSLVersion directive enables object access rejection, if the client attempts to connect with an SSL protocol version other than the one specified.

Syntax	`SSLVersion ALL`
Scope	One per directory stanza.
Default	None.
Module	`mod_ibm_ssl`
Multiple instances in the configuration file	One instance per virtual host and global server.
Values	SSLV2|SSLV3|TLSV1|ALL

Related tasks

"Choosing the level of client authentication" on page 33
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

"Choosing the type of client authentication protection" on page 34
If you enable client authentication, the server validates clients by checking for trusted certificate authority (CA) root certificates in the local key database.

Lightweight Directory Access Protocol

This section describes topics on how to set up and use the Lightweight Directory Access Protocol (LDAP) with IBM HTTP Server.

LDAP concepts and tasks

This section describes topics on how to use the Lightweight Directory Access Protocol (LDAP) with IBM HTTP Server.

Introducing the Lightweight Directory Access Protocol

This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

LDAP is a standard protocol that provides a means of storing and retrieving information about people, groups, or objects on a centralized X.500 or LDAP directory server. X.500 enables that information to be organized and queried, using LDAP, from multiple web servers using a variety of attributes. LDAP queries can be as simple or complex as is required to identify a desired individual entity or group of entities. LDAP reduces required system resources by including only a functional subset of the original X.500 Directory Access Protocol (DAP).

The IBM HTTP Server LDAP module enables the use of an X.500 directory server for authentication and authorization purposes. IBM HTTP Server can use this capability to limit access of a resource to a controlled set of users. This capability reduces the administrative overhead usually required to maintain user and group information for each individual Web server.

You can configure the IBM HTTP Server LDAP module to use TCP/IP or Secure Sockets Layer (SSL) connections to the X.500 directory server. The LDAP module can be configured to reference a single LDAP server or multiple servers.

X.500 overview. X.500 provides a directory service with components that are capable of more efficient retrieval. LDAP uses two of these components: The information model, which determines the form and character, and the namespace, which enables information indexing and referencing.

The X.500 directory structure differs from other directories in information storage and retrieval. This directory service associates information with attributes. A query based on attributes generates and passes to the LDAP server, and the server returns the respective values. LDAP uses a simple, string-based approach for representing directory entries.

An X.500 directory consists of typed entries that are based on the ObjectClass attribute. Each entry consists of attributes. The ObjectClass attribute identifies the type of entry, for example, a person or organization, that determines the required and optional attributes.

You can divide entries, arranged in a tree structure, among servers in geographical and organizational distribution. The directory service names entries, according to their position within the distribution hierarchy, by a distinguished name (DN).

Lightweight Directory Access Protocol overview. Accessing an X.500 directory requires the Directory Access Protocol (DAP). However, DAP requires large amounts of system resources and support mechanisms to handle the complexity of the protocol. To enable desktop workstations to access the X.500 directory service, LDAP was introduced.

LDAP, a client and server-based protocol can handle some of the heavy resources required by DAP clients. An LDAP server can only return results or errors to the client, requiring little from the client. If unable to answer a client request, an LDAP Server must chain the request to another X.500 server. The server must complete the request, or return an error to the LDAP server, which in turn passes the information to the client.

Related tasks

"Configuring Lightweight Directory Access Protocol on IBM HTTP Server" on page 71
This section describes how to configure LDAP to protect files on IBM HTTP Server.

Querying the Lightweight Directory Access Protocol server

The Lightweight Directory Access Protocol (LDAP) accesses the X.500 directory using text strings called filters. When these query strings pass to the LDAP server, the server returns the requested portions of the specified entity.

LDAP filters use attributes to simplify queries to the LDAP server. For example, you can use a filter such as "objectclass=person" to limit your query to entities that represent people as opposed to groups or equipment.

- **To authorize a user as a member of a group, add the following directive to the configuration file:**

  ```
  LDAPRequire group "group_name"
  ```

  For example:

```
LDAPRequire group "Administrative Users"
```

- **To authorize a user by filter, add the following directive to the configuration file:**

  ```
  LDAPRequire filter "ldap_search_filter"
  ```

  For example, to enable access to the resource by a programmer in your department:

  ```
  LDAPRequire filter"(&(objectclass=person)(cn=*)(ou=programmer)(o=department))"
  ```

  or to enable access for John Doe only:

  LDAPRequire filter ″(&(objectclass=person)(cn=John Doe))″

  Related tasks

  “Configuring Lightweight Directory Access Protocol on IBM HTTP Server” This section describes how to configure LDAP to protect files on IBM HTTP Server.

Configuring Lightweight Directory Access Protocol on IBM HTTP Server

This section describes how to configure LDAP to protect files on IBM HTTP Server.

1. Edit the `httpd.conf` IBM HTTP Server configuration file.
2. Determine the resource you want to limit access to (for example <Directory ″/secure_info″>).
3. Add the following directives to the container to be protected with values specific to your environment:
   - `LdapConfigFile <path_to_ldap.prop>`
   - `AuthType Basic`
   - `AuthName ″Title of your protected Realm″`
   - `require valid-user`
4. To restrict access to a group, also add the following directive:

   ```
   LDAPRequire group "my_group_name"
   ```

   You can also restrict access to a pattern of user with the following directive:

   ```
   LDAPRequire filter "LDAP_attribute_filter"
   ```

   For example:

   ```
   LDAPRequire filter "(&(objectclass=person) (cn=*) (ou=IHS) (o=IBM))"
   ```

   Another example is to require a specific user with the following directive:

   ```
   LDAPRequire filter "(&(objectclass=person) (cn=John Doe))"
   ```
5. Edit the `ldap.prop` configuration file. If you do not have one yet, you can use the `ldap.prop.sample` file that ships with IBM HTTP Server. If you have questions about the correct values, check with your LDAP server administrator. Update the following directives with values that are correct for your environment:
   a. Enter the Web server connection information.
   b. Change the ldap.transport value if you intend to use SSL.
   c. Enter the client connection information.
   d. Update the filter values, if necessary.
   e. Adjust the timeout settings, if necessary.
   f. Fill in the Key database and stash file information, if necessary.

If you have the LDAP client installed on your computer, you can use ldapsearch as a tool to test the values you intend to use for the various values.

Related concepts

"Introducing the Lightweight Directory Access Protocol" on page 69
This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

"Using Secure Sockets Layer and the Lightweight Directory Access Protocol module"
IBM HTTP Server provides the ability to use a secure connection between the LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

Using Secure Sockets Layer and the Lightweight Directory Access Protocol module

IBM HTTP Server provides the ability to use a secure connection between the LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

To enable this feature, edit the `ldap.prop` LDAP configuration file and change the value of `ldap.transport` to `SSL`. Create or obtain a certificate database file (`X.kdb`) and a password stash file (`Y.sth`). You can use IKEYMAN to obtain a key database file. You must use the **ldapstash** program to create the stash file. You will also need to change the values for `ldap.URL` and `ldap.group.URL` to use port 636 instead of port 389.

The key database file contains the certificates which establish identity. The LDAP server can require that the Web server provide a certificate before allowing queries. When using a certificate with an SSL connection between the LDAP module and the LDAP server, the user ID that IBM HTTP Server is configured to use must have write permission to the key database file containing the certificate.

Certificates establish identity to prevent other users from stealing or overwriting your certificates (and therefore your identity). If someone has read permission to the key database file, they can retrieve the user's certificates and masquerade as that user. Grant read or write permission only to the owner of the key database file.

Supported Lightweight Directory Access Protocol servers

IBM HTTP Server supports the following LDAP servers:

- iPlanet/Netscape Directory Server
- IBM SecureWay Directory Server
- Microsoft Active Directory

LDAP directives

This section describes the LDAP directives.

LdapConfigfile directive

The LdapConfigFile directive indicates the name of the LDAP properties file associated with a group of LDAP parameters.

Syntax	`LdapConfigFile <Fully qualified path to configuration file>`
Scope	Single instance per directory stanza
Default	`c:\program files\ibm http server\conf\ldap.prop.sample`
Module	`mod_ibm_ldap`
Multiple instances in the configuration file	yes
Values	Fully qualified path to a single configuration file. Use this directive in the `httpd.conf` file.

Related concepts

"Introducing the Lightweight Directory Access Protocol" on page 69
This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

"Using Secure Sockets Layer and the Lightweight Directory Access Protocol module" on page 72
IBM HTTP Server provides the ability to use a secure connection between the LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

LdapRequire directive

The LdapRequire directive indicates the group when using LDAP authentication.

Syntax	`LDAPRequire filter <filter name>` or `LDAPRequire group <group1 [group2.group3....]>`
Scope	Single instance per directory stanza
Default	None
Module	`mod_ibm_ldap`
Multiple instances in the configuration file	yes
Values	`LDAPRequire filter "(&(objectclass=person)(cn=*)(ou=IHS)(o=IBM))", or LDAPRequire group "sample group".` Use this directive in the `httpd.conf` file.

Related concepts

"Introducing the Lightweight Directory Access Protocol" on page 69
This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

"Using Secure Sockets Layer and the Lightweight Directory Access Protocol module" on page 72
IBM HTTP Server provides the ability to use a secure connection between the LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

Ldap.application.authType directive

The Ldap.application.authType directive specifies the method for authenticating the Web server to the LDAP server.

Syntax	`ldap.application.authType=None`
Scope	Single instance per directory stanza

Default	None
Module	`mod_ibm_ldap`
Multiple instances in the configuration file	yes
Values	• None: If the LDAP server does not require the Web server to authenticate. • Basic: Uses the distinguished name (DN) of the Web server as the user ID, and the password stored in the stash file, as the password.

Related concepts

"Introducing the Lightweight Directory Access Protocol" on page 69
This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

"Using Secure Sockets Layer and the Lightweight Directory Access Protocol module" on page 72
IBM HTTP Server provides the ability to use a secure connection between the LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

Ldap.application.DN directive

The Ldap.application.DN directive indicates the distinguished name (DN) of the Web server. Use this name as the user name when accessing an LDAP server using basic authentication. Use the entry specified in the LDAP server to access the directory server.

Syntax	`ldap.application.DN=cn=ldapadm,ou=ihs test,o=IBM,c=US`
Scope	Single instance per directory stanza
Default	None
Module	`mod_ibm_ldap`
Multiple instances in the configuration file	yes
Values	Distinguished name

Related concepts

"Introducing the Lightweight Directory Access Protocol" on page 69
This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

"Using Secure Sockets Layer and the Lightweight Directory Access Protocol module" on page 72
IBM HTTP Server provides the ability to use a secure connection between the LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

Ldap.application.password.stashFile directive

The Ldap.application.password.stashFile directive indicates the name of the stash file containing the encrypted password for the application to authenticate to the LDAP server when Server Authentication type is Basic.

Syntax	`ldap.application.password.stashFile=c:\IHS\ldap.sth`
Scope	Single instance per directory stanza

Default	None
Module	`mod_ibm_ldap`
Multiple instances in the configuration file	yes
Values	Fully qualified path to the stash file. You can create this stash file with the **ldapstash** command.

Related concepts

"Introducing the Lightweight Directory Access Protocol" on page 69
This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

"Using Secure Sockets Layer and the Lightweight Directory Access Protocol module" on page 72
IBM HTTP Server provides the ability to use a secure connection between the LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

Ldap.cache.timeout directive

The ldap.cache.timeout directive caches responses from the LDAP server. If you configure the Web server to run as multiple processes, each process manages its own copy of the cache.

Syntax	`ldap.cache.timeout= <secs>`
Scope	Single instance per directory stanza
Default	600
Module	`mod_ibm_ldap`
Multiple instances in the configuration file	yes
Values	The maximum length of time, in seconds, a response returned from the LDAP server remains valid.

Related concepts

"Introducing the Lightweight Directory Access Protocol" on page 69
This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

"Using Secure Sockets Layer and the Lightweight Directory Access Protocol module" on page 72
IBM HTTP Server provides the ability to use a secure connection between the LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

Ldap.group.attributes directive

The ldap.group.attributes directive indicates the filter used to determine if a distinguished name (DN) is an actual group through an LDAP search.

Syntax	`ldap.group.attribute= attribute1 [attribute2...]`
Scope	Single instance per directory stanza
Default	`groupofnames groupofuniquenames`
Module	`mod_ibm_ldap`
Multiple instances in the configuration file	yes

Values	Filter name

Related concepts

"Introducing the Lightweight Directory Access Protocol" on page 69
This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

"Using Secure Sockets Layer and the Lightweight Directory Access Protocol module" on page 72
IBM HTTP Server provides the ability to use a secure connection between the LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

Ldap.group.dnattributes directive

The ldap.group.dnattributes specifies the filter used to determine, through an LDAP search, if a distinguished name (DN) is an actual group.

Syntax	`ldap.group.memberattribute = <ldap filter>`
Scope	Single instance per directory stanza
Default	`groupofnames groupofuniquenames`
Module	`mod_ibm_ldap`
Multiple instances in the configuration file	yes
Values	An ldap filter - See the `ldap.prop.sample` directive for more information on the use of this directive.

Related concepts

"Introducing the Lightweight Directory Access Protocol" on page 69
This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

"Using Secure Sockets Layer and the Lightweight Directory Access Protocol module" on page 72
IBM HTTP Server provides the ability to use a secure connection between the LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

Ldap.group.memberattribute directive

The ldap.group.memberattribute directive specifies the attribute to retrieve unique groups from an existing group.

Syntax	`ldap.group.memberattribute = <attribute>`
Scope	Single instance per directory stanza
Default	`uniquegroup`
Module	`mod_ibm_ldap`
Multiple instances in the configuration file	yes
Values	An ldap attribute - See the `ldap.prop.sample` directive for more information on the use of this directive.

Related concepts

"Introducing the Lightweight Directory Access Protocol" on page 69
This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

"Using Secure Sockets Layer and the Lightweight Directory Access Protocol module" on page 72
IBM HTTP Server provides the ability to use a secure connection between the LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

Ldap.group.memberAttributes directive

The ldap.group.memberAttributes directive serves as a means to extract group members, once the function finds a group entry in an LDAP directory.

Syntax	`ldap.group.memberAttributes= attribute [attribute2....]`
Scope	Single instance per directory stanza
Default	`member` and `uniquemember`
Module	`mod_ibm_ldap`
Multiple instances in the configuration file	yes
Values	Must equal the distinguished names of the group members. You can use more than one attribute to contain member information.

Related concepts

"Introducing the Lightweight Directory Access Protocol" on page 69
This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

"Using Secure Sockets Layer and the Lightweight Directory Access Protocol module" on page 72
IBM HTTP Server provides the ability to use a secure connection between the LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

Ldap.group.name.filter directive

The ldap.group.name.filter directive indicates the filter LDAP uses to search for group names.

Syntax	`ldap.group.name.filter = <group name filter>`
Scope	Single instance per directory stanza
Default	`(&(cn=%v1)(|(objectclass=groupOfNames)(objectclass=groupOfUniqueNames))`
Module	`mod_ibm_ldap`
Multiple instances in the configuration file	yes
Values	An LDAP filter. See Querying the LDAP server using LDAP search filters.

Related concepts

"Introducing the Lightweight Directory Access Protocol" on page 69
This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

"Using Secure Sockets Layer and the Lightweight Directory Access Protocol module" on page 72
IBM HTTP Server provides the ability to use a secure connection between the LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

Ldap.group.URL directive

The ldap.group.URL directive specifies a different location for a group on the same LDAP server. You cannot use this directive to specify a different LDAP server from that specified in the ldap.URL directive.

Syntax	`ldap.group.URL = ldap://<hostname:port>/<BaseDN>`
Scope	Single instance per directory stanza
Default	None
Module	`mod_ibm_ldap`
Multiple instances in the configuration file	yes
Values	• host name: Host name of the LDAP server. • port number: Optional port number on which the LDAP server listens. The default for TCP connections is 389. If you use SSL, you must specify the port number. • BaseDN: Provides the root of the LDAP tree in which to perform the search for groups.

This property becomes required if the LDAP URL for groups differs from the URL specified by the ldap.URL property.

Related concepts

"Introducing the Lightweight Directory Access Protocol" on page 69
This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

"Using Secure Sockets Layer and the Lightweight Directory Access Protocol module" on page 72
IBM HTTP Server provides the ability to use a secure connection between the LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

Ldap.key.file.password.stashfile directive

The ldap.key.file.password.stashfile directive indicates the stash file containing the encrypted keyfile password; use the ldapstash command to create this stash file.

Syntax	`ldap.key.file.password.stashfile =d:\ <Key password file name>`
Scope	Single instance per directory stanza
Default	None
Module	`mod_ibm_ldap`
Multiple instances in the configuration file	yes
Values	Fully qualified path to the stash file.

Related concepts

"Introducing the Lightweight Directory Access Protocol" on page 69
This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

"Using Secure Sockets Layer and the Lightweight Directory Access Protocol module" on page 72
IBM HTTP Server provides the ability to use a secure connection between the LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

Ldap.key.fileName directive

The ldap.key.fileName directive indicates the file name of the key file database. This option becomes required when you use Secure Sockets Layer (SSL).

Syntax	`ldap.key.fileName=d:\<Key file name>`
Scope	Single instance per directory stanza
Default	None
Module	`mod_ibm_ldap`
Multiple instances in the configuration file	yes
Values	Fully qualified path to the key file.

Related concepts

"Introducing the Lightweight Directory Access Protocol" on page 69
This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

"Using Secure Sockets Layer and the Lightweight Directory Access Protocol module" on page 72
IBM HTTP Server provides the ability to use a secure connection between the LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

Ldap.key.label directive

The ldap.key.file.password.stashfile directive indicates the certificate label name the Web server uses to authenticate to the LDAP server.

Syntax	`My Server Certificate`
Scope	Single instance per directory stanza
Default	None
Module	`mod_ibm_ldap`
Multiple instances in the configuration file	yes
Values	A valid label used in the key database file. This label becomes required only when using Secure Sockets Layer (SSL) and the LDAP server requests client authentication from the Web server.

Related concepts

"Introducing the Lightweight Directory Access Protocol" on page 69
This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

"Using Secure Sockets Layer and the Lightweight Directory Access Protocol module" on page 72

IBM HTTP Server provides the ability to use a secure connection between the LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

Ldap.realm directive

The ldap.key.realm directive indicates the name of the protected area, as seen by the requesting client.

Syntax	`ldap.realm==<Protection Realm>`
Scope	Single instance per directory stanza
Default	None
Module	`mod_ibm_ldap`
Multiple instances in the configuration file	yes
Values	A description describing the protected page.

Related concepts

"Introducing the Lightweight Directory Access Protocol" on page 69
This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

"Using Secure Sockets Layer and the Lightweight Directory Access Protocol module" on page 72
IBM HTTP Server provides the ability to use a secure connection between the LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

Ldap.search directive

The ldap.search directive searches subgroups when specifying the LdapRequire group <group> directives. Groups can contain both individual members and other groups.

Syntax	`ldap.search = <secs>`
Scope	Single instance per directory stanza
Default	1
Module	`mod_ibm_ldap`
Multiple instances in the configuration file	yes

Values

An integer. When doing a search for a group, if a member in the process of authentication is not a member of the required group, any subgroups of the required group are also searched. For example:

```
group1 >group2 (group2 is a member of group1)
group2 >group3 (group3 is a member of group2)
group3 >jane   (jane is a member of group3)
```

If you search for jane and require her as a member of group1, the search fails with the default ldap.search.depth value of 1. If you specify `ldap.group.search.depth>2`, the search succeeds.

Use `ldap.group.search.depth=<depth to search -- number>` to limit the depth of subgroup searches. This type of search can become very intensive on an LDAP server. Where group1 has group2 as a member, and group2 has group1 as a member, this directive limits the depth of the search. In the previous example, group1 has a depth of 1, group2 has a depth of 2 and group3 has a depth of 3.

Related concepts

"Introducing the Lightweight Directory Access Protocol" on page 69
This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

"Using Secure Sockets Layer and the Lightweight Directory Access Protocol module" on page 72
IBM HTTP Server provides the ability to use a secure connection between the LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

Ldap.search.timeout directive

The ldap.search.timeout directive indicates the maximum time, in seconds, to wait for an LDAP server to complete a search operation.

Syntax	`ldap.search.timeout = <secs>`
Scope	Single instance per directory stanza
Default	10
Module	`mod_ibm_ldap`
Multiple instances in the configuration file	yes
Values	Length of time, in seconds.

Related concepts

"Introducing the Lightweight Directory Access Protocol" on page 69
This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

"Using Secure Sockets Layer and the Lightweight Directory Access Protocol module" on page 72
IBM HTTP Server provides the ability to use a secure connection between the

LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

Ldap.transport directive

The ldap.transport directive indicates the transport method used to communicate with the LDAP server.

Syntax	`ldap.transport = TCP`
Scope	Single instance per directory stanza
Default	TCP
Module	`mod_ibm_ldap`
Multiple instances in the configuration file	yes
Values	TCP or SSL

Related concepts

"Introducing the Lightweight Directory Access Protocol" on page 69
This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

"Using Secure Sockets Layer and the Lightweight Directory Access Protocol module" on page 72
IBM HTTP Server provides the ability to use a secure connection between the LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

Ldap.url directive

The ldap.url directive indicates the URL of the LDAP server to authenticate against.

Syntax	`ldap.url = ldap://<hostname:port>/<BaseDN>`
	
	where:
	• hostname: Represents the host name of the LDAP server.
	• port: Represents the optional port number on which the LDAP server listens. The default for TCP connections is 389. You must specify the port number if you use SSL.
	• BaseDN: Provides the root of the LDAP tree in which to perform the search for users.
	
	For example:
	`ldap.URL=ldap://<ldap.ibm.com:489/o=Ace Industry, c=US>`

Scope	Single instance per directory stanza
Default	None
Module	`mod_ibm_ldap`
Multiple instances in the configuration file	yes

Related concepts

"Introducing the Lightweight Directory Access Protocol" on page 69
This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

"Using Secure Sockets Layer and the Lightweight Directory Access Protocol module" on page 72
IBM HTTP Server provides the ability to use a secure connection between the LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

Ldap.idleConnection.timeout directive

The ldap.idleConnection.timeout directive caches connections to the LDAP server for performance.

Syntax	`ldap.idleConection.timeout = <secs>`
Scope	Single instance per directory stanza
Default	600
Module	`mod_ibm_ldap`
Multiple instances in the configuration file	yes
Values	Length of time, in seconds, before an idle LDAP server connection closes due to inactivity.

Related concepts

"Introducing the Lightweight Directory Access Protocol" on page 69
This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

"Using Secure Sockets Layer and the Lightweight Directory Access Protocol module" on page 72
IBM HTTP Server provides the ability to use a secure connection between the LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

Ldap.user.authType directive

The ldap.usr.authType directive indicates the method for authenticating the user requesting a Web server. Use this name as the user name when accessing an LDAP server.

Syntax	`ldap.user.authType = BasicIfNoCert`
Scope	Single instance per directory stanza
Default	Basic
Module	`mod_ibm_ldap`
Multiple instances in the configuration file	yes
Values	Basic, Cert, BasicIfNoCert

Related concepts

"Introducing the Lightweight Directory Access Protocol" on page 69
This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

"Using Secure Sockets Layer and the Lightweight Directory Access Protocol module" on page 72

IBM HTTP Server provides the ability to use a secure connection between the LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

Ldap.user.cert.filter directive

The ldap.usr.cert.filter directive indicates the filter used to convert the information in the client certificate passed over Secure Sockets Layer (SSL) to a search filter for and LDAP entry.

Syntax	`ldap.user.cert.filter= (&(objectclass=person)(cn=%v1))`
Scope	Single instance per directory stanza
Default	"(&(objectclass=person) (cn=%v1, ou=%v2, o=%v3,c=%v4))"
Module	`mod_ibm_ldap`
Multiple instances in the configuration file	yes
Values	An LDAP filter. See Querying the LDAP server using LDAP search filters.

Secure Socket Layer (SSL) certificates include the following fields, all of which you can convert to a search filter:

Certificate field	Variable
common name	%v1
organizational unit	%v2
organization	%v3
country	%v4
locality	%v5
state or country	%v6
serial number	%v7

When you generate the search filter, you can find the field values in the matching variable fields (%v1, %v2). The following table shows the conversion:

User certificate	Filter conversion
Certificate	cn=Road Runner, o=Acme Inc, c=US
Filter	(cn=%v1, o=%v3, c=%v4)
Resulting query	(cn=RoadRunner, o=Acme, Inc, c=US)

Related concepts

"Introducing the Lightweight Directory Access Protocol" on page 69
This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

"Using Secure Sockets Layer and the Lightweight Directory Access Protocol module" on page 72
IBM HTTP Server provides the ability to use a secure connection between the LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

Ldap.user.name.fieldSep directive

The ldap.usr.name.fieldSep directive indicates characters as valid field separator characters when parsing the user name into fields.

Syntax	`ldap.user.name.fieldSep=/`
Scope	Single instance per directory stanza
Default	The space, comma, and the tab (/t) character.
Module	`mod_ibm_ldap`
Multiple instances in the configuration file	yes
Values	Characters. If '/' represents the only field separator character and the user enters "Joe Smith/Acme", then '%v2' equals "Acme".

Related concepts

"Introducing the Lightweight Directory Access Protocol" on page 69
This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

"Using Secure Sockets Layer and the Lightweight Directory Access Protocol module" on page 72
IBM HTTP Server provides the ability to use a secure connection between the LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

Ldap.user.name.filter directive

The ldap.usr.name.filter directive indicates the filter used to convert the user name entered in a search filter for an LDAP entry.

Syntax	`ldap.user.name.filter=<`*`user name filter`*`>`
Scope	Single instance per directory stanza
Default	`"((objectclass=person) (cn=%v1 %v2))"`, where *%v1* and *%v2* represent characters entered by the user. For example, if the user enters `"Paul Kelsey"`, the resulting search filter becomes `"((objectclass=person)(cn=Paul Kelsey))"`. You can find search filter syntax described in Querying the LDAP server using LDAP search filters. However, because the Web server cannot differentiate between multiple returned entries, authentication fails when the LDAP server returns more than one entry. For example, if the user makes the ldap.user.name.filter= `"((objectclass=person)(cn=%v1* %v2*))"` and enters **Pa Kel**, the resulting search filter becomes `"(cn=Pa* Kel*)"`. The filter finds multiple entries such as (cn=Paul Kelsey) and (cn=Paula Kelly) and authentication fails. You must modify your search filter.
Module	`mod_ibm_ldap`
Multiple instances in the configuration file	yes
Values	An LDAP filter. See Querying the LDAP server using LDAP search filters.

Related concepts

"Introducing the Lightweight Directory Access Protocol" on page 69
This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

"Using Secure Sockets Layer and the Lightweight Directory Access Protocol module" on page 72
IBM HTTP Server provides the ability to use a secure connection between the LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

Ldap.version directive

The ldap.version directive indicates the version of the LDAP protocol used to connect to the LDAP server. the protocol version used by the LDAP server determines the LDAP version. This directive is optional.

Syntax	`ldap.version=3`
Scope	Single instance per directory stanza
Default	`ldap.version=3`
Module	`mod_ibm_ldap`
Multiple instances in the configuration file	yes
Values	2 or 3

Related concepts

"Introducing the Lightweight Directory Access Protocol" on page 69
This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

"Using Secure Sockets Layer and the Lightweight Directory Access Protocol module" on page 72
IBM HTTP Server provides the ability to use a secure connection between the LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

Ldap.waitToRetryConnection.interval directive

The ldap.waitToRetryConnection.interval directive indicates the time the Web server waits between failed attempts to connect.

If an LDAP server goes down, the Web server continues to try to connect.

Syntax	`ldap.waitToRetryConnection.interval=<`*`secs`*`>`
Scope	Single instance per directory stanza
Default	`300`
Module	`mod_ibm_ldap`
Multiple instances in the configuration file	yes
Values	Time (in seconds)

Related concepts

"Introducing the Lightweight Directory Access Protocol" on page 69
This section addresses questions about what Lightweight Directory Access Protocol (LDAP) is and how it works, and provides high level overviews of X.500 and LDAP.

"Using Secure Sockets Layer and the Lightweight Directory Access Protocol module" on page 72
IBM HTTP Server provides the ability to use a secure connection between the LDAP module running in the Web server and the LDAP directory server. If this feature is enabled, any communication between the Web server and the directory server is encrypted.

Fast Response Cache Accelerator

This section describes topics on how to set up and use Fast Response Cache Accelerator (FRCA) with IBM HTTP Server.

Fast Response Cache Accelerator concepts and tasks

This section describes the key aspects of the Fast Response Cache Accelerator, how it can be used improve HTTP server performance, how it can be monitored, and what its restrictions are.

Setting up the Fast Response Cache Accelerator

This section describes how to set up the Fast Response Cache Accelerator for IBM HTTP Server.

Controlling cached content:

The cache accelerator can improve the performance of the IBM HTTP Server when serving static content, such as text and image files.

When the cache accelerator is enabled, the default configuration setting allows all static files to be cached. The cache automatically loads during server operation so that individual files do not need to be listed. Use the AfpaCache directive to turn caching on or off for specific directories.

The cache accelerator will remove files from the cache when they change to avoid serving stale content.

Enabling the cache accelerator:

By default, the cache accelerator is not enabled.

To enable the cache accelerator, edit the `httpd.conf` configuration file and delete the comment character (#) from the beginning of the LoadModule directive as follows:

```
#LoadModule ibm_afpa_module modules/mod_afpa_cache.so
```

becomes

```
LoadModule ibm_afpa_module modules/mod_afpa_cache.so
```

Customizing cache accelerator logging:

By default, the cache accelerator generates an access log of all requests that are served out of the cache. In order to minimize the effect of logging on performance, this is a separate file from the normal Apache access log.

Enable the cache accelerator access log if you want to maintain a record of requests served by the cache accelerator. Requests that are not served out of the cache will

be logged in the cache accelerator access log file. The cache accelerator access log file provides a useful way to verify that caching is enabled and to identify cached files.

Note: Even though a particular file might be cached, it might not always be served from the cache. Therefore, not every request for a cached file will result in a cache accelerator access log entry.

If you do not need access logging, turn the logging off for better performance. To set cache accelerator logging off, edit the `httpd.conf` configuration file. On AIX platforms, configure AfpaLogging Off. On Windows operating systems, insert a comment character (#) at the beginning of the AfpaLogFile line. For example:

```
#AfpaLogFile "_path_to_server_/logs/afpalog" V-ECLF
```

On Windows operating systems, the log file has a date stamp that automatically appends to its name. Everyday at midnight the server closes the current access log and creates a new one. This action enables the log file to process without having to stop and restart the server. Under heavy load conditions the log file can grow rapidly. Provide sufficient space on the hard drive for storage.

For each request that is served out of the cache accelerator, a log entry in the access log displays the following:

- Source host address
- Date and time of the request
- HTTP method of the request and what is requested
- HTTP return code, which indicates whether the request is honored
- Size of the returned data

A log entry can also optionally display the following:

- Target virtual host (use the formatting option V-CLF or V-ECLF)
- HTTP referer (use the formatting option V-CLF or V-ECLF)
- HTTP user agent (use the formatting option V-CLF or V-ECLF)

Enabling caching of servlets and JavaServer Pages files:

You can use the cache accelerator with WebSphere Application Server to cache certain dynamically-generated servlet and JavaServer Pages (JSP) files. This feature is only available on Windows versions of IBM HTTP Server.

In order to enable IBM HTTP Server for dynamic page caching, enable cache accelerator. In addition, the `afpaplugin_20.dll` component compatible with IBM HTTP Server V6.0 must be configured by WebSphere Application Server.

For details on how to enable this capability, see the external caching description in the WebSphere Application Server Information Center documentation.

Identifying caching and operational restrictions

This section describes the caching and operational restrictions for the Fast Response Cache Accelerator.

Restrictions on cached content:

This section discusses the caching restrictions for the Fast Response Cache Accelerator.

Caching does not occur on the following page types:

- Default welcome pages
- Requests ending in "/"
- Access-protected documents and pages requested over Secure Sockets Layer (SSL)

Caching limitations exist for the following situations:

- The cache accelerator supports only limited multilanguage content negotiation. Caching occurs for only a single language version, where a given URL maps to multiple translated versions.
- The cache accelerator must not be used with locally-mounted network file systems, such as Network File System (NFS) or Windows shared drives.
- The cache accelerator does not cache proxied content.

Cache accelerator operational restrictions:

This section discusses the operational restrictions for the Fast Response Cache Accelerator.

The following operational restrictions apply:

- When the cache accelerator is enabled, the default value of 0 for the MaxRequestsPerChild directive should be used, because graceful server restart is not supported with the cache accelerator.
- Currently, cache accelerator does not support IPv6 or Windows 64 bit operating systems.
- Cache accelerator may not be used when certain antivirus software is enabled. Currently Norton Antivirus has been identified as one such program.
- Cache accelerator access log entries are not integrated with the Apache access log.
- Only access logging facilities exist for monitoring the cache accelerator.
- On a given machine, only one instance of the IBM HTTP Server can have the cache accelerator enabled.
- Do not install the IBM HTTP Server on a machine running the IBM Netfinity Web Server Accelerator.

AIX considerations for Fast Response Cache Accelerator

The following items must be considered when you using Fast Response Cache Accelerator (FRCA) on AIX platforms:

- On AIX platforms, the FRCA kernel extension must load before starting IBM HTTP server with cache accelerator enabled. To do this, issue the **frcactrl load** command. This is normally configured to run whenever the system boots and before IBM HTTP Server starts. See the AIX man pages for more details about the **frcactrl** command.
- In order to place an upper bound on the percentage of CPU time that the FRCA kernel extension can spend in its interrupt (high priority) context, use the **frcactrl pctonintr** command. Increasing this above the default value of 80% is not recommended in order to allow other applications a reasonable amount of time to execute. Decrease this value if more time needs to be allocated to other applications, but note that reducing the value will result in more cache misses, even if a file is in the cache.
- You can find a sample AIX configuration in the `httpd.conf.sample.afpa` file, which is located in the `conf` directory of the IBM HTTP Server installation.

AFPA directives

This section describes the AFPA directives. The Fast Response Cache Accelerator utilizes a special high-performance component, based on the IBM Advanced Fast Path Architecture, from which the AFPA prefix is derived.

AfpaBindLogger directive

AfpaBindLogger allows you to bind the Fast Response Cache Accelerator logging thread in the kernel to a specific processor.

The format of the command is AfpaBindLogger [-1, 0, 1, ..., n], where -1 leaves the logging thread unbound and a number from 0 to total number of processors on the system, binds the logging thread to that processor.

Syntax	AfpaBindLogger [-1,0,1,..,n]
Scope	One per physical Apache server
Default	(-1)
Notes	Valid on AIX operating systems only.

AfpaCache directive

The AfpaCache directive turns the Fast Response Cache Accelerator on or off for a particular scope (such as a directory).

This directive applies to all descendants in a scope, unless otherwise modified by another directive.

Description	Turns static file caching on or off in the current scope.
Scope	Server configuration, virtual host, directory
Syntax	On or off
Usage	AfpaCache on
Override	Options
Multiple instances in the configuration file	Allowed
Notes	Valid on Windows and AIX operating systems.

AfpaDynacacheMax directive

The AfpaDynacacheMax directive is used on Windows to control the total amount of memory that can be utilized for caching of servlets and JavaServer Pages files.

When static files are cached, there is very little overhead for each entry since the file itself does not take up space in the cache, just the file handle. However, for servlets and JavaServer Pages files, the body of the response is stored in physical memory, so care must be taken to avoid consuming all available memory. Without this directive, cache accelerator will automatically set the upper bound to be approximately one eighth of physical memory. Use the directive to override that default.

Syntax	AfpaDynacacheMax size (Megabytes)
Scope	One directive per server
Notes	Valid on Windows operating systems

AfpaEnable directive

The AfpaEnable directive enables the Fast Response Cache Accelerator (cache accelerator).

If this directive is present and mod_afpa_cache.so is loaded, the cache accelerator listens on the port specified by the AfpaPort directive.

Syntax	AfpaEnable
Description	Enables fast response cache accelerator
Scope	One per physical Apache server
Notes	Valid on AIX and Windows operating systems.

AfpaLogFile directive

The AfpaLogFile directive defines the cache accelerator log file name, location, and logging format.

Scope	One per physical Apache server
Syntax	AfpaLogFile *log_file_name* [CLF | ECLF | V-CLF | V-ECLF| BINARY]
Notes	Valid on AIX and Windows operating systems. On Windows operating systems, the current date is used as the filetype for the log file and the log file is automatically rolled over at midnight each day.

The log formats are as follows:

- CLF = Common Log Format
- ECLF = Extended Common Log Format
- V-CLF = Common Log Format with virtual host information
- V-ECLF = Extended Common Log Format with virtual host information
- BINARY = Binary log with virtual host information (AIX only)

AfpaLogging directive

The AfpaLogging directive turns the fast response cache accelerator logging on or off.

Scope	One per physical Apache server
Syntax	AfpaLogging On | Off
Notes	Valid only on AIX operating systems.

AfpaMaxCache directive

The AfpaMaxCache directive specifies the maximum file size inserted into the fast response cache accelerator cache.

Syntax	AfpaMaxCache [size (bytes)]
Scope	One per physical Apache server
Default	none
Notes	Valid only on AIX operating systems.

AfpaMinCache directive

The AfpaMinCache directive specifies the minimum file size inserted into the Fast Response Cache Accelerator cache.

Syntax	AfpaMinCache [size]
Scope	One per physical Apache server
Default	none

Notes	Valid only on AIX operating systems.

AfpaPort directive

The AfpaPort directive tells the cache accelerator on which TCP port to listen.

The AfpaPort directive issues a listen for all TCP network adapters that are active on the server machine. The listen is effective for all TCP addresses.

Syntax	AfpaPort *port number*
Scope	One directive per server
Notes	Valid on AIX and Windows operating systems

AfpaRevalidationTimeout directive

AfpaRevalidationTimeout sets the time interval for revalidation of a cached object.

Once the RevalidationTimeout has been exceeded for a cached object, a fresh copy will be cached.

Syntax	AfpaRevalidationTimeout [value]
Scope	Global
Default	60 seconds
Notes	Valid on AIX operating systems only.

AfpaSendServerHeader directive

The AfpaSendServerHeader directive specifies whether or not the fast response cache accelerator sends the HTTP Server header in the response.

Syntax	AfpaSendServerHeader true or false
Scope	One per physical Apache server
Default	true
Notes	Valid only on AIX operating systems.

FastCGI

This section describes topics on using the Fast Common Gateway Interface protocol (FastCGI) with IBM HTTP Server.

FastCGI concepts and tasks

This section provides an information overview of the Fast Common Gateway Interface (FastCGI) protocol, FastCGI applications and the FastCGI Web site.

Learning about FastCGI

FastCGI is an interface between Web servers and applications which combines some of the performance characteristics of native Web server modules with the Web server indepedence of the Common Gateway Interface (CGI) programming interface. FastCGI, a language independent, scalable, open extension to CGI, provides high performance and persistence without the limitations of server-specific APIs. The FastCGI interface is described at http://www.fastcgi.com/.

IBM HTTP Server provides FastCGI support with the mod_fastcgi module. The mod_fastcgi module implements the capability for IBM HTTP Server to manage FastCGI applications and to allow them to process requests.

A FastCGI application typically uses a programming library such as the FastCGI development kit from http://www.fastcgi.com/. IBM HTTP Server does not provide a FastCGI programming library for use by FastCGI applications.

FastCGI applications are not limited to a particular development language. FastCGI application libraries currently exist for Perl, C/C++, Java, Python and the transmission control layer (TCL).

Example of mod_fastcgi configuration

The following directive is required to load mod_fastcgi into the server:

```
LoadModule fastcgi_module modules/mod_fastcgi.so
```

Other directives are then necessary to indicate which requests should be sent to FastCGI applications.

A complete configuration example for Windows operating systems. In this example, the `c:/Program Files/IBM HTTP Server 6.0/fcgi-bin/` directory contains FastCGI `echo.exe` applications. Requests from Web browsers for the`/fcgi-bin/echo.exe` URI will be handled by the FastCGI `echo.exe` application :

```
LoadModule fastcgi_module modules/mod_fastcgi.so
<IfModule mod_fastcgi.c>
       AllowOverride None
       Options +ExecCGI
       SetHandler fastcgi-script
</Directory>

FastCGIServer "C:/Program Files/IBM HTTP Server 6.0/fcgi-bin/echo.exe" -processes 1

</IfModule>
```

A complete configuration example for UNIX and Linux platforms. In this example, the `/opt/IBMIHS/fcgi-bin/` directory contains FastCGI applications, including the `echo.exe` application. Requests from Web browsers for the `/fcgi-bin/echo` URI will be handled by the FastCGI `echo.exe` application :

```
LoadModule fastcgi_module modules/mod_fastcgi.so
<IfModule mod_fastcgi.c>
ScriptAlias /fcgi-bin/ "/opt/IBMIHS/fcgi-bin/"

<Directory "/opt/IBMIHS/fcgi-bin/"
    AllowOverride None
       Options +ExecCGI
       SetHandler fastcgi-script
</Directory>

FastCGIServer "/opt/IBMIHS/fcgi-bin/echo" -processes 1
</IfModule>
```

Using FastCGI applications

FastCGI applications use TCP or UNIX sockets to communicate with the Web server. This scalable architecture enables applications to run on the same platform as the Web server, or on many machines scattered across an enterprise network.

You can port FastCGI applications to other Web server platforms. Most popular Web servers support FastCGI directly, or through commercial extensions.

FastCGI applications run fast because of their persistency. These applications require no per-request startup and initialization overhead. This persistency enables the development of applications, otherwise impractical within the CGI paradigm, like a huge Perl script, or an application requiring a connection to one or more databases.

Sending mail to the FastCGI Web site

For more information on FastCGI, visit the FastCGI Web site. To receive FastCGI related announcements and notifications of module updates, send mail to fastcgi-announce-request@idle.com with subscribe in the Subject field. To participate in the discussion of mod_fastcgi and FastCGI application development, send mail to fastcgi-developers-request@idle.com with subscribe in the Subject field.

The IBM HTTP Server Fast CGI plug-in provides an alternative method of producing dynamic content.

FastCGI directives

This section describes the FastCGI directives.

FastCGIAccessChecker directive

The FastCGIAccessChecker directive defines a FastCGI application as a per-directory access validator.

Syntax	`FastCGIAccessChecker` *`file name`* `[-compat]`
Scope	directory, location
Default	Directory
Module	mod_fastcgi
Multiple instances in the configuration file	yes
Values	File name

The Apache Access phase precedes user authentication and the HTTP headers submitted with the request determine the decision to enable access to the requested resource. Use FastCGI-based authorizers when a dynamic component exists as part of the access validation decision, like the time, or the status of a domain account.

If the FastCGI application file name does not have a corresponding static or external server definition, the application starts as a dynamic FastCGI application. If the file name does not begin with a slash (/), then the application assumes that the file name is relative to the ServerRoot.

Use the FastCgiAccessChecker directive within Directory or Location containers. For example:

```
<Directory htdocs/protected>
FastCgiAccessChecker fcgi-bin/access-checker
</Directory>
```

Mod_fastcgi sends nearly all of the standard environment variables typically available to CGI and FastCGI request handlers. All headers returned by a FastCGI access-checker application in a successful response (Status: 200), pass to subprocesses, or CGI and FastCGI invocations, as environment variables. All headers returned in an unsuccessful response pass to the client. Obtain FastCGI specification compliant behavior by using the -compat option.

Mod_fastcgi sets the environment variable FCGI_APACHE_ROLE to ACCESS_CHECKER, to indicate the Apache-specific authorizer phase performed.

The HTTP Server does not support custom failure responses from FastCGI authorizer applications. See the ErrorDocument directive for a workaround. A FastCGI application can serve the document.

FastCGIAccessCheckerAuthoritatve directive

The FastCGIAccessCheckerAuthoritatve directive enables access checking passing to lower level modules.

Syntax	`FastCGIAccessCheckerAuthoritative On | Off`
Scope	directory, location
Default	`FastCGIAccessCheckerAuthoritative On`
Module	mod_fastcgi
Multiple instances in the configuration file	yes
Values	On or off

Setting the FastCgiAccessCheckerAuthoritative directive explicitly to Off, enables access checking passing to lower level modules, as defined in the Configuration and `modules.c` files, if the FastCGI application fails to enable access.

By default, control does not pass on and a failed access check results in a forbidden reply. Consider the implications carefully before disabling the default.

FastCGIAuthenticator directive

The FastCGIAuthenticator directive defines a FastCGI application as a per-directory authenticator.

Syntax	`FastCGIAuthenticator file name [-compat]`
Scope	directory
Default	None
Module	mod_fastcgi
Multiple instances in the configuration file	yes
Values	File name

Authenticators verify the requester by matching the user name and password that is provided against a list or database of known users and passwords. Use FastCGI-based authenticators when the user database is maintained within an existing independent program, or resides on a machine other than the Web server.

If the FastCGI application file name does not have a corresponding static or external server definition, the application starts as a dynamic FastCGI application. If the file name does not begin with a slash (/), then the file name is assumed to be relative to the ServerRoot.

Use the FastCgiAuthenticator directive within directory or location containers, along with an AuthType and AuthName directive. This directive only supports the basic user authentication type. This authentication type needs a require, or FastCgiAuthorizer directive, to work correctly.

```
/Directory htdocs/protected>
AuthType Basic
AuthName ProtectedRealm
FastCgiAuthenticator fcgi-bin/authenticator
require valid-user
</Directory>
```

The Mod_fastcgi directive sends nearly all of the standard environment variables that are typically available to CGI and FastCGI request handlers. All headers returned by a FastCGI authentication application in a successful response (Status: 200) pass to subprocesses, or CGI and FastCGI invocations, as environment variables. All headers returned in an unsuccessful response are passed to the client. Obtain FastCGI specification compliant behavior by using the `-compat` option.

The Mod_fastcgi directive sets the `FCGI_APACHE_ROLE` environment variable to `AUTHENTICATOR`, indicating the Apache-specific authorizer phase performed.

This directive does not support custom failure responses from FastCGI authorizer applications. See the ErrorDocument directive for a workaround. A FastCGI application can serve the document.

FastCGIAuthenticatorAuthoritative directive

The FastCGIAuthenticatorAuthoritative directive enables authentication passing to lower level modules defined in the configuration and `modules.c` files, if explicitly set to off and the FastCGI application fails to authenticate the user.

Syntax	`FastCGIAuthenticatorAuthoritative` *`On`* | *`Off`*
Scope	directory
Default	`FastCgiAuthenticatorAuthoritative On`
Module	mod_fastcgi
Multiple instances in the configuration file	yes
Values	On or off

Use this directive in conjunction with a well protected AuthUserFile directive, containing a few administration-related users.

By default, control does not pass on and an unknown user results in an Authorization Required reply. Consider implications carefully before disabling the default.

FastCGIAuthorizer directive

Defines a FastCGI application as a per-directory authorizer.

Syntax	`FastCgiAuthorizer file name` *`[-compat]`*
Scope	directory
Default	None
Module	mod_fastcgi
Multiple instances in the configuration file	yes
Values	File name

Authorizers validate whether an authenticated user can access a requested resource. Use FastCGI-based authorizers when a dynamic component exists as part of the authorization decision, such as the time, or currency of the user's bills.

If the FastCGI application file name does not have a corresponding static or external server definition, the application starts as a dynamic FastCGI application. If the file name does not begin with a slash (/) then the file name is assumed relative to the ServerRoot.

Use FastCgiAuthorizer within Directory or Location containers. Include an AuthType and AuthName directive. This directive requires an authentication directive, such as FastCgiAuthenticator, AuthUserFile, AuthDBUserFile, or AuthDBMUserFile to work correctly.

```
<Directory htdocs/protected>
AuthType Basic
AuthName ProtectedRealm
AuthDBMUserFile conf/authentication-database
FastCgiAuthorizer fcgi-bin/authorizer
</Directory>
```

The Mod_fastcgi directive sends nearly all of the standard environment variables typically available to CGI and FastCGI request handlers. All headers returned by a FastCGI authentication application in a successful response (Status: 200) pass to subprocesses, or CGI and FastCGI invocations, as environment variables. All headers returned in an unsuccessful response pass on to the client. Obtain FastCGI specification compliant behavior by using the -compat option.

The Mod_fastcgi directive sets the environment variable FCGI_APACHE_ROLE to AUTHORIZER, to indicate the Apache-specific authorizer phase performed.

This directive does not support custom failure responses from FastCGI authorizer applications. See the ErrorDocument directive for a workaround. A FastCGI application can serve the document.

FastCGIAuthorizerAuthoritative directive

Enables authentication passing to lower level modules, as defined in the configuration and `modules.c` files, when explicitly set to Off, if the FastCGI application fails to authenticate the user.

Syntax	`FastCgiAuthorizerAuthoritative file name` *`On`* | *`Off`*
Scope	directory
Default	`FastCgiAuthorizerAuthoritative file name On`
Module	mod_fastcgi
Multiple instances in the configuration file	yes
Values	On or off

Use this directive in conjunction with a well protected AuthUserFile containing a few administration-related users.

By default, control does not pass on and an unknown user results in an Authorization Required reply. Consider the implications carefully before disabling the default.

FastCGIConfig directive

Defines the default parameters for all dynamic FastCGI applications.

Syntax	`FastCgiConfig option option...` The FastCgiConfig directive does not affect static or external applications.
Scope	directory
Default	None
Module	mod_fastcgi
Multiple instances in the configuration file	yes
Values	Dynamic applications start upon demand. Additional application instances start to accommodate heavy demand. As demand fades, the number of application instances decline. Many of the options govern this process.

Option can include one of the following (case insensitive):

- **appConnTimeout n (0 seconds).** The number of seconds to wait for a connection to the FastCGI application to complete or 0, to indicate use of a blocking connect(). If the timeout expires, a SERVER_ERROR results. For non-zero values, this amount of time used in a select() to write to the file descriptor returned by a non-blocking connect(). Non-blocking connect()s are troublesome on many platforms. See also -idle-timeout; this option produces similar results, but in a more portable manner.
- **idle-timeout n (30 seconds).** The number of seconds of FastCGI application inactivity allowed before the request aborts and the event logs at the error LogLevel. The inactivity timer applies only when a pending connection with the FastCGI application exists. If an application does not respond to a queued request within this period, the request aborts. If communication completes with the application, but not with the client (a buffered response), the timeout does not apply.
- **autoUpdate none.** This option causes the mod_fastcgi module to check the age of the application on disk before processing each request. For recent applications, this function notifies the process manager and stops all running instances of the application. Build this type of functionality into the application. A problem can occur when using this option with -restart.
- **gainValue n (0.5).** A floating point value between 0 and 1 that is used as an exponent in the computation of the exponentially decayed connection times load factor of the currently running dynamic FastCGI applications. Old values are scaled by (1 - gainValue), so making values smaller, weights them more heavily compared to the current value, which is scaled by gainValue.
- **initial-env name[=value] none.** A name-value pair passed in the initial environment when instances of the application spawn. To pass a variable from the Apache environment, do not provide the ″=″ (if the variable is not actually in the environment, it is defined without a value). To define a variable without a value, provide the ″=″ without any value. This option is repeatable.
- **init-start-delay n (1 second).** The minimum number of seconds between the spawning of instances of this application. This delay decreases the demand placed on the system at server initialization.
- **killInterval n (300 seconds).** The killInterval determines how often the dynamic application instance killing policy is implemented within the process manager. Lower numbers result in a more aggressive policy, while higher numbers result in a less aggressive policy.
- **listen-queue-depth n (100).** The depth of the listen() queue, also known as the backlog, shared by all instances of this application. A deeper listen queue allows

the server to cope with transient load fluctuations without rejecting requests; it does not increase throughput. Adding additional application instances can increase throughput and performance, depending upon the application and the host.

- **maxClassProcesses n (10).** The maximum number of dynamic FastCGI application instances allowed to run for any one FastCGI application.
- **maxProcesses n (50).** The maximum number of dynamic FastCGI application instances allowed to run at any time.
- **minProcesses n (5).** The minimum number of dynamic FastCGI application instances the process manager allows to run at any time, without killing them due to lack of demand.
- **multiThreshhold n (50).** An integer between 0 and 100 used to determine whether to terminate any instance of a FastCGI application. If the application has more than one instance currently running, this attribute helps to decide whether to terminate one of them. If only one instance remains, singleThreshhold is used instead.
- **pass-header header none.** The name of an HTTP Request Header passed in the request environment. This option makes the contents of headers available to a CGI environment.
- **priority n (0).** The process priority assigned to the application instances using setpriority().
- **processSlack n (5 seconds).** If the sum of all currently running dynamic FastCGI applications exceeds maxProcesses - processSlack, the process manager invokes the killing policy. This action improves performance at higher loads, by killing some of the most inactive application instances before reaching the maxProcesses value.
- **restart none.** This option causes the process manager to restart dynamic applications upon failure, similar to static applications.
- **Restart-delay n (5 seconds).** The minimum number of seconds between the respawning of failed instances of this application. This delay prevents a broken application from soaking up too much of the system.
- **singleThreshhold n (0).** An integer between 0 and 100, used to determine whether the last instance of a FastCGI application can terminate. If the process manager computed load factor for the application is lower than the specified threshold, the last instance is terminated. Specify a value closer to 1, to make your executables run in the idle mode for a long time. If memory or CPU time is a concern, a value closer to 100 is more applicable. A value of 0, prevents the last instance of an application from terminating; this value is the default. Changing this default is not recommended, especially if you set the -appConnTimeout option.
- **startDelay n (3 seconds).** The number of seconds the Web server waits while trying to connect to a dynamic FastCGI application. If the interval expires, the process manager is notified with hope that another instance of the application starts. Set the startDelay value smaller than the appConnTimeout value, to be effective.
- **updateInterval n (300 seconds).** The updateInterval decides how often statistical analysis is performed to determine the fate of dynamic FastCGI applications.

FastCGIExternalServer directive

The FastCGIExternalServer defines file name as an external FastCGI application.

It operates the same as the Fastcgiserver directive, except that the CGI application is running in another process outside of the Web server.

Syntax	`FastCgiExternalServer` *`file name -host hostnameport [-appConnTimeout n]`* `FastCgiExternalServer` *`file name -socket file name [-appConnTimeout n]`*
Scope	Server configuration
Default	None
Module	mod_fastcgi
Multiple instances in the configuration file	yes

Values

- **appConnTimeout *n* (0 seconds).** The number of seconds to wait for a connection to the FastCGI application to complete, or 0, to indicate use of a blocking `connect()` method. If the timeout expires, a SERVER_ERROR results. For non-zero values, this indicator is the amount of time used in a `select()` method to write to the file descriptor returned by a non-blocking `connect()` method. Non-blocking `connect()` methods are troublesome on many platforms. See also -idle-timeout; this option produces similar results, but in a more portable manner.
- **Idle-timeout n (30 seconds).** The number of seconds of FastCGI application inactivity allowed before the request aborts and the event is logged (at the `error LogLevel`). The inactivity timer applies only as long as a connection is pending with the FastCGI application. If a request is queued to an application, but the application does not respond by writing and flushing within this period, the request aborts. If communication is complete with the application but incomplete with the client (a buffered response), the timeout does not apply.
- **flush none.** Force a write to the client as data is received from the application. By default, the `mod_fastcgi` option buffers data to free the application quickly.
- **host hostname:port none.** The hostname, or IP address and TCP port number (1-65535) the application uses for communication with the Web server. The `-socket` and `-host` options are mutually exclusive.
- **Pass-header header none.** The name of an HTTP Request Header passed in the request environment. This option makes the header contents available, to a CGI environment.
- **socket file name none.**
  - **On UNIX operating systems.** The file name of the UNIX domain socket the application uses for communication with the Web server. The file name is relative to the FastCgiIpcDir option. The `-socket` and `-port` options are mutually exclusive.
  - **On Windows operating systems.** The name of the pipe the application uses for communicating with the Web server. The name is relative to the FastCgiIpcDir option. The -socket and -port options are mutually exclusive.

FastCGIIpcDir directive

The FastCGIIpcDir directive specifies directory as the place to store the UNIX socket files used for communication between the applications and the Web server.

Syntax	• On UNIX platforms - FastCgiIpcDir *directory* • On Windows operating systems - FastCgiIpcDir *name*
Scope	Server configuration
Default	None
Module	mod_fastcgi
Multiple instances in the configuration file	yes
Values	directory or name

On UNIX platforms. The FastCgiIpcDir directive specifies directory as the place to store and find, in the case of external FastCGI applications, the UNIX socket files that are used for communication between the applications and the Web server. If the directory does not begin with a slash (/) then it is assumed to be relative to the ServerRoot. If the directory does not exist, the function attempts to create the directive with appropriate permissions. Specify a directory on a local file system. If you use the default directory, or another directory within `/tmp`, `mod_fastcgi` breaks if your system periodically deletes files from the `/tmp` directory.

On Windows operating systems. The `FastCgiIpcDir` directive specifies *name* as the root for the named pipes used for communication between the application and the Web server. Define the name in the form >\\.*pipe**pipename.* . The pipename syntax can contain any character other than a backslash.

The `FastCgiIpcDir` directive must precede any `FastCgiServer` or `FastCgiExternalServer` directives, which make use of UNIX sockets. Ensure a readable, writeable, and executable directory by the Web server. No one should have access to this directory.

FastCGIServer directive

The FastCGIServer directive defines file name as a static FastCGI application.

The Process Manager starts one instance of the application with the default configuration specified in parentheses below. Should a static application instance die for any reason, the mod_fastcgi module spawns another instance for replacement and logs the event at the warn LogLevel.

Syntax	`FastCgiServer file name` *`[options]`*
Scope	Server configuration
Default	None
Module	mod_fastcgi
Multiple instances in the configuration file	yes
Values	directory or name

You can use one of the following case-insensitive options:

- **appConnTimeout *n* (0 seconds).** The number of seconds to wait for a connection to the FastCGI application to complete, or 0, to indicate use of a blocking connect(). If the timeout expires, a SERVER_ERROR results. For non-zero values, this indicator is the amount of time used in a select() to write to the file

descriptor returned by a non-blocking connect(). Non-blocking connect()s prove troublesome on many platforms. See the -idle-timeout option; it produces similar results but in a more portable manner.

- **Idle-timeout *n* (30 seconds).** The number of seconds of FastCGI application inactivity allowed before the request aborts and the event logs at the error LogLevel. The inactivity timer applies only when a pending connection with the FastCGI application exists. If an application does not respond to a queued request within this period, the request aborts. If communication completes with the application, but does not complete with the client (a buffered response), the timeout does not apply.
- **initial-env name [=value] none]none.** A name-value pair passed in the FastCGI application initial environment. To pass a variable from the Apache environment, do not provide the ″=″ (variables not actually in the environment, are defined without a value). To define a variable without a value, provide the ″=″ without a value. You can repeat this option.
- **init-start-delay *n* (1 second).** The minimum number of seconds between the spawning of instances of this application. This delay decreases the demand placed on the system at server initialization.
- **Flush none.** Force a write to the client as data arrives from the application. By default, mod_fastcgi buffers data to free the application quickly.
- **Listen-queue-depth *n* (100).** The depth of the listen() queue, also known as the backlog, shared by all of the instances of this application. A deeper listen queue enables the server to cope with transient load fluctuations, without rejecting requests; this option does not increase throughput. Adding additional application instances can increase throughput and performance, depending upon the application and the host.
- **Pass-header header none.** The name of an HTTP Request Header passed in the request environment. This option makes the contents of headers available to a CGI environment.
- **processes *n* (1).** The number of application instances to spawn at server initialization.
- **Priority *n* (0).** The process priority assigned to the application instances, using setpriority().
- **port *n* none.** The TCP port number (1-65535) the application uses for communication with the Web server. This option makes the application accessible from other machines on the network. The -socket and -port options are mutually exclusive.
- **Restart-delay *n* (5 seconds).** The minimum number of seconds between the respawning of failed instances of this application. This delay prevents a broken application from using too many system resources.
- **Socket file name**:
  - On UNIX platforms: The file name of the UNIX domain socket that the application uses for communication with the Web server. The module creates the socket within the directory specified by FastCgiIpcDir. This option makes the application accessible to other applications, for example, cgi-fcgi on the same machine, or through an external FastCGI application definition, FastCgiExternalServer. If neither the -socket nor the -port options are given, the module generates a UNIX domain socket file name. The -socket and -port options are mutually exclusive.
  - On Windows operating systems: The name of the pipe for the application to use for communication with the Web server. The module creates the named pipe off the named pipe root specified by the FastCgiIpcDir directive. This option makes the application accessible to other applications, like cgi-fcgi on

the same machine or through an external FastCGI application definition, FastCgiExternalServer. If neither the -socket nor the -port options are given, the module generates a name for the named pipe. The -socket and -port options are mutually exclusive. If the file name does not begin with a slash (/), then this file name is assumed relative to the ServerRoot.

FastCGIsuEXEC directive

The FastCGIsuEXEC directive supports the suEXEC-wrapper.

Syntax	`FastCgiSuexec` *`On	Off file name`*
Scope	Server configuration
Default	`FastCgiSuexec Off`
Module	mod_fastcgi
Multiple instances in the configuration file	yes
Values	The FastCgiSuexec directive requires suEXEC enabling in Apache for CGI. To use the same suEXEC-wrapper used by Apache, set FastCgiSuexec to On. To use a different suEXEC-wrapper, specify the file name of the suEXEC-wrapper. If the file name does not begin with a slash (/), then the file name is assumed relative to the ServerRoot.

When you enable the FastCgiSuexec directive, the location of static or external FastCGI application definitions becomes important. These differences inherit their user and group from the User and Group directives in the virtual server in which they were defined. User and Group directives should precede FastCGI application definitions. This function does not limit the FastCGI application to the virtual server in which it was defined. The application can service requests from any virtual server with the same user and group. If a request is received for a FastCGI application, without an existing matching definition running with the correct user and group, a dynamic instance of the application starts with the correct user and group. This action can lead to multiple copies of the same application running with a different user and group. If this causes a problem, preclude navigation to the application from other virtual servers, or configure the virtual servers with the same user and group.

See the Apache documentation for more information about suEXEC and the security implications.

Using IBM HTTP Server with the WebSphere Application Server administrative console

This section describes how to administer IBM HTTP Server using the WebSphere Application Server administrative console.

Overview: IBM HTTP Server remote administration

IBM HTTP Server remote administration using WebSphere Application Server Network Deployment. You can administer and configure IBM HTTP Server V6.0 using the WebSphere Administrative Console. The IBM HTTP Server V6.0 installation includes the IBM administration server. When you install IBM HTTP Server on a machine without the WebSphere Application Server, the IBM administration server is necessary for administration. In order for the IBM administration server to handle requests for the administration of IBM HTTP Server, the IBM administration server must be started and defined to an

unmanaged WebSphere Application Server node. Administration of IBM HTTP Server is available without the IBM administration server if the IBM HTTP Server is installed on a machine with a WebSphere managed node.

You must define IBM HTTP Server through the WebSphere administrative console. Once defined, an administrator can administer and configure IBM HTTP Server through the WebSphere administrative console. Administration includes the ability to start and stop the IBM HTTP Server. You can display and edit the IBM HTTP Server configuration file, and you can view the IBM HTTP Server error and access logs. The plug-in configuration file can be generated for IBM HTTP Server and propagated to the remote or locally-installed IBM HTTP Server.

IBM HTTP Server remote administration using WebSphere Application Server Express and Base. Administration function for IBM HTTP Server with the WebSphere Application Server Express or Base product is limited to generation and propagation of the plug-in configuration file for IBM HTTP Server. IBM HTTP Server will always be defined to an unmanaged WebSphere Application Server Express or Base node.

Installing the IBM administration server

The IBM HTTP Server administration server installs by default during a typical IBM HTTP Server installation. During a custom installation you can select the administration module.

If you are planning on managing an IBM HTTP Server Web server on a managed node (through nodeagent), you must install IBM HTTP Server with *log on as system rights*.

Starting the IBM HTTP administration server on Windows operating systems

To start the IBM HTTP administration server, click **Start > Programs > IBM HTTP Server > Start Administration Server**.

A message box is displayed that indicates the server has started. To confirm that IBM HTTP administration server started successfully, check the `admin_error.log` file for a ″start successful″ message. If you use the developer installation option, then the IBM HTTP administration server does not install as a service. You have to run the `apache.exe` file from a command line with the `-f` option. On Windows operating systems, from the default directory, type:

```
apache -f  conf\admin.conf
```

If the IBM HTTP administration server does not start:

- Open the Control Panel.
- Click **Services**.
- Double-click IBM HTTP Administration Server to start the server.

To confirm that IBM HTTP Administration Server started successfully, check the `admin_error.log` file for a ″start successful″ message.

Starting the IBM HTTP administration server on Linux and UNIX platforms

The `adminctl` command is used to start and stop IBM HTTP administration server. You can find the `adminctl` command in the `bin` subdirectory, within the IBM HTTP Server installation directory. If that directory is not in your PATH, the full path should be given on the command line.

Starting and stopping IBM HTTP administration server. To start or stop IBM HTTP administration server using the default `admin.conf` configuration file, run the `adminctl start` command to start the server or run the `adminctl stop` command to stop the server.

Issue the commands from the default directories, based on your operating system:

- AIX: `/usr/IBMIHS/bin/adminctl start|stop`
- HP-UX: `/opt/IBMIHS/bin/adminctl start|stop`
- Linux: `/opt/IBMIHS/bin/adminctl start|stop`
- Solaris: `/opt/IBMIHS/bin/adminctl start|stop`

For example, The `adminctl` command is not in your PATH, the IBM HTTP Server installation directory is `/usr/IBMIHS`, and the default configuration file is used as follows:

```
# /usr/IBMIHS/bin/adminctl start
# /usr/IBMIHS/bin/adminctl stop
```

To confirm that IBM HTTP Server started successfully, check the `admin_access.log`.

Running the setupadm script

The setupadm script establishes permissions for configuration file updates.

You cannot update the configuration files after a default server installation, unless you run the setupadm script, or you set permissions manually.

The setupadm script prompts you for the following input:

- **User ID** - The user ID that you use to log on to the administration server. The script creates this user ID.
- **Group name** - The administration server accesses the configuration files and authentication files through group file permissions. The script creates the specified group through this script.
- **Directory** - The directory where you can find configuration files and authentication files.
- **File name** - The following file groups and file permissions change:
  - Single file name
  - File name with wildcard
  - All (default) - All of the files in the specific directory
  - Processing - The setupadm script changes the group and file permissions of the configuration files and authentication files.

The administration server requires read and write access to configuration files and authentication files to perform Web server configuration data administration. In addition to the Web server files, you must change the permissions to the targeted plug-in configuration files. See Setting permissions manually for instructions.

The administration server has to execute **adminctl restart** as root to perform successful restarts of the IBM HTTP Server.

Setting permissions manually

The setupadm script will create users and groups and set file permissions for them. This section describes how to do this manually.

Create a new user and unique group for the IBM HTTP Server administration server

Follow these steps, noted by platform:

- **On AIX platforms:**
  1. Launch SMIT.
  2. Click **Security and Users**.
  3. Click **Groups > Add a Group**.
  4. Enter the group name, for example, **admingrp**.
  5. Click **OK**. Go back to **Security and Users**.
  6. Click **Users > Add a User**.
  7. Enter the user name, for example, **adminuser**.
  8. Enter the primary group you just created.
  9. Click **OK**.
- **On HP_UX and Linux platforms:**
  - Run the following command from a command line:

    ```
    groupadd <group_name>
    useradd -g <group_name> <user_ID>
    ```
- **On Solaris platforms:**
  1. Launch the administration tool.
  2. Click **Browse > Groups**.
  3. Click **Edit > Add**.
  4. Enter the group name, for example, **admingrp**.
  5. Click **OK**.
  6. Click **Browse > Users**.
  7. Click **Edit > Add**.
  8. Enter the user name, for example, **adminuser** and the primary group name, for example, **admingrp**.
  9. Click **OK**.

Updating file permissions on UNIX and Linux platforms

Once you have created a user and group, set up file permissions as follows:

1. Update the permissions for the targeted IBM HTTP Server `conf` directory.
   a. At a command prompt, change to the directory where you installed IBM HTTP Server.
   b. Type the following commands:

      ```
      chgrp <group_name> <directory_name>
      chmod g+rw <directory_name>
      ```
2. Update the file permission for the targeted IBM HTTP Server configuration files.

a. At a command prompt, change to the directory that contains the configuration files.
b. Type the following commands:

```
chgrp <group_name> <file_name>
chmod g+rw <file_name>
```

3. Update the `admin.conf` configuration file for the IBM HTTP Server administration server.
   a. Change to the IBM HTTP Server administration server `admin.conf` directory.
   b. Search for the following lines in the `admin.conf` file:

```
User nobody
Group nobody
```

   c. Change those lines to reflect the user ID and unique group name you created. For example:

```
User userID
Group group_name
```

4. Update the file permission for the targeted plug-in configuration files.
   a. At a command prompt, change to the directory that contains the plug-in configuration files.
   b. Type the following commands:

```
chgrp <group_name> <file_name>
chmod g+rw <file_name>
```

You have set up read and write access for the configuration and authentication files. Now you can perform Web server configuration data administration.

Enabling access to the administration server using the htpasswd utility

The administration server is installed with authentication enabled. This means that the administration server will not accept a connection without a valid user ID and password. This is done to protect the IBM HTTP Server configuration file from unauthorized access.

Launch the **htpasswd** utility that is shipped with the administration server. This utility creates and updates the files used to store user names and password for basic authentication of users who access your Web server. Locate **htpasswd** in the `bin` directory.

- On Windows operating systems: `htpasswd -cm <install_dir>\conf\admin.passwd [login name]`
- On Linux and UNIX platforms: `./htpasswd -cm <install_dir>/conf/admin.passwd [login name]`

where `<install_dir>` is the IBM HTTP Server installation directory and `[login name]` is the user ID that you use to log into the administration server.

You will be prompted for a password two times. The password file is referenced in the `admin.conf` file with the AuthUserFile directive. Your users will now have to enter a password when accessing protected areas. For further information on authentication configuration, see the Apache Authentication, Authorization and Access Control documentation.

Password protection

This section describes topics on how to secure IBM HTTP Server.

Enabling access to the administration server using the htpasswd utility

The administration server is installed with authentication enabled. This means that the administration server will not accept a connection without a valid user ID and password. This is done to protect the IBM HTTP Server configuration file from unauthorized access.

Launch the **htpasswd** utility that is shipped with the administration server. This utility creates and updates the files used to store user names and password for basic authentication of users who access your Web server. Locate **htpasswd** in the `bin` directory.

- On Windows operating systems: `htpasswd -cm <install_dir>\conf\admin.passwd [login name]`
- On Linux and UNIX platforms: `./htpasswd -cm <install_dir>/conf/admin.passwd [login name]`

where `<install_dir>` is the IBM HTTP Server installation directory and `[login name]` is the user ID that you use to log into the administration server.

You will be prompted for a password two times. The password file is referenced in the `admin.conf` file with the AuthUserFile directive. Your users will now have to enter a password when accessing protected areas. For further information on authentication configuration, see the Apache Authentication, Authorization and Access Control documentation.

Protecting access to other Web server resources

This topic describes how to protect access to documents and applications using password protection.

If you want to require a valid user id and password before displaying a particular file, or a directory of files, set up password protection.

If you do not want any protection for your server, you can use the default settings for IBM HTTP Server. Without protection, any client can access documents in directories enabled in the configuration file. Setting up password protection will enable you to limit access to certain files.

For example, if your server serves stock quotes, you want to verify that the client requesting the quote has paid for the quotation service. Password protection limits access to certain files.

To set password protection see the documentation in the Apache User's Guide for the Apache Authentication, Authorization and Access Control documentation.

Chapter 6. Third-party modules

This section describes topics on using third-party modules with IBM HTTP Server.

Identifying viable compilers

Apache modules and third-party module testing incorporated the compilers and compiler levels that are included in the following list. Other compilers can work, but testing was limited to these environments:

- AIX - C for AIX V5.0.2.3 or VisualAge C++ Professional V5.0.2.3
- HP - HP_UXaC++ Compiler (A.03.xx)
- Linux for Intel - gcc 3.2.2
- Linux for PowerPC - gcc 3.2.2
- Linux for S/390 - gcc 3.2
- Solaris - SunWorkShop V5.0
- Windows - Microsoft Visual C++ 6.0

Locating build components for the UNIX and Linux platforms

You can find the key components for building dynamic shared objects (DSOs) and dynamic link libraries (DLLs) on the UNIX and Linux platforms as follows:

- Locate the header files in the `include` directory.
- Locate the apxs script in the `bin` directory.

Build method options for dynamic modules

There are several options for building dynamic modules:

- **Configuration Scripts**. Some Apache modules include configuration scripts with the module source. These configuration scripts make compiling and installing modules easy.

  Sometimes on Linux and UNIX platforms, these configuration scripts have a WITH_APXS option. If you use this option, verify that you point to the APXS script that installs with IBM HTTP Server in the `bin` directory.

  Check the configuration script parameters if you experience problems building or running your dynamically shared object (DSO) built module.
- **The example_module directory**. On Linux and UNIX platforms, IBM HTTP Server ships a sample module (`mod_example.c`) in the `example_module` directory. See Apache APXS for more information.
- **APXS Script** (Linux and UNIX platforms only):

  The installation process places APXS, a Perl script, in the IBM HTTP Server `bin` directory. This script builds dynamically shared objects on AIX, HP, Linux, and Solaris platforms.

  To use the APXS script, verify that you have Perl script V5.003 or later installed. Verify that the path to the Perl executable on the first line of the APXS script is correct, for example, `/usr/bin/local/perl`. See Apache APXS for information about using the APXS script on specific platforms. You might have to change this line to accommodate your Perl installation directory.

Considerations for building third-party modules

This section describes considerations for building third-party modules to run as dynamic shared object modules or dynamic link libraries.

If you already have an existing Apache or IBM HTTP Server installed in the default directory, `/usr/lib/apache` for Apache or `/usr/IBMIHS` for IBM HTTP Server, third-party module configuration scripts can look for and use parameters based on the existing Apache or IBM HTTP Server directory. To avoid this problem, perform one of the following actions:

- Delete your Apache directory.
- Verify that you use third-party configuration script options, `prefix=<installation root>`, where: `prefix` represents the installation root for the IBM HTTP Server.

Considerations for building dynamic modules on Windows operating systems

The following restrictions apply when building a module to run with the IBM HTTP Server:

- Link your dynamic module to the libraries that are contained in `lib` directory where the server is installed.
- The Apache HTTP Server module API is defined by the header files that are contained in the `include` directory where the server is installed. Your module should include any of these header files as needed.
- You must not modify any file or data structure that is contained in any file in the `include` directory where the server is installed.

Chapter 7. Troubleshooting

This section describes topics on how to perform problem determination on IBM HTTP Server.

Knowing what to do first

This section describes how to start troubleshooting IBM HTTP Server.

1. Ensure you run the IBM Developer Kit, Java Edition V1.4, or the Java Runtime Environment (JRE) V1.4 on AIX, Linux, HP, Solaris and Windows operating systems. You must install and run the IBM Developer Kit, Java edition, or the JRE yourself.
2. Check the error log to help you determine the type of problem. You can find the error logs in the directory specified by the ErrorLog directive in the configuration file. Depending on the operating system, the default directories are:
   - On AIX platforms: `/usr/IBMIHS/logs/error_log`
   - On HP platforms: `/opt/IBMIHS/logs/error_log`
   - On Linux platforms: `/opt/IBMIHS/logs/error_log`
   - On Solaris platforms: `/opt/IBMIHS/logs/error_log`
   - On Windows operating systems: `/<server_root>/logs/error_log`

Experiencing an IBM HTTP Server Service logon failure on Windows operating systems

When installing the IBM HTTP Server, prompts appear for a login ID and password. The ID you select must have the capability to log on as a service.

If you get an error when you try to start the IBM HTTP Server Service, indicating a failure to start as a service, try one of the following:

1. Click **Start > Programs > Administrative Tools > User Manager**.
2. Select the user from the User Manager list.
3. Click **Policies > User Rights**.
4. Select the **Show Advanced User Rights** check box.
5. Click **Log on as a Service**, from the right drop-down menu, *or*

1. Click **Start > Settings > Control Panel**.
2. Open Administrative Tools.
3. Open Services. The local user you select is created in Local Users and Groups, under Computer Management.
4. Click **Service > Actions > Properties**.
5. Choose the Log on tab.
6. Select this account option and click **Browse**, to select the user to associate with the service.

Symptoms of poor server response time

If you notice that server CPU utilization appears low, but client requests for static pages take a long time to service, your server may be running out of server threads to handle requests.

This situation results when you have more inbound requests than you have Apache threads to handle those requests. New connections queue in the TCP/IP stack listen queue wait for acceptance from an available thread. As a thread becomes available, it accepts and handles a connection off of the listen queue. Connections can take a long time to reach the top of the listen queue. This condition will be logged in a single error message in the error log:

- The message on AIX, Linux, Solaris, or HP-UX platforms is: ″Server reached MaxClients setting, consider raising the MaxClients setting″
- The message on Windows operating systems is: ″Server ran out of threads to serve requests. Consider raising the ThreadsPerChild setting″

If you configured Apache to listen on multiple ports, you can find responses slow on one port (port 80, for example) but adequate on another port (port 443, for example). The disparity in response time results from each port having its own listen queue. You may have a deep port 80 queue and a shallow port 443 queue. Apache does not attempt to balance the number of connections received from each queue. When a connection becomes available, Apache and the operating system consume from either queue at random. To address this scenario, add more Apache threads to handle inbound connections. This is accomplished by increasing the number of processes or increasing the number of threads per process.

Identifying error messages

This section describes error messages you may see when using IBM HTTP Server.

SSL certificate revocation list

This section provides information on identifying directives for certificate revocation list (CRL) and those supported in global servers and virtual hosts.

Certificate revocation provides the ability to revoke a client certificate given to the IHS server by the browser when the key becomes compromised or when access permission to the key gets revoked. CRL represents a database which contains a list of certificates revoked before their scheduled expiration date.

If you want to enable certificate revocation in the IBM HTTP Server, publish the CRL on a Lightweight Directory Access Protocol (LDAP) server. Once the CRL is published to an LDAP server, you can access the CRL using the IBM HTTP Server configuration file. The CRL determines the access permission status of the requested client certificate.

Identifying directives needed to set up a certificate revocation list. The SSLClientAuth directive can include two options at once:

- SSLClientAuth 2 crl
- SSLClientAuth1 crl

The CRL option turns CRL on and off inside an SSL virtual host. If you specify CRL as an option, then you elect to turn CRL on. If you do not specify CRL as an option, then CRL remains off. If the first option for SSLClientAuth equals 0/none,

then you cannot use the second option, CRL. If you do not have client authentication on, then CRL processing does not take place.

Identifying directives supported in global or server and virtual host. Global server and virtual host support the following directives:

- SSLCRLHostname: The IP Address and host of the LDAP server, where the CRL database resides.
- SSLCRLPort: The port of the LDAP server where the CRL database resides; the default equals 389.
- SSLCRLUserID: The user ID to send to the LDAP server where the CRL database resides; defaults to anonymous if you do not specify the bind.
- SSLStashfile: The fully qualified path to file where the password for the user name on the LDAP server resides. This directive is not required for an anonymous bind. Use when you specify a user ID. Use the **sslstash** command, located in the bin directory of IBM HTTP Server, to create your CRL password stash file. The password you specify using the **sslstash** command should equal the one you use to log in to your LDAP server.

  Usage: `sslstash [-c] <directory_to_password_file_and_file_name> <function_name> <password>`

where:

- **-c**: Creates a new stash file. If not specified, an existing file updates.
- **File**: Represents the fully qualified name of the file to create, or update.
- **Function**: Indicates the function for which to use the password. Valid values include crl, or crypto.
- **Password**: Represents the password to stash.

Cache messages

The following message appears due to caching problems:

- Message: **SSL0600S: Unable to connect to session ID cache**
  - Reason: The server was not able to connect to the Session ID caching daemon.
  - Solution: Verify that the daemon was successfully started.

Secure Sockets Layer stash utility errors

The following messages appear due to SSL Stash utility errors:

- Message: **SSL0700S: Invalid function <*function*>**
  - Reason: An invalid parameter was entered. The valid values are crl or crypto.
  - Solution: Rerun the command with the proper function.
- Message: **SSL0701S: The password was not entered.**
  - Reason: The password was not entered on the command line.
  - Solution: Rerun the command with the password added.
- Message: **SSL0702S: Password exceeds the allowed length of 512.**
  - Reason: The password that was entered is longer than the allowed maximum of 512 characters.
  - Solution: Use a shorter password.

Viewing error messages from a target server start

If you encounter an error starting a target server, the error message, line number in the configuration file and the actual line text that caused the error display.

To view the line text error in context:

1. Click **View configuration > Edit configuration**.
2. Select the text.
3. Copy the text.
4. Go to **View configuration > Edit configuration** and press **Ctrl + F** for Find.
5. Paste the text.
6. Click **OK**.

Could not connect to IBM HTTP Server administration server error

If you get the following error:

```
"Could not connect to IHS Administration server error"
```

when you are managing an IBM HTTP Server using the WebSphere administrative console, try one of the following:

- Verify that the IBM HTTP Server administration server is running.
- Verify that the Web server hostname and port that is defined in the WebSphere administrative console matches the IBM HTTP Server administration host name and port.
- Verify that the firewall is not preventing you from accessing the IBM HTTP Server administration server from the WebSphere administrative console.
- Verify that the user ID and password that is specified in the WebSphere administrative console, under remote managed, is created in the `admin.passwd` file, using the **htpasswd** command.
- If trying to connect securely, verify that you export the IBM HTTP Server administration server keydb personal certificate into the WebSphere key database as a signer certificate. This key database will be specified by the `com.ibm.ssl.trustStore` in the `sas.client.props` file in the profile your console is running in. This is mainly for self-signed certificates.
- If you still have problems, check the IBM HTTP Server `admin_error.log` file and the WebSphere Application Server logs (`trace.log`) to see if problem can be determined.

Hints and tips for managing IBM HTTP Server using the WebSphere administrative console

The following list describes hints and tips on starting, stopping and obtaining status for IBM HTTP Server using the WebSphere administrative console:

- On Windows operating systems, the IBM HTTP Server you are managing must be installed as a service. If you are managing IBM HTTP Server on a managed node using NodeAgent, you must install IBM HTTP Server with log on as system rights.
- When defining a Web server for Windows operating systems using the WebSphere administrative console, use the actual service name, instead of the display name . The actual service name will not contain spaces. If you do not do this, you will have problems starting and stopping the service on the Windows 2003 operating system, and on a managed node.

- Do not modify the PID file in the `httpd.conf` file or use a non-standard httpd executable. Assumptions are made based off the designated IBM HTTP Server installation directory when defining a Web server. The executable name is expected to be `<install_dir>\bin\apache.exe` on Windows operating systems, or `<install_dir>/bin/apachectl` on UNIX platforms. For status purposes, obtain the PID in the `<install_dir>/logs/httpd.pid` file. If these values are changed, starting, stopping and obtaining status of IBM HTTP Server will not work.
- Status is obtained using the Web server host name and port that you have defined. You do not use the remote administration port. If "use secure protocol" is defined, SSL will be used.
- If you have problems starting and stopping IBM HTTP Server, check the WebSphere console logs (trace) and if using the IBM HTTP Server administration server, check the `admin_error.log` file.
- If you have problems starting and stopping IBM HTTP Server on a managed node using nodeagent, you can try to start and stop the server by setting up the managed profile and issuing the `startserver <IBM HTTP Server> -nowait -trace` command and check the `startServer.log` file for the IBM HTTP Server specified.

The following list describes hints and tips for viewing log files, editing configuration files and propagating the plug-in configuration file:

- File permissions must be correct in order to transfer a file on UNIX platforms. The **setupadm** script is provided to set appropriate file permissions.
- Access to files is controlled by AdminAllowDirective in the `admin.conf` file. Access is granted to the `conf` and `logs` directory from the IBM HTTP Server installation directory. If you are reading or writing plug-in configuration or trace files, you must add an entry to the `admin.conf` file to allow access there.
- Always back up the configuration file. It is possible on the upload of the configuration file, information will be lost.

GSKit certificate support limitations

You cannot use IKEYMAN to create certificates with key sizes larger than 1024 bits. However, you can import certificates with key sizes up to 4096 into the key database.

Known problems with hardware cryptographic support

You must install the `bos.pkcs11` package on the AIX platform to get the PKCS11 module, and to intialize the device on AIX.

An added update to the `bos.pkcs11` package fixed a forking problem. Obtain the most recent copy of the bos.pkcs11package from the IBM PSeries Support Site, to ensure you have this fix.

The `ikmuser.sample` file that is shipped with the GSKit Toolkit typically installs in the following directories, depending on the platform:

- AIX: `/usr/opt/ibm/gskta/classes`
- HP: `/opt/ibm/gsk7/classes`
- Linux: `/usr/local/ibm/gsk7/classes`
- Solaris: `/opt/ibm/gsk7/classes`
- Windows: C:\Program Files\ibm\gsk7\classes

Renaming this file to `ikmuser.properties` in the `classes` directory enables IKEYMAN to use it for a cryptographic token.

If you are having problems using the IBM eBusiness Cryptographic Accelerator Device with IBM HTTP Server 6.0, do the following:

1. Reboot the machine.
2. Kill **pkcsslotd** and the shared memory it created. To determine what shared memory was created, type`ipcs -a` and for a size of 270760. This was the memory created by **pkcsslotd**.
3. Do an export EXPSHM=ON.
4. Start the pkcs11 process: `/etc/rc.pkcsw11`
5. Restart the IBM HTTP Server: `./apachectl start`

Known problems on the HP-UX platform

You cannot install one version of GSKit onto another. Delete the current GSKit files from your system before installing a new GSKit version.

LDAP Secure Sockets Layer limitation with Netscape LDAP server. The LDAP client has a limitation when using Secure Sockets Layer (SSL) to communicate with a Netscape directory server. If the Netscape directory server has client authentication enabled, the connection fails. If the IBM HTTP Server uses SSL with LDAP to check authentication information on a Netscape Directory Server, ensure that client authentication is not enabled on the directory server.

Known problems on the Solaris platform

A known problem on the Solaris operating system includes specifying a valid ServerName directive.

On some Solaris machines (level unknown), an error is received at IBM HTTP Server startup (apachectl). The error indicates that the ServerName directive is not set in the IBM HTTP Server configuration file, `httpd.conf`. To resolve this problem, supply a valid ServerName directive.

Known problems on the Linux PowerPC platform

Getting the suexec module to work. The suexec module does not work unless you install IBM HTTP Server V6.0 into the default location.

Running the /<*ihs install root*>/bin/httpd command. Source the /<*ihs install root*>/bin/envvars file first to ensure you can run the **/<*ihs install root*>/bin/httpd** command, used to start the IBM HTTP Server. To source the envvars file enter the following at command line:

./<*ihs install root*>/bin/envvars

The envvars file contains the path to the libraries needed to run the **/<*ihs install root*>/bin/httpd** command.

Known problems on Windows operating systems

Problems when the IBM HTTP Server runs on the same system as a Virtual Private Networking Client. A problem occurs when the IBM HTTP Server runs on a system, along with a Virtual Private Networking client, for example, Aventail Connect. You can experience the following problem, or see the following error message:

- The IBM HTTP Server does not start - reference Apache FAQ.
- The IBM HTTP Server does not start. The error log contains the following message:

  ```
  "[crit] (10045) The attempted operation is not supported for the type of object
  referenced: Parent: WSADuplicateSocket failed for socket ###"
  ```

Aventail Connect is a Layered Service Provider (LSP) that inserts itself, as a shim, between the Winsock 2 API and the Windows native Winsock 2 implementation. The Aventail Connect shim does not implement WASDuplicateSocket, the cause of the failure. The shim is not unloaded when Aventail Connect is shut down.

Fix the problem by doing one of the following:

- Explicitly unloading the shim
- Rebooting the machine
- Temporarily removing the Aventail Connect V3.x shim

Configuring security on Internet Explorer V5.01x

If IBM HTTP Server uses a Verisign Global Server ID for SSL transactions, a 40-bit encryption browser can get a connection to a server at 128-bit encryption. This connection does not work for someone using Internet Explorer 5.01x. You can fix this situation by adding the following directives to the IBM HTTP Server configuration file (add the directives in the order shown):

- SSLCipherSpec 34
- SSLCipherSpec 35
- SSLCipherSpec 3A
- SSLCipherSpec 33
- SSLCipherSpec 36
- SSLCipherSpec 39
- SSLCipherSpec 32
- SSLCipherSpec 31
- SSLCipherSpec 30

Contacting Customer Service and Support

For additional help, see the WebSphere Application Server support page.

You can also contact the IBM Software Support Center (1-800-IBM-SERV in the US and Canada).

Glossary

authentication
: In computer security, verification of the identity of a user or the user's eligibility to access an object.

cache To place, hide, or store frequently used information locally for quick retrieval.

cache accelerator
Provides support for caching on multiple Web servers and on servers with multiple IP addresses.

certificate authority (CA)
In computer security, an organization that issues certificates. The certificate authority authenticates the certificate owner's identity and the services that the owner is authorized to use. It also manages the issuance of new certificates and revokes certificates from unauthorized users who are no longer authorized to use them. A certificate authority is considered to be trusted when a user accepts any certificate issued by that certificate authority as proof of the certificate owner's identity.

certificate revocation list (CRL)
A list of certificates that need to be revoked before their expiration date.

cipher In Cryptographic Support, data that is unintelligible to all except those who have the key to decode it to plaintext.

cipher specifications
Indicate the data encryption algorithm and key size to use for secure connections.

cryptographic support
The IBM licensed program that provides support for the encryption and decryption of data, according to the Data Encryption Algorithm, and for the management of cryptographic keys and personal identification numbers (PINs).

Data Encryption Standard (DES)
In computer security, the National Institute of Standards and Technology (NIST) Data Encryption Standard, adopted by the U.S. government as Federal Information Processing Standard (FIPS) Publication 46, which allows only hardware implementations of the data encryption algorithm.

digital certificate
A form of personal identification that can be verified electronically. Only the certificate owner who holds the corresponding private key can present a certificate for authentication through a Web browser session. Anyone can verify that the certificate is valid by using a readily available public key.

digital signature
Information that is encrypted with an entity private key and is appended to a message to assure the recipient of the authenticity and integrity of the message. The digital signature proves that the message was signed by the entity that owns, or has access to, the private key or shared secret symmetric key.

directive
A statement that is used in the configuration file for a Web server to define a particular setting for the server.

distinguished name (DN)
In computer security, information that uniquely identifies the owner of a certificate.

dynamic shared object (DSO)
A mechanism which provides a way to build a piece of program code in a special format for loading at run time into the address space of an

executable program. The DSO gets knowledge of the executable program symbol set as if it had been statically linked with it in the first place

encrypt
In Cryptographic Support, to systematically scramble information so that it cannot be read without knowing the coding key.

environment variable
A variable that specifies how an operating system or another program runs, or the devices that the operating system recognizes.

Fast Common Gateway Interface Protocol (FastCGI)
The Fast Common Gateway Interface (FastCGI) is an enhancement to the existing Common Gateway Interface (CGI), which is a standard for interfacing external applications with Web servers.

handshake
A Secure Sockets Layer (SSL) session always begins with an exchange of messages called the SSL handshake. The handshake allows the server to authenticate itself to the client by using public key techniques, and then allows the client and the server to cooperate in the creation of symmetric keys used for rapid encryption, decryption, and tamper detection during the session that follows. Optionally, the handshake also allows the client to authenticate itself to the server.

Java An object-oriented programming language for portable interpretive code that supports interaction among remote objects. Java was developed and specified by Sun Microsystems, Incorporated.

Java Development Kit (JDK)
A software package that can be used to write, compile, debug, and run Java applets and applications.

Java Runtime Environment (JRE)
A subset of the Java Development Kit (JDK) that contains the core executables and files that constitute the standard Java platform. The JRE includes the Java Virtual Machine (JVM), core classes, and supporting files.

Java Virtual Machine (JVM)
A software implementation of a central processing unit (CPU) that runs compiled Java code (applets and applications).

key In computer security, a sequence of symbols that is used with a cryptographic algorithm for encrypting or decrypting data.

key database
Exists as 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.

key file
In the Distributed Computing Environment (DCE), a file that contains encryption keys for noninteractive principals.

key pair
Contains a public, distributed key and a private key. A key pair is issued by a public key cryptography system and is used in combination with each other to validate and authenticate a connection between a client and server for secure connections.

Lightweight Directory Access Protocol (LDAP)
In TCP/IP, a protocol that enables users to locate people, organizations, and other resources in an Internet directory or intranet directory.

module
A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading.

password stashing
The password is encrypted in a file or on a hard drive. Your keydb password needs to reside in a file in order to use secure sockets layer (SSL).

PKCS12
Sometimes referred to as PFX files; PKCS#12 files are used by several programs including Netscape, MSIE and MS Outlook.

plug-in
A self-contained software component that modifies (adds or changes) function in a particular software system. When a user adds a plug-in to a software system, the foundation of the original software system remains intact. The development of plug-ins requires well defined application programming interfaces (APIs).

port (1) A system or network access point for data entry or exit. (2) A connector on a device to which cables for other devices such as display stations and printers are attached. (3) The representation of a physical connection to the link hardware. A port is sometimes referred to as an adapter; however, there can be more than one port on an adapter. One or more ports are controlled by a single data link control (DLC) process. (4) In the Internet suite of protocols, a specific logical connector between the Transmission Control Protocol (TCP) or the User Datagram Protocol (UDP) and a higher level protocol or application. (5) To modify a computer program to enable it to run on a different platform.

port number
In the Internet suite of protocols, the identifier for a logical connector between an application entity and the transport service.

private key
In secure communication, an algorithmic pattern used to encrypt messages that only the corresponding public key can decrypt. The private key is also used to decrypt messages that were encrypted by the corresponding public key. The private key is kept on the user's system and is protected by a password.

public key
In secure communication, an algorithmic pattern used to decrypt messages that were encrypted by the corresponding private key. A public key is also used to encrypt messages that only the corresponding private key can decrypt. Users broadcast their public keys to everyone with whom they must exchange encrypted messages.

public key infrastructure (PKI)
An infrastructure that supports digital signatures and other public key-enabled security services.

Secure Sockets Layer (SSL)
A security protocol that provides communication privacy. SSL enables client and server applications to communicate in a way that is designed to

prevent eavesdropping, tampering, and message forgery. SSL was developed by Netscape Communications Corporation and RSA Data Security, Inc.

stash file
A file that hides other data files within.

symmetric keys
In computer security, the two keys in a key pair. The keys are called symmetric because each key holds as much of the encryption pattern as the other does.

trust policy
Contains a trusted list of certificates that are used to control the trust and validity period of certificates. It enables one to limit the trust of certificates issued by a certificate authority.

trusted root
A certificate signed by a certificate authority (CA), designated as a trusted CA on your server.

virtual host
Refers to the practice of maintaining more than one server on one machine, differentiated by their apparent host name.

X.500 The directory services standard of International Telecommunication Union (ITU), International Organization for Standardization (ISO), and International Electrotechnical Commission (IEC).

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, New York 10594 USA

Trademarks and service marks

For trademark attribution, visit the IBM Terms of Use Web site (http://www.ibm.com/legal/us/).