Apache Web Server SSL Configuration for Command View Applications

Revision 1.1

10/8/2002

A White Paper

Network Storage Solutions Lab

Hewlett-Packard

1 Table of Contents

2 Introduction

This paper discusses how to set up a Windows NT/2000 version of the Apache web server for SSL operation and the steps required to use the Apache SSL web server with Command View.

3 Apache Configuration for Running Command View Applications

Diagram below shows the basic Command View web server configuration.

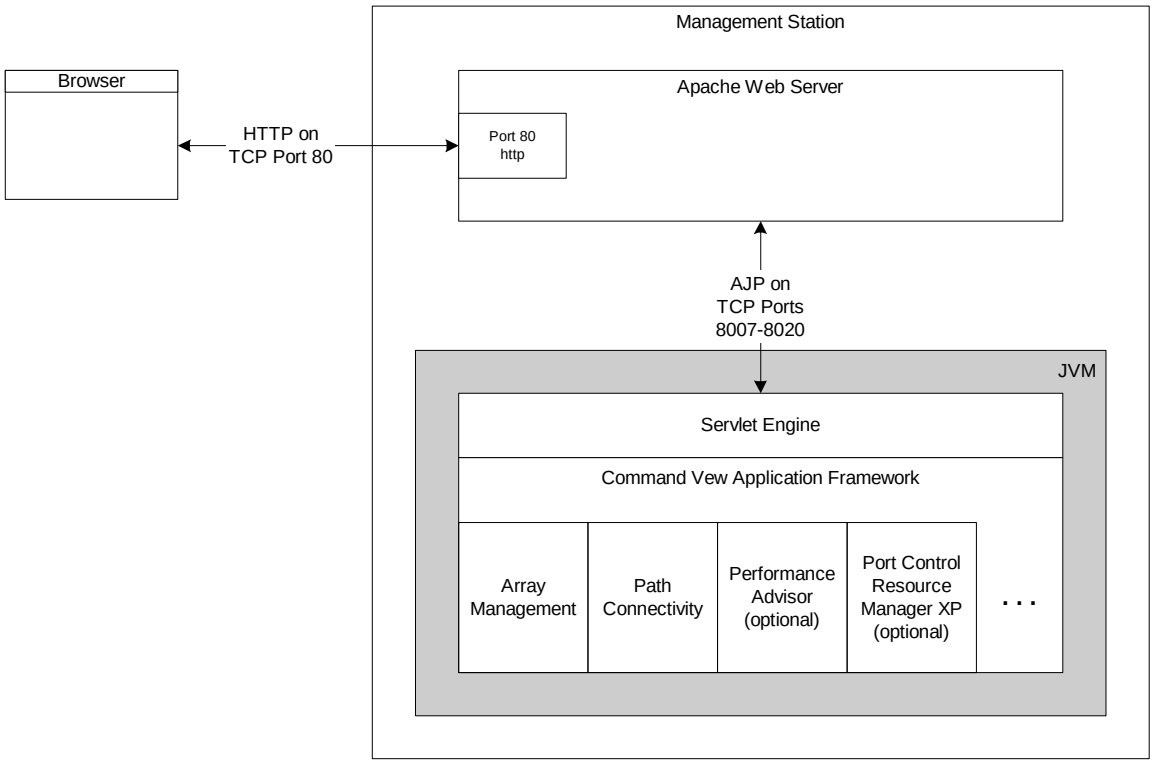


Figure 1

The Apache web server handles all http requests. The Command View Application Framework and Application modules are implemented as a collection of Java servlets. Application requests from a browser are routed by the web server to the appropriate application. In order to perform this routing, the apache web server must be informed how to route application specific URLs.
The Apache Web Server is made aware of the Command View Application Framework by using a "mod_jk" communications module. Configuration files that contain Apache and "mod_jk" configuration commands are incorporated into Apache's master "httpd.conf" configuration file using the Apache "include" directive. The configuration directives will not be discussed here, but it will be necessary for the user to add these "include" directives to an Apache SSL web server's "httpd.conf" file. This will be discussed in greater detail later in this document.

4 SSL Encryption

4.1 Introduction

This section attempts to explain the concepts of the Secure Socket Layer (SSL) within a narrow context of client-server communication in order to provide some background to improve understanding of the material in this document that for the most part deals with specific SSL client and server configuration for the Apache web server and the ***hp StorageWorks command view xp*** application.

4.2 Purpose

The purpose of SSL is to ensure that the server to which you are connecting can be trusted and to prevent access to information as it is transmitted between the client and server.
Once a secure SSL connection is established by validating the server, all data transmissions are encrypted between the client and server.

4.3 Encryption Concepts

A digital certificate is just a statement signed by an independent and trusted third party. In the case of SSL this third party is the Certificate Authority (CA) like VeriSign or Thawte.
The digital signature is a digital stamp using a cryptographic algorithm. SSL communication relies on these cryptographic algorithms for establishing both the trust relationship and the secure communications.
The trust relationship is based on a form of cryptographic algorithm that uses a unique public and private key pair. The private key is used to encrypt information that only the public key can decrypt. The CA distributes its public key which can then be used to verify that a certificate has been issued by that CA.
Once the SSL connection is established with a trusted server, then the actual data encryption is usually performed using a symmetric key cryptographic algorithm due to the much greater efficiency.
Most SSL secure communication uses a 1024 bit public/private key for certificate verification and a 128 bit symmetric key for data encryption. Each of which provides roughly the same very high level of security.

4.4 Digital Certificates and Certificate Authorities

To establish the trust relationship, the owner (company or individual) of the server must apply for a digital certificate from a Certificate Authority (CA) such as VeriSign or Thawte. Before granting a certificate, the CA will request detailed information from the applicant and perform the necessary checks to verify the information about the applicant.
The CA will then issue a digital certificate to the applicant that can be used with a SSL configured web server.

4.5 Self-Signed Certificates

It is possible to “sign” your own server certificate, however your “signature” will not be widely recognized by browsers or other software and it will be necessary to install your server certificate individually on all clients in order to get them to trust your server.
Most web browsers and other software come with pre-installed certificates from the major Certificate Authorities. Therefore CA signed server certificates are automatically trusted by web browsers and the Java SSL security software.

5 Apache SSL Configuration for Running Command View Applications

Moving to a SSL configuration is a little like entering another country where different laws apply. Connections between clients and servers can no longer be established without first passing through “security”. In order to establish a SSL connection, a client must “trust” the server to which it is connecting. This trust relationship is managed through digital certificates. These digital SSL certificates are required to establish SSL connections. The purchase and installation of these certificates is the responsibility of the customer because they are granted by the CA only to the owner of the server.
If a server certificate is purchased for the server from a CA, then it is only necessary to install the server certificate into the server. Other components shown, come with certificates pre-installed from the major CA companies. If a self-signed server certificate is installed into the server however, then the self-signed server certificate will need to be imported into all the other components in order to get them to "trust" the server. It is strongly recommended that a CA signed certificate be purchased for the server and that a self-signed certificate be used only for limited testing.

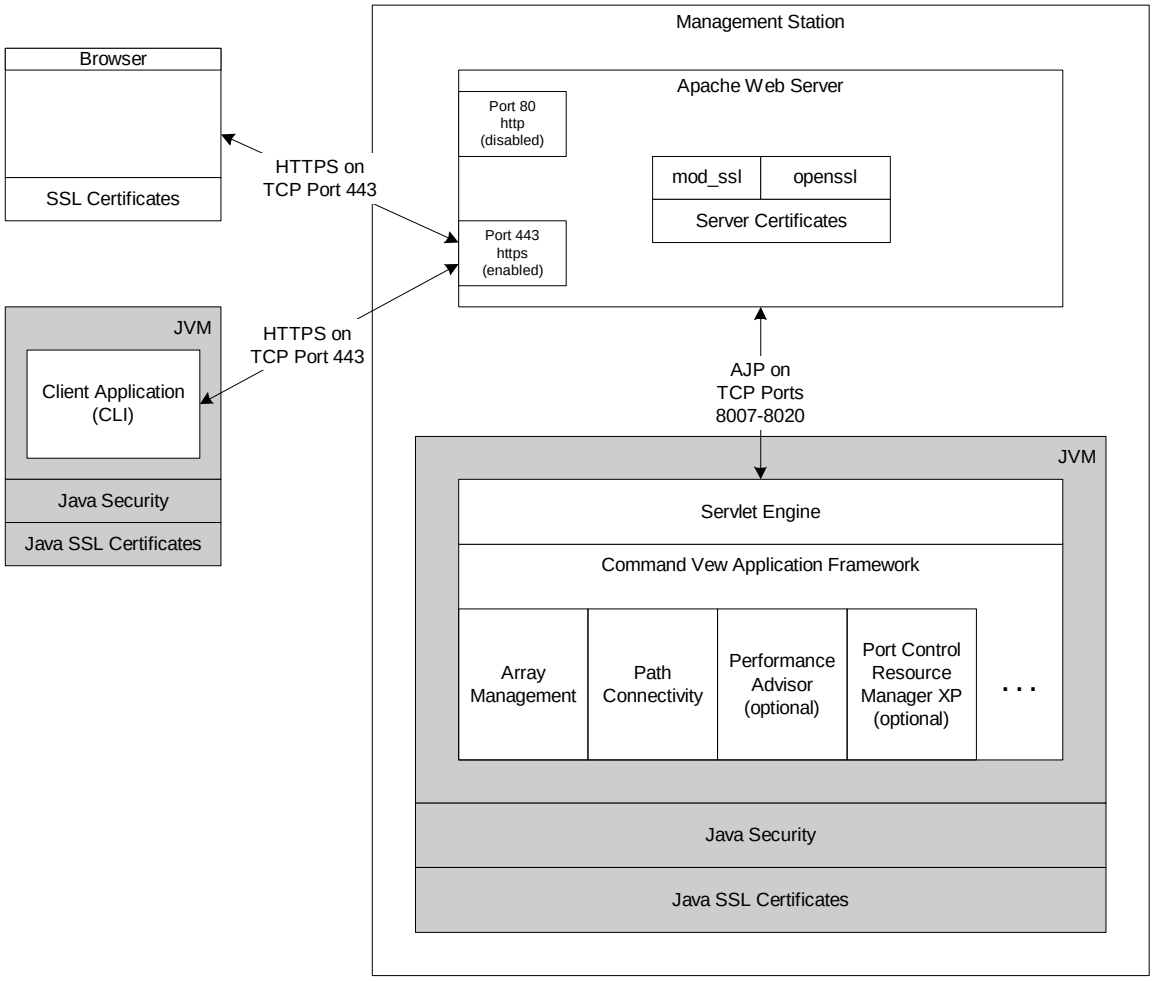


Figure 3

6 Configuring Command View for SSL

The following steps are necessary to prepare Command View for SSL operation.

- Install an Apache SSL enabled web server. Refer to section “7 Apache SSL Web Server for Windows NT/2000” in this document.
- Create a private key and certificate request for the Apache SSL web server. Refer to section “8 Certificates” in this document.
- Configure the Command View Management Service for SSL operation. Refer to section “9 Command View Management Service SSL Requirements” in this document.
- Configure client/host software for SSL operation. Refer to section “10 Client/Host SSL Requirements” in this document.
- If the SNMP SubAgent (Proxy Agent) is being used, configure it for SSL operation. Refer to section “11 SubAgent (Proxy Agent) SSL Requirements” in this document.

7 Apache SSL Web Server for Windows NT/2000

The Apache web server installed during the Command View installation does not support SSL (https) connections. At the time of this writing the Apache+mod_ssl+openssl software was not available as a compined pre-build binary installation for the Windows operating system directly from www.apache.org or www.modssl.org .

Therefore we will need to do one of the following:

- Create it from open source code distributions of Apache (www.apache.org), modSSL (www.modssl.org), and OpenSSL (www.openssl.org) .
  In this case you will be building the a Apache+mod_ssl+openssl web server and installing it in place of the current HPSS\apache directory.
- Acquire a pre-built open source binary installation from another individual or organization, e.g. the “OpenSA” web server (www.opensa.org) .
- Use a commercial Apache+mod_ssl web server like the Covalent SSL server.

Each of these options will be discussed further in the following sections.

NOTE: The Command View HPSS\apache directory is always installed whether or not you chose to install the Apache service from the installer. This is to provide you with the required configuration files that are installed as part of the original installation, and to provide you with the HpssApache service to start and run the Apache web server should you choose to enable this service.

7.1 The Apache+mod_ssl+openssl Web Server

The SSL protocol layer sits between the Apache web server and the TCP/IP layer, and handles the encryption and decryption when a secure connection is established with a client. The Apache web server distribution from www.apache.org does not include the SSL layer. This layer must be added using an Apache "SSL module".
The Apache web server functionality is extended through "modules". Each module adds additional functionality and in addition, each module can be configured by adding directives to the "httpd.conf" file. For the SSL layer, we have chosen the "mod_ssl" Apache module. This Apache "mod_ssl" version of the Apache web server consists of three major components: the Apache web server, the "mod_ssl" module, and the OpenSSL encryption library.

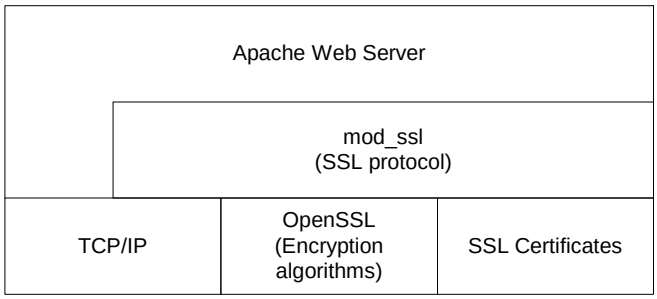


Figure 2

The "mod_ssl" module requires the "OpenSSL" encryption library to perform the encryption and decryption operations associated with the SSL protocol.
In addition, when using the SSL protocol, it is necessary to install the appropriate digital certificate on the server.
The OpenSSL software also includes a separate application for creating and maintaining SSL keys and certificates. OpenSSL will be used later to create a private key and certificate request in order to obtain a CA signed server certificate for the server.
More information on "Apache" is available from: http://www.apache.org
More information on "mod_ssl" is available from: http://www.modssl.org
More information on "OpenSSL is available from: http://www.openssl.org

7.1.1 Building the Apache+mod_ssl+openssl Web Server

To build the basic Apache+mod_ssl+openssl web server from the source code distributions, it is first necessary to compile the OpenSSL source files and install the resulting OpenSSL binaries. Next, the Apache web server source code must be modified to include the “mod_ssl” module. Finally, the Apache web server must be compiled with the “mod_ssl” module additions to produce the final Apache+mod_ssl binary executable.
Refer to the appendix for detailed instructions how to build an Apache+mod_ssl+openssl web server from the open source distributions.

7.1.2 Replacing the HPSS Apache Web Server

At this point it is assumed that you have build or acquired pre-build binaries of the following components:

- OpenSSL
- Apache+mod_ssl web server

It is also assumed that OpenSSL is installed on the Command View host (refer to appendix C, "Compiling and Installing OpenSSL), and it is on this host that the Apache+mod_ssl web server will be used to replace the Command View Apache web server.
The Command View HPSS\apache web server httpd.conf configuration file is pre-configured with default “mod_ssl” directives that will only take affect if used with an Apache+mod_ssl+openssl web server. This allows the httpd.conf file installed during the Command View installation to be used with an Apache+mod_ssl+openssl web server built from the open source distributions.
The httpd.conf file contains SSL directives that can be identified by looking for the following directives:
<IfModule mod_ssl.c> (this begins a SSL directive section)
</IfModule> (this ends a SSL directive section)
All directives between the <IfModule mod_ssl.c> and </IfModule> directives, apply only when the Apache “mod_ssl” module is present.

NOTE 1: You will be replacing the existing Apache web server, but you will be relying on the HpssApache service that was installed as part of the original Command View installation to start and stop the Apache SSL web server. If you chose not to install the Apache component in the original installation, both the Apache files and the HpssApache Apache service were installed, but the service “Startup Type” setting was set to “Disabled”. This setting will need to be changed from “Disabled” to “Automatic” once the Apache SSL is installed (see below).

NOTE 2: If you have already installed ***Performance Advisor***, you will need to copy an additional “authentication” directory to the Apache SSL web server directory (see below).

Use the following instructions to install the Apache SSL web server to replace the functionality of the original “HPSS\apache” web server.

1) Stop the Command View services.
2) You will need to copy several files from the existing Command View "**C:\HPSS\apache**" sub-directories to the corresponding "**C:\Apache**" sub-directories.
   a) Copy file **C:\HPSS\apache\conf\httpd.conf** to the **C:\Apache\conf** directory.
   b) Copy file **C:\HPSS\apache\htdocs\index.html** to the **C:\Apache\htdocs** directory.
   c) Copy file **C:\HPSS\apache\conf\mod_jk_common.conf** to the **C:\Apache\conf** directory.
   d) Copy file **C:\HPSS\apache\conf\workers.properties** to the **C:\Apache\conf** directory.
   e) Copy file **C:\HPSS\apache\modules\mod_jk.dll** to the **C:\Apache\modules** directory.
   f) If ***Performance Advisor*** is installed, you must also copy the **C:\HPSS\apache\auth** directory and all files within to **C:\Apache**. There should now be a **HPSS\Apache\auth** directory.
3) Replace the **C:\HPSS\apache** with the newly built **C:\Apache**
   a) Rename the **C:\HPSS\apache** directory to “**apache_http**”.
   b) Copy the Apache SSL web server directory and all it’s all subdirectories (**C:\Apache**) to **C:\HPSS\apache**.
4) In the httpd.conf file:
   a) All “ServerName” directives should use the fully qualified DNS name of the server.
   b) Uncomment the following line:
      LoadModule ssl_module modules/mod_ssl.so (remove the “#” from in front of the “LoadModule ssl_module modules/mod_ssl.so” directive.
   c) Comment the following lines:
      i) **Port 80** (place a “#” before the “Port 80” directive)
      ii) **Listen 80** (place a “#” before the “Listen 80” directive)
5) If you installed Command View with the Apache component “unchecked”, you will need to set the **HpssApache** service “Startup Type” to “Automatic”. Refer to appendix B for details how to change the “Startup Type” of the HpssApache service.
6) Start the Command View services.

7.2 Using the OpenSA Web Server

The OpenSA web server is also an open source Windows Apache+mod_ssl+openssl web server that is available as a pre-built binary from OpenSA (www.opensa.org).
Use the following instructions to install the OpenSA web server to replace the functionality of the original "HPSS\apache" web server.

1) Disable the **HpssApache** service (see appendix A or B).
2) Obtain the ***OpenSA*** binary distribution from ***OpenSA*** the web site (http://www.opensa.org).
3) Install the ***OpenSA*** web server accepting the default installation parameters.
   NOTE: The installer may ask you to enter certain other parameters like the "Server Name" that will be used to modify the Apache "httpd.conf" configuration file.
4) You will need to copy several files from the Command View "**C:\HPSS\apache**" sub-directories to the corresponding "**OpenSA\Apache**" sub-directories.
   a) Copy file **C:\HPSS\apache\htdocs\index.html** to the **OpenSA\Apache\htdocs** directory.
   b) Copy file **C:\HPSS\apache\conf\mod_jk_common.conf** to the **OpenSA\Apache\conf** directory.
   c) Copy file **C:\HPSS\apache\conf\workers.properties** to the **OpenSA\Apache\conf** directory.
   d) Copy file **C:\HPSS\apache\modules\mod_jk.dll** to the **OpenSA\Apache\modules** directory.
   e) If ***Performance Advisor*** is installed, you must also copy the **C:\HPSS\apache\auth** directory and all files within to **OpenSA\Apache**. There should now be a **OpenSA\Apache\auth** directory.
5) Modify the **OpenSA\Apache\conf\httpd.conf** configuration file as follows:
   a) Add the following lines at the top of the file:
      i) **Include "C:/HPSS/Apache/conf/mod_jk_common.conf"**
      ii) **Include "C:/HPSS/e2e/tomcat/conf/mod_jk.conf"**
      iii) **Include "C:/HPSS/dm/tomcat/conf/mod_jk.conf"**
      iv) If Performance Advisor is installed add:
         (1) **Include "C:/HPSS/pa/tomcat/conf/mod_jk.conf"**
      v) If Application Policy Manager XP is installed add:
         (1) **Include "C:/HPSS/apm/tomcat/conf/tomcat-apm.conf"**
   b) All "ServerName" directives should use the fully qualified DNS name of the server.
   c) Delete or comment the "**Port 80**" directive, approximately at line 236 of the file (place a "#" before the "Port 80" directive).
   d) Delete or comment the "**Listen 80**" directive, approximately at line 245 of the file (place a "#" before the "Listen 80" directive).
   e) At the root directory directive "<Directory />", change the "AllowOverride None" to "AllowOverride AuthConfig".
6) Restart the OpenSA Apache service.

7.3 Using a Commercial Windows Apache+mod_ssl Web Server

The Covalent SSL web server is one example of a commercial Windows Apache web server that can be used instead of the Commmand View Apache web server. The Covalent SSL web server is based on Apache and modSSL. It does not include OpenSSL but rather relies upon, and includes, a different SSL library. Covalent provides their own user interface for creating and managing SSL certificates. Refer to the Covalent documentation for managing certificates.

The Covalent SSL web server is used in this example to replace the functionality of the Command View Apache web server:

1) Disable the **HpssApache** service (see appendix A or B).
2) Install the ***Covalent SSL*** web server.
   NOTE: The installer may ask you to enter certain other parameters like the “Server Name”.
3) You will need to copy several files from the Command View "**C:\HPSS\apache**" sub-directories to the corresponding "**Covalent\Apache**" sub-directories.
   a) Copy file C:\HPSS\apache\htdocs\index.html to the Covalent \Apache\htdocs directory.
   b) Copy file **C:\HPSS\apache\conf\mod_jk_common.conf** to the **Covalent \Apache\conf** directory.
   c) Copy file **C:\HPSS\apache\conf\workers.properties** to the **Covalent \Apache\conf** directory.
   d) Copy file **C:\HPSS\apache\modules\mod_jk.dll** to the **Covalent \Apache\modules** directory.
   e) If ***Performance Advisor*** is installed
      i) Copy the **C:\HPSS\apache\auth** directory and all files within to **Covalent \Apache**.There should now be a **Covalent\Apache\auth** directory.
4) Modify the **Covalent \Apache\conf\httpd.conf** configuration file as follows:
   a) Add the following lines at the top of the file:
      i) **Include "C:/HPSS/Apache/conf/mod_jk_common.conf"**
      ii) **Include "C:/HPSS/e2e/tomcat/conf/mod_jk.conf"**
      iii) **Include "C:/HPSS/dm/tomcat/conf/mod_jk.conf"**
      iv) If Performance Advisor is installed add:
         (1) **Include "C:/HPSS/pa/tomcat/conf/mod_jk.conf"**
      v) If Application Policy Manager XP is installed add:
         (1) **Include "C:/HPSS/apm/tomcat/conf/tomcat-apm.conf"**
   b) All “ServerName” directives should use the fully qualified DNS name of the server.
   c) Delete or comment the “**Port 80**” directive.
   d) Delete or comment the “**Listen 80**” directive.
   e) At the root directory directive “<Directory />”, change the “AllowOverride None” to “AllowOverride AuthConfig”.
5) Restart the Covalent Apache service.

8 Certificates

In practice we only need to create a private key and certificate for the server. It is possible for the server to request certificate authentication from a client, but it is not common and is not covered in this document. It is strongly recommended that a CA signed certificate be purchased for the server and that a self-signed certificate be used only for limited testing.

In this section we discuss:

- Creating a private server key.
- Creating a certificate-signing request (CSR).
- Purchasing a CA signed certificate.
- Creating a temporary self-signed certificate to use while waiting for the CA to process and return the CA signed certificate.
- Installing the self-signed certificate.
- Removing the self-signed certificate.
- Installing the CA signed certificate.

8.1 Certificate Authorities

Two widely used certificate authorities (CA) are Verisign (http://www.verisign.com) and Thawte (http://www.thawte.com). Information and forms for ordering a certificate are available from each company's web site.

8.2 Creating, Purchasing and Installing a Server Certificate

This section assumes that you have OpenSSL correctly installed on your system and that the "openssl.cnf" file is located directly in the "openssl" directory.

8.2.1 Creating a Private Server Key

Create a private key as follows:

- **openssl genrsa -out server_domain_name.key 1024**

The file can be named anything, but if the server's domain name (e.g. hostname.hp.com) is used it is easier to identify to which host the key (and later certificate) belongs.

8.2.2 Creating a Certificate Signing Request (CSR)

In order to purchase a certificate you will need to generate a certificate signing request (CSR).

- **openssl req -new -key server_domain_name.key -out server_domain_name.csr -config C:\openssl\openssl.cnf**

Follow the same file naming convention chosen for the private key file using an extension of ".csr".

The program will ask you to enter information required to complete your CSR.

- Country Name (2 letter code) [AU]:
- State or Province Name (full name) [Some-State]:
- Locality Name (eg, city) []:
- Organization Name (eg, company) [Internet Widgits Pty Ltd]:
- Organizational Unit Name (eg, section) []:
- Common Name (eg, YOUR name) []:
  NOTE: Enter the server's DNS name here, for example: www.hp.com
- Email Address []:
- A challenge password []:
  Optional – Press "Enter" for none.
- An optional company name []:
  Optional – Press "Enter" for none.

Check your request with:

- **openssl req –text –noout –in server_domain_name.csr**

8.2.3 Creating a Self-Signed Certificate

Create a temporary self-signed certificate to test the server operation:

- **openssl x509 -req -days 365 -in server_domain_name.csr -signkey server_domain_name.key -out server_domain_name.crt**

8.2.4 Installing the Self-Signed Certificate

Proper Command View SSL operation requires that the self-signed certificate be installed in both the server and the JRE on the management station.

8.2.4.1 Installing the Self-Signed Certificate in the Server

The Apache SSL server requires that the key and certificate be known to the server. The both Apache SSL server configurations discussed above are configured to look for the key and certificate in the following directories:

- Key: apache\conf\ssl.key\
  To avoid having to modify the httpd.conf file, create a new copy the server_domain_name.key file with the name "server.key". Otherwise edit the apache\conf\httpd.conf file to reflect the actual name of your key file.
- Certificate: apache\conf\ssl.crt\
  To avoid having to modify the httpd.conf file, create a new copy the server_domain_name.crt file with the name "server.crt". Otherwise edit the apache\conf\httpd.conf file to reflect the actual name of your certificate file.

8.2.4.2 Installing the Self-Signed Certificate in the JRE

If you decide to use a self-signed certificate, you must also install the certificate in the JRE certificate store. This is because the Java part of the application communicates with the Apache SSL web server and must trust the web server's certificate. This is not necessary for a CA signed certificate because the JRE comes with pre-installed certificates from the major CA companies. For a self-signed certificate, the JRE must be told to "trust" the server certificate by installing the server certificate into the JRE certificate file.

- Copy the server_domain_name.crt file to the **C:\Program Files\JavaSoft\JRE\1.3.1\lib\security** directory.
- Open a shell window.
- Change directory to the **C:\Program Files\JavaSoft\JRE\1.3.1\lib\security directory**.
- NOTE: The next step assumes that "**C:\Program Files\JavaSoft\JRE\1.3.1\bin"** is in your path environment variable list. Otherwise you must include the full quoted path to the "keytool" program ("**C:\Program Files\JavaSoft\JRE\1.3.1\bin\keytool" -import -alias server_domain_name -file server_domain_name.crt -keystore cacerts)**
- Type the following command:
    - **keytool -import -alias server_domain_name -file server_domain_name.crt -keystore cacerts**
- You will be prompted for a password. The default password is "changeit".
- Answer yes when asked if the certificate should be trusted.

8.2.4.3 Installing the Self-Signed Certificate in the Browser

Both Internet Explorer and Netscape will let you continue past a failed SSL authentication or give you an option to install the server certificate into the browser's certificate storage area. In the case of a self-signed certificate, the browser will not by default recognize the signing authority (you). Therefore the SSL authentication will fail and you will be prompted to continue or install the self-signed certificate. Please refer to the each browser's documentation or help file if you desire to install the self-signed certificate.

8.2.5 Purchasing a CA Signed Certificate

You can request a "trusted" certificate from a certificate authority like VeriSign or Thawte.

The process of purchasing a certificate varies slightly depending on where you choose to purchase your certificate. For an example of the certificate request process, refer to the document "Securing Your Apache Web Server With a Thawte Digital Certificate" published by Thawte which is available from their web site at: http://www.thawte.com.
Section 7 from "Securing Your Apache Web Server With a Thawte Digital Certificate" is included below in the following section.

8.2.5.1 From: "Securing Your Apache Web Server With a Thawte Digital Certificate"

7 "Request a "trusted" certificate"

Thawte SSL certificates and SuperCerts are requested online from Thawte. During the certificate request process, you will be asked to copy and paste your CSR (Certificate Signing Request) into a text area on the online enrollment form. Please ensure that you are submitting the correct CSR, if you have generated more than one (you can check your CSR as follows: "openssl req –text –noout –in csrfilename.csr").

You will have to provide all the requested information during the enrollment process, and send us documentation proving your, or your company's, identity (a company registration certificate for instance). You can view detailed instructions for obtaining a Thawte SSL certificate at:
https://www.thawte.com/certs/server/request.html

The enrollment process for SuperCerts is the same as for SSL certificates. However, during the process you will need to check the box that indicates that you would like a SuperCert. You will also have to generate a 1024-bit key, and make sure your Apache Server is 128-bit enabled.

Once you have completed the online request process, Thawte will take a number of steps to verify your identity and the other details you provided in the CSR. Thawte performs a considerable amount of background checking before it issues the certificate. As a result, it could take a few days to verify your company identity and details, and issue the certificate. During that period, you can track the progress of your request on your personal status page at http://www.thawte.com/cgi/server/status.exe

SuperCerts *are SSL certificates that allow "international" browsers to "step-up" to 128-bit encryption. Internet Explorer 5.01, Netscape Communicator 4.7 and later browsers recognize Thawte's SuperCerts. 128-bit encryption is regarded as being impossible to "crack". For more information on SuperCerts please see: http://www.thawte.com/certs/server/128bit/contents.html*

8.2.6 Installing the CA Signed Certificate

Before installing the CA signed certificate, you must remove the self-signed certificate.

You can simply copy the CA signed certificate of the same name over the current self-signed certificate in the apache\conf\ssl.crt\ directory.

If you have imported the self-signed certificate into the JRE keystore "cacerts", then remove the certificate with the following command:

- **keytool –delete –alias server_domain_name –keystore cacerts**

9 Command View Management Service SSL Requirements

The command view management service communicates with the Apache web server and the configuration must be modified when the web server is configured for SSL operation.

The "***HPSS\CVManagementServer\config\ServletHost.cfg***" file contains the following lines:

```
# replace hostname with the correct
# hostname of CommandView server
SERVLET_URL=http://localhost/hpstmgmt/servlet/MarsDm
#Use the below for SSL based communication
#SERVLET_URL=https://localhost/hpstmgmt/servlet/MarsDm
```

Edit this to read as follows:

```
# replace hostname with the correct
# hostname of CommandView server
#SERVLET_URL=http://localhost/hpstmgmt/servlet/MarsDm
#Use the below for SSL based communication
SERVLET_URL=https://localhost/hpstmgmt/servlet/MarsDm
```

Note that the SERVLET_URL property should start with "https".

10 Client/Host SSL Requirements

Some client and host machines will also require the JRE and Java Secure Socket Extension (JSSE) to be installed on the client or host if an Apache SSL web server is used on the Command View management station.

- JRE version 1.3.1
- Java Secure Socket Extension (JSSE)
- Self-signed server certificate must be installed into the JRE certificate store.

10.1 General Cli install

Each client or host installation includes specific instructions. Please read the installation notes.

NOTE: If you are using Command View in an SSL only configuration, AND you are using a self-signed server certificate, you MUST install the self-signed server certificate into EACH appropriate client or host certificate store in order for the client or host to trust the server. If you are using a purchased CA signed server certificate, you DO NOT need to install the certificate on either the client of host.
For installing a self-signed certificate into a client or host, refer to section “8.2.4.2 Installing the Self-Signed Certificate in the JRE” in this document. The exact location of the “cacerts” certificate store will depend on the OS type of the particular client or host.

10.2 Array Management

- No host agent SSL requirements.
- CLI client’s CVCLI.properties file must be configured for SSL operation.
- CLI clients require JRE/JSSE. If a self-signed server certificate is used for the management station, then this server certificate must be imported into the JRE certificate store. Please follow the client specific installation instructions.

10.3 Path Connectivity

- No host agent SSL requirements.
- CLI client’s E2ECLI.properties file must be configured for SSL operation.
- CLI clients require JRE/JSSE. If a self-signed server certificate is used for the management station, then this server certificate must be imported into the JRE certificate store. Please follow the client specific installations.

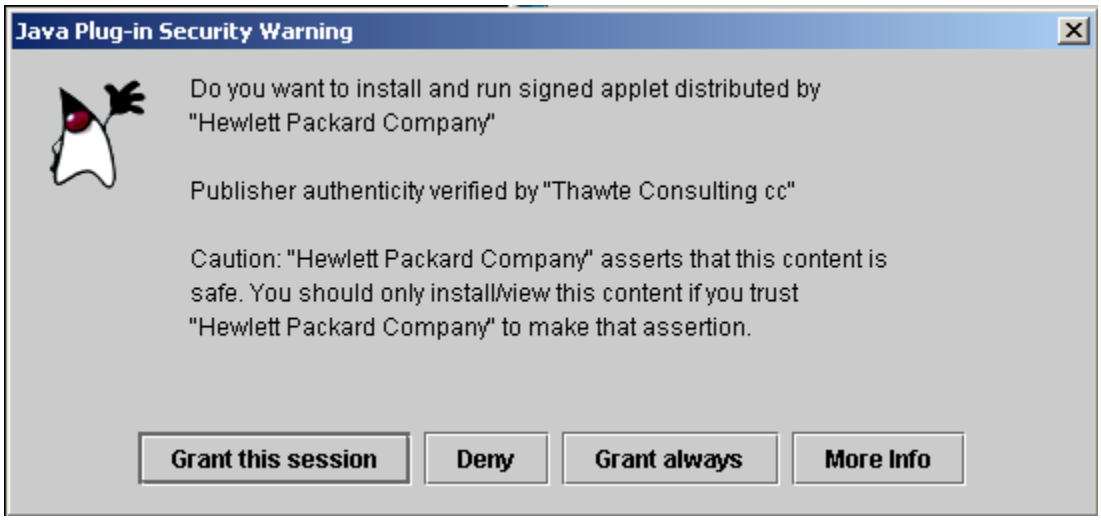
10.4 Performance Advisor

- Performance Advisor host agents require SSL configuration. Please refer to the host installation instructions.
- No client SSL requirements.

10.5 Application Policy Manager XP

10.5.1 Client Browser Access

When accessing the Application Policy Manager XP (APM) application on a client's browser for the first time from a secure web server, the following prompt will be displayed:



If you select the "Grant always" button, then all future access to APM application will take the browser directly to the APM screens. This is the recommended selection.

If you select the "Grant this session" button, then the next time the APM application is accessed, this prompt will be displayed again.

If you select the "Deny" button, then the APM application will not be able to be accessed during this session. The next time the APM application is browsed to, this prompt will be display again.

10.5.2 Client Command-Line Interface Installation and Access

The steps for installing the APM Command-Line Interface (CLI) are listed below.

1. On the Management Station, navigate to <HPSS Install location> \apm\cli.
2. Copy the files cli.jar gamera.jar, jsse.jar, jnet.jar, and jcert.jar to the host or client the CLI is to be installed to.
3. If this is a Windows NT/W2K install, copy apm.bat from the <HPSS Install location> \apm\cli\dos to the same location on the host as specified in step 2.
4. If this is a UNIX install, copy apm.sh from the <HPSS Install location> \apm\cli\unix to the same location on the host as specified in step 2.
5. If this is a HP-UX 10.20 install, copy apm.sh from the <HPSS Install location> \apm\cli\1020 to the same location on the host as specified in step 2.
6. If the Management Station is using a Self-Signed Certificate for it's SSL Secure Web Service, then this Self-Signed Certificate will need to be manually install into keystore used by the Java JRE on the host. (See above Section 8.2.4.2 Installing Certificate in the JRE)

Note: All APM CLI commands communicating with a Secure Web Server will need to include the security switch "-s" parameter in the command line (see APM CLI documentation). Also, HP-UX 10.20 machines using the "-s" parameter do not actually have a secured connection to the Management Station.

11 SubAgent (Proxy Agent) SSL Requirements

The proxy agent also communicates with the web server and must be configured for SSL operation. The "***servletAddr.conf***" configuration file must be configured for SSL. Directions are in the file.

12 Testing the SSL Configuration

Open a browser window and type:

https://server_domain_name
or
https://server_domain_name/hpstmgmt/servlet/MarsDm

13 Configuration Options

The Apache SSL server can be configured to service requests for http on port 80, https or port 443, or both simultaneously.
By default we have chosen to provide instructions to configure the server for only secure SSL (https) operation. However with the Apache configuration provided, the server can be used for normal http connections in addition to the SSL https connections by simply uncommenting the "Port 80" line and the "Listen 80" lines of the httpd.conf configuration file.

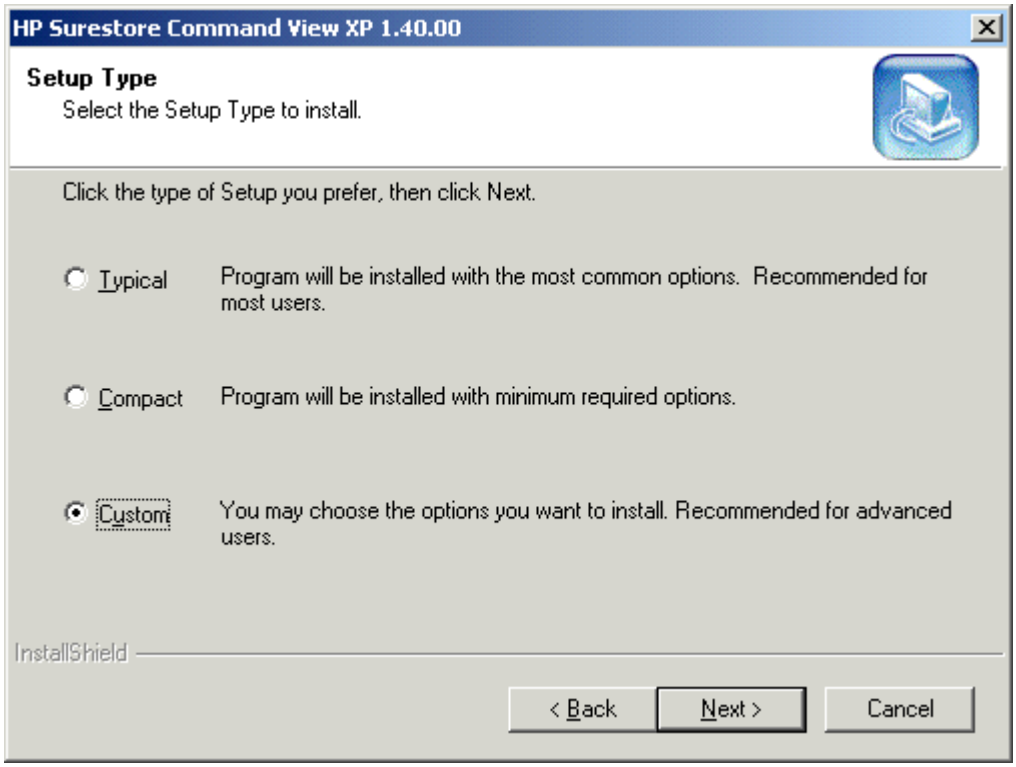
14 References

- OpenSSL – http://www.openssl.org
- Mod_ssl – http://www.modssl.org
- Apache – http://www.apache.org
- Thawte – http://www.thawte.com

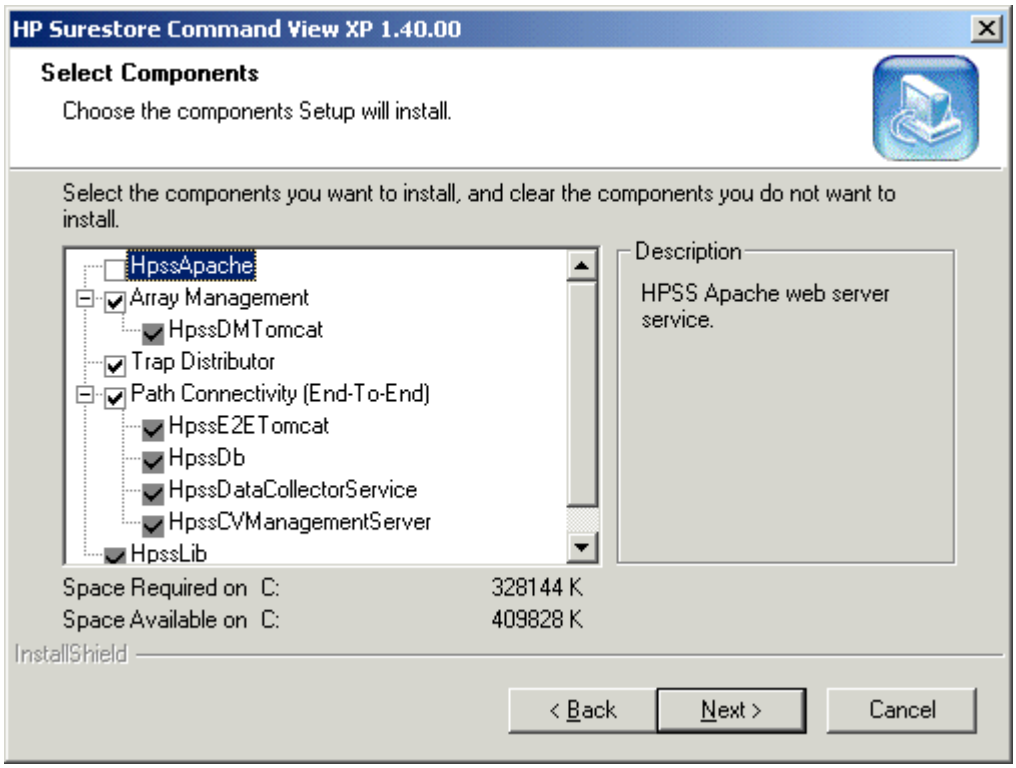
15 Appendixes

15.1 Appendix A – Disabling the Command View Apache Web Server During Installation

If you intend to use Command View with a different Apache web server or Apache SSL web server that has it's own installer (e.g. OpenSA or Covalent SSL), then that web server will provide its own runtime options.
NOTE: If you have already fully installed Command view, see "Appendix B – Disabling the Command View Apache Web Server".
To disable the Command View Apache web server during installation, install the Command View application as follows:



1) Select the "Custom" installation.
2) Press "Next".

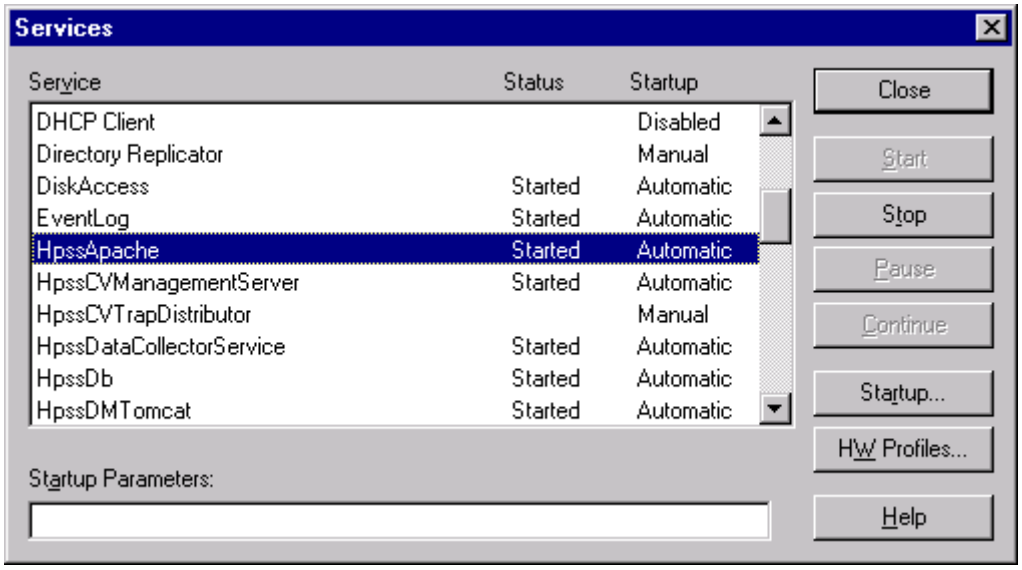


3) Uncheck the “HpssApache” checkbox.
4) Press “Next”.
5) Press “Yes” and continue with the installation.

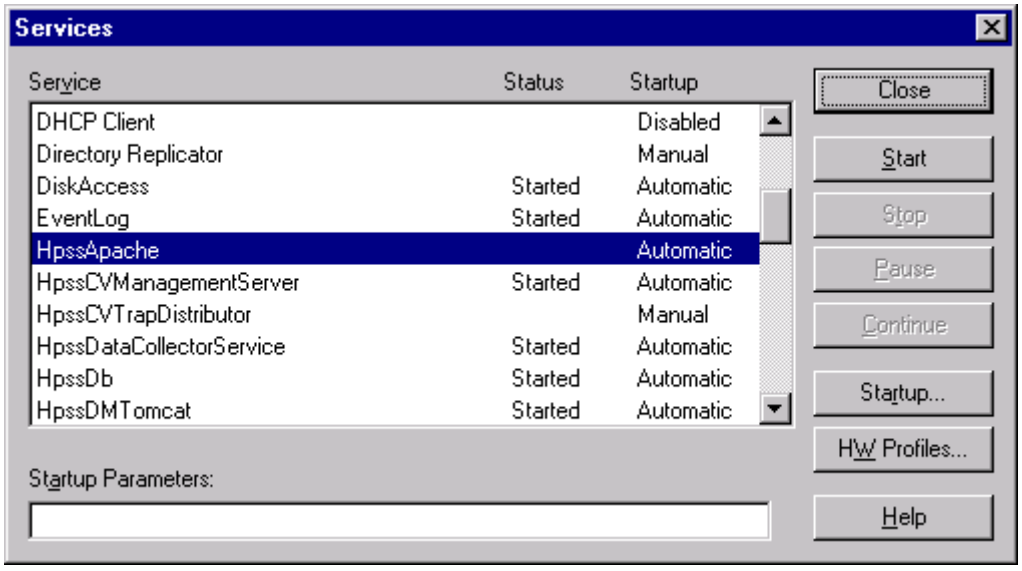
15.2 Appendix B – Disabling the Command View Apache Web Server After Installation

15.2.1 For Windows NT

1) From the start menu: Start->Settings->Control Panel->Services
2) This should bring up the Service control panel.

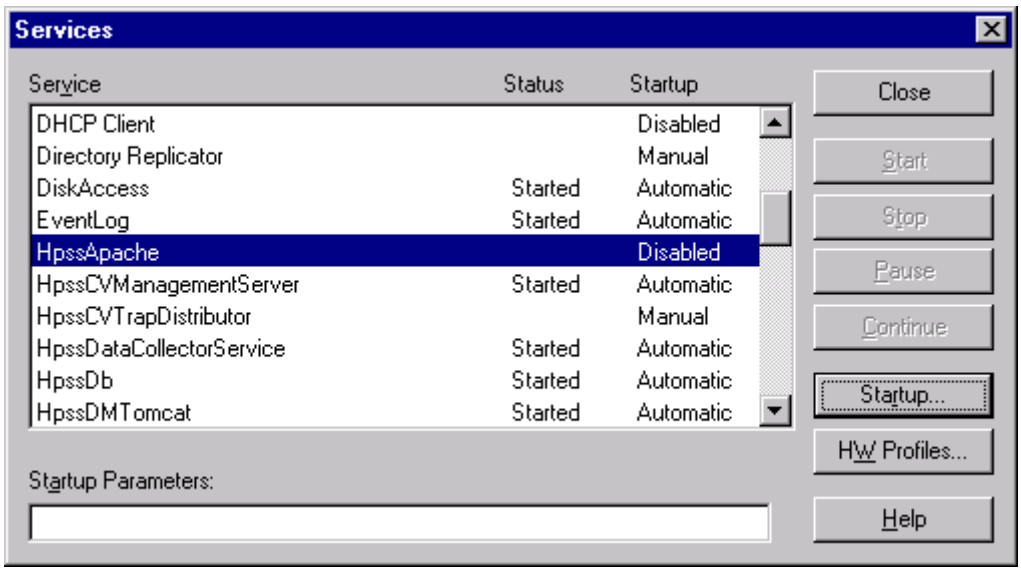


3) Scroll and select the HpssApache service.
4) Stop the HpssApache service by selecting the “Stop” button.
5) Answer “Yes” to the confirmation dialog.



6) Select the “Startup” button to change the “Startup Type”.

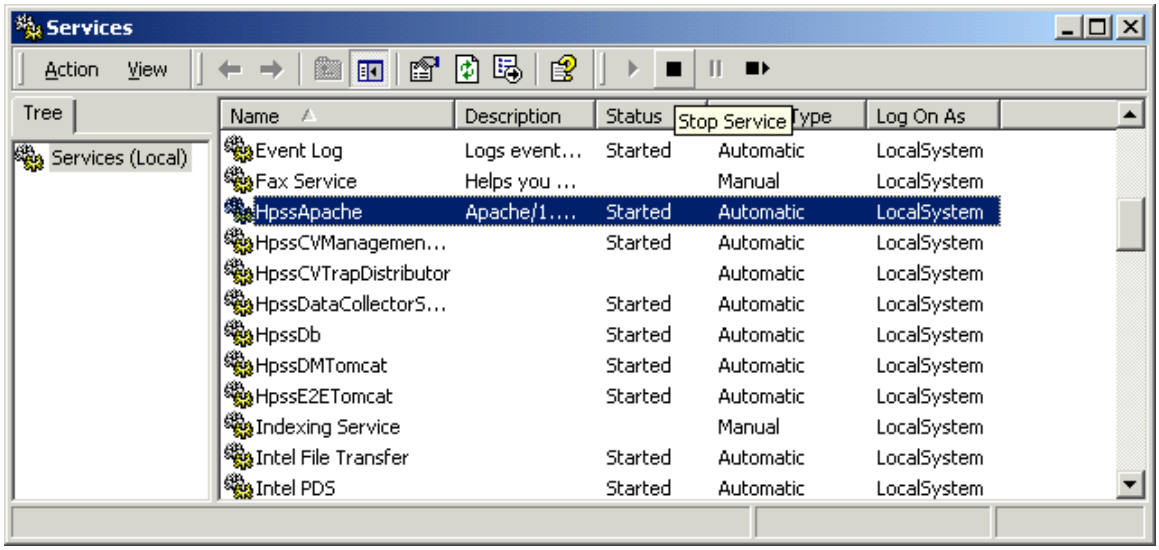
7) Set the “Startup Type” to “Disabled”.
8) Select “OK”.



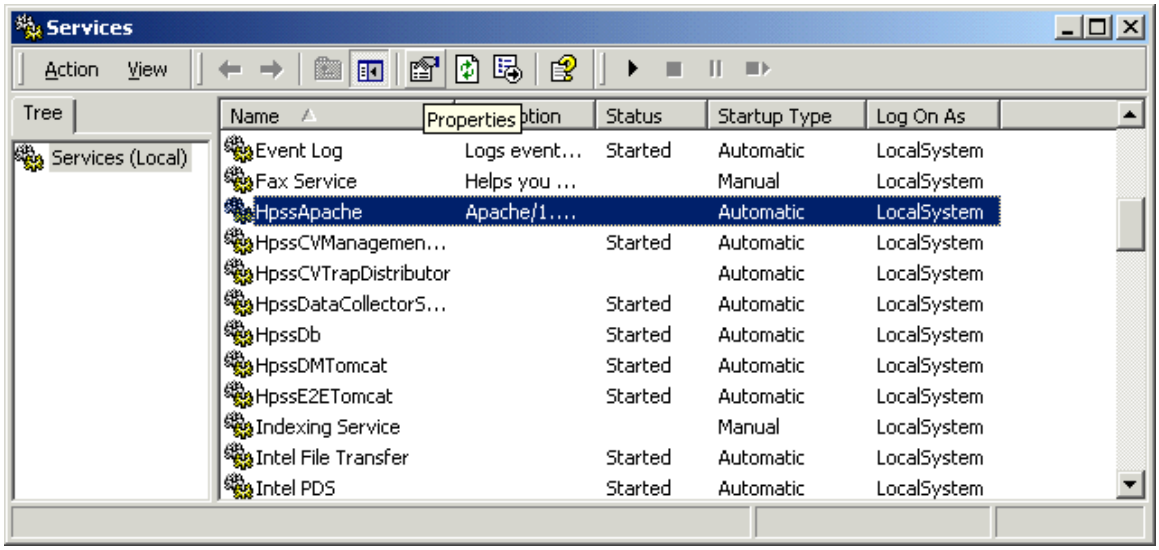
9) The HpssApache service should now be stopped and disabled.

15.2.2 For Windows 2000

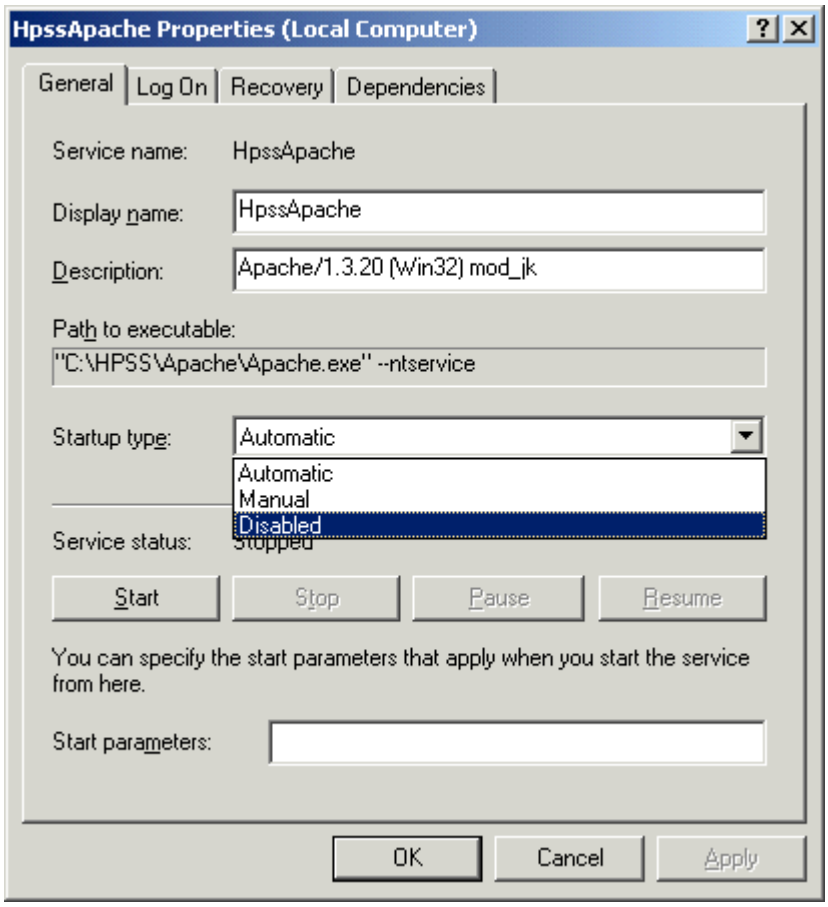
1) From the start menu: Start->Settings->Control Panel->Administrative Tools->Services
This should bring up the Service control panel.



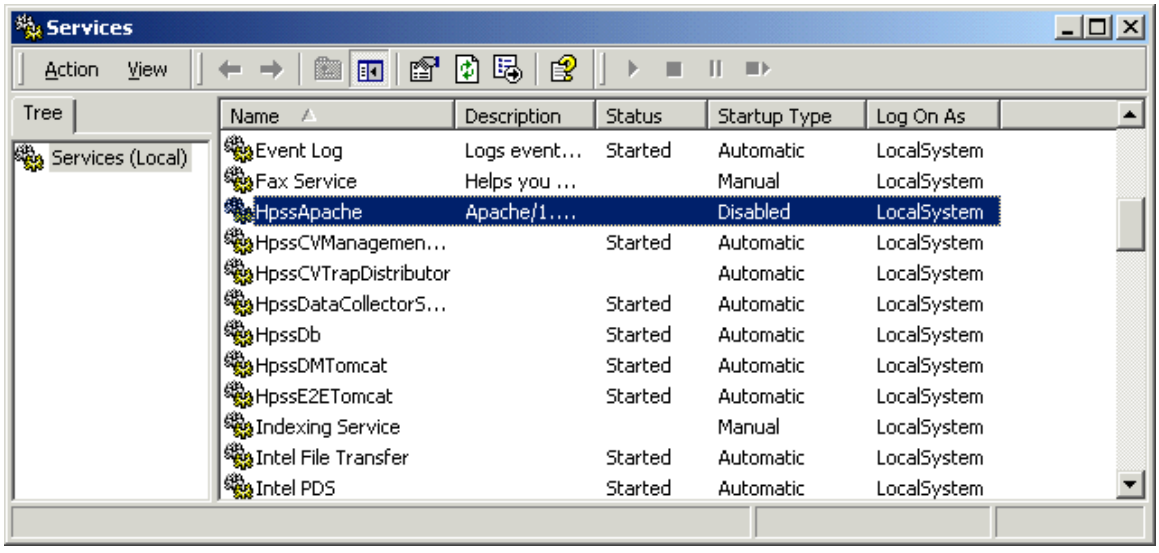
2) Scroll and select the HpssApache service.
3) Stop the HpssApache service by selecting the “Stop” button icon in the tool bar.
4) Answer “Yes” if there is a confirmation dialog.



5) Select the “Properties” button icon in the toolbar.



6) Change the “Startup Type” to “Disabled”.
7) Select “OK”.



8) The HpssApache service should now be stopped and disabled.

15.3 Appendix C - Building and Installing the Apache+mod_ssl+openssl Web Server

For the basic Apache+mod_ssl+openssl web server, it is first necessary to compile the OpenSSL source files. Next, the Apache web server source code must be modified to include the "mod_ssl" module. Finally, the Apache web server must be compiled with the "mod_ssl" module additions to produce the final binary executable.

15.3.1 What You Will Need

- Command View 1.4 or later.
- OpenSSL version 0.9.6b.
- mod_ssl version mod_ssl-2.8.4-1.3.20.
- Apache Web Server version 1.3.20.
- Microsoft Visual C++ 6.0.
- Perl 5.
- Awk.

15.3.1.1 OpenSSL

The OpenSSL distribution is available from: http://www.openssl.org

15.3.1.2 mod_ssl

The mod_ssl distribution is available from: http://www.modssl.org

15.3.1.3 Apache Web Server

The Apache web server is available from: http://www.apache.org

15.3.1.4 Microsoft Visual C++ 6.0

Visual C++ is only commercially available from Microsoft or a reseller.

15.3.1.5 Perl 5

Perl is available from: http://www.activestate.com

15.3.1.6 Awk

Awk is available from: http://cm.bell-labs.com/cm/cs/who/bwk/awk95.exe

15.3.2 Installing the Development Tools

Before you begin, install Microsoft Visual C++, Perl, and Awk.
The awk95.exe must be renamed to “awk.exe” and placed in a directory that is listed in your “path” environment variable.

15.3.3 Compiling and Installing OpenSSL

These instructions are taken from INSTALL.W32 file included with the OpenSSL source and the INSTALL.Win32 included with the mod_ssl source files. The source files are packaged in a tar.gz file. You can use WinZip to extract the source files from this
file. To build and install OpenSSL, do the following:

1) Open a command prompt.
2) Change to the directory that you unpacked the OpenSSL source files to.
3) Run the “configure” perl script to set up the OpenSSL installation by typing: **perl Configure VC-WIN32**
4) Create the make files for Visual C++ by typing: **ms\do_ms**
5) Compile the OpenSSL files by typing: **nmake -f ms\ntdll.mak Note:** If the compilation returns errors, see the INSTALL.W32 file for more information.
6) Change to the out32dll directory by typing: **cd out32dll** Test the compiled files by typing: **..\ms\test**
7) Now install OpenSSL to your system:
   a) Create a directory on the drive of your choice (we will use C: for these instructions) called **openssl**, e.g. **C:\openssl**
   b) Create a directory under **C:\openssl** called **bin**.
   c) Create a directory under **C:\openssl** called **lib**.
   d) Create a directory under **C:\openssl** called **include**.
   e) In the OpenSSL source directory, under the **inc32** directory, is a directory called **openssl**, copy that directory into the **C:\openssl\include** directory.
   f) Copy **ssleay32.lib** and **libeay32.lib** from the **out32dll** directory in the OpenSSL source directory to **C:\openssl\lib**.
   g) Copy **ssleay32.dll**, **libeay32.dll**, and **openssl.exe** from the **out32dll** directory in the OpenSSL source directory to **C:\openssl\bin**.
   h) In the OpenSSL source directory under the **apps** directory copy **openssl.cnf** to the **C:\openssl** directory.
   i) Put **C:\openssl\bin** in your **path** environment variable.

15.3.4 Preparing Apache with mod_ssl

These instructions are taken from the install.w32 file included with the mod_ssl source files. As with OpenSSL, the source files for mod_ssl are packaged in a tar.gz file.

1) Open a command prompt.
2) Go to the mod_ssl directory created when you unpacked the mod_ssl source files by typing: **cd mod_ssl-2.6.6-1.3.20**
3) Prepare the apache make files to compile both mod_ssl and apache with SSL hooks by typing: **configure.bat --with-apache=C:\apache_1.3.20 --with-ssl=C:\openssl** where **C:\apache_1.3.20** is where you unpacked the Apache 1.3.20 source files.

15.3.5 Compiling the Apache+mod_ssl Source

These instructions are taken from the install.w32 file included with the mod_ssl source files.
Please read the Apache\htdocs\manual\win_compiling.html document before beginning.

1) Open a command prompt.
2) Change to the directory you unpacked the Apache source files to.
3) Change directory to the src directory by typing: **cd src**
4) Compile Apache by typing: **nmake /f makefile.win**
5) Group the Apache files into one directory by typing: **nmake /f makefile.win INSTDIR=C:\Apache installr.** This creates a directory on your C: drive called Apache, i.e. "**C:\Apache**".

15.4 Appendix D - Apache Default SSL Configuration

This section contains a listing of a Command View Apache **httpd.conf** file as configured for SSL and with the "Include" directives for all applications shipping at the time of this writing. This listing contains the HPSS\Apache web server configuration directives and the "mod_ssl" SSL directives.
All instances of "@@ServerRoot@@" in the following listing would be replaced with the actual path of the Apache installation directory (using forward slashes for directory separators, e.g. "C:/HPSS/Apache").
All instances of "@@HPSSRoot@@" in the following listing would be replaced with the actual path of the Command View installation directory (using forward slashes for directory separators, e.g. "C:/HPSS").
All instances of "@@ServerName@@" in the following listing would be replaced with the actual fully qualified DNS name of the server, e.g. "hostname.hp.com".
The **httpd.conf** file is located in the "@@ServerRoot@@"**/conf** directory, e.g. "C:/HPSS/Apache/conf".
For SSL operation, an appropriate private key and certificate must be installed. The default locations to install the key and certificate are "@@ServerRoot@@"**/conf/ssl.key** directory and the "@@ServerRoot@@"**/conf/ssl.crt** directory respectively.
The following two directives (in the httpd.conf file) must refer to the key file and the certificate file respectively:

```
SSLCertificateKeyFile "@@ServerRoot@@/conf/ssl.key/server.key"
SSLCertificateFile "@@ServerRoot@@/conf/ssl.crt/server.crt"
```

For instance if you name your key and certificate with your host name as recommended (e.g. hostname.hp.com), then the key and certificate files would be **hostname.hp.com.key** and **hostname.hp.com.crt**.

In this case the directives in the httpd.conf file should read:

```
SSLCertificateKeyFile "@@ServerRoot@@/conf/ssl.key/hostname.hp.com.key"
SSLCertificateFile "@@ServerRoot@@/conf/ssl.crt/hostname.hp.com.crt"
```

httpd.conf

```
Include "@@ServerRoot@@/conf/mod_jk_common.conf"
Include "@@HPSSRoot@@/e2e/tomcat/conf/mod_jk.conf"
Include "@@HPSSRoot@@/dm/tomcat/conf/mod_jk.conf"

# Include the configuration for PA Tomcat
Include "@@HPSSRoot@@/pa/tomcat/conf/mod_jk.conf"
# APM configuration for Tomcat
Include @@HPSSRoot@@\apm\tomcat\conf\tomcat-apm.conf

#
# Based upon the NCSA server configuration files originally by Rob McCool.
#
# This is the main Apache server configuration file.  It contains the
# configuration directives that give the server its instructions.
# See <URL:http://www.apache.org/docs/> for detailed information about
# the directives.
#
# Do NOT simply read the instructions in here without understanding
# what they do.  They're here only as hints or reminders.  If you are unsure
# consult the online docs. You have been warned.
#
# After this file is processed, the server will look for and process
# /Apache/conf/srm.conf and then /Apache/conf/access.conf
# unless you have overridden these with ResourceConfig and/or
# AccessConfig directives here.
#
# The configuration directives are grouped into three basic sections:
#  1. Directives that control the operation of the Apache server process as a
#     whole (the 'global environment').
#  2. Directives that define the parameters of the 'main' or 'default' server,
#     which responds to requests that aren't handled by a virtual host.
#     These directives also provide default values for the settings
```



```
#     of all virtual hosts.
#  3. Settings for virtual hosts, which allow Web requests to be sent to
#     different IP addresses or hostnames and have them handled by the
#     same Apache server process.
#
# Configuration and logfile names: If the filenames you specify for many
# of the server's control files begin with "/" (or "drive:/" for Win32), the
# server will use that explicit path.  If the filenames do *not* begin
# with "/", the value of ServerRoot is prepended -- so "logs/foo.log"
# with ServerRoot set to "/usr/local/apache" will be interpreted by the
# server as "/usr/local/apache/logs/foo.log".
#
# NOTE: Where filenames are specified, you must use forward slashes
# instead of backslashes (e.g., "c:/apache" instead of "c:\apache").
# If a drive letter is omitted, the drive on which Apache.exe is located
# will be used by default.  It is recommended that you always supply
# an explicit drive letter in absolute paths, however, to avoid
# confusion.
#

### Section 1: Global Environment
#
# The directives in this section affect the overall operation of Apache,
# such as the number of concurrent requests it can handle or where it
# can find its configuration files.
#

#
# ServerType is either inetd, or standalone.  Inetd mode is only supported on
# Unix platforms.
#
ServerType standalone

#
# ServerRoot: The top of the directory tree under which the server's
# configuration, error, and log files are kept.
#
ServerRoot "@@ServerRoot@@"

#
# PidFile: The file in which the server should record its process
# identification number when it starts.
#
PidFile logs/httpd.pid

#
# ScoreBoardFile: File used to store internal server process information.
# Not all architectures require this.  But if yours does (you'll know because
# this file will be  created when you run Apache) then you *must* ensure that
# no two invocations of Apache share the same scoreboard file.
#
ScoreBoardFile logs/apache_runtime_status

#
# In the standard configuration, the server will process httpd.conf (this
# file, specified by the -f command line option), srm.conf, and access.conf
# in that order.  The latter two files are now distributed empty, as it is
# recommended that all directives be kept in a single file for simplicity.
# The commented-out values below are the built-in defaults.  You can have the
# server ignore these files altogether by using "/dev/null" (for Unix) or
# "nul" (for Win32) for the arguments to the directives.
#
#ResourceConfig conf/srm.conf
#AccessConfig conf/access.conf

#
# Timeout: The number of seconds before receives and sends time out.
#
Timeout 300

#
```



```
# KeepAlive: Whether or not to allow persistent connections (more than
# one request per connection). Set to "Off" to deactivate.
#
KeepAlive On

#
# MaxKeepAliveRequests: The maximum number of requests to allow
# during a persistent connection. Set to 0 to allow an unlimited amount.
# We recommend you leave this number high, for maximum performance.
#
MaxKeepAliveRequests 100

#
# KeepAliveTimeout: Number of seconds to wait for the next request from the
# same client on the same connection.
#
KeepAliveTimeout 15

#
# Apache on Win32 always creates one child process to handle requests.  If it
# dies, another child process is created automatically.  Within the child
# process multiple threads handle incoming requests.  The next two
# directives control the behaviour of the threads and processes.
#

#
# MaxRequestsPerChild: the number of requests each child process is
# allowed to process before the child dies.  The child will exit so
# as to avoid problems after prolonged use when Apache (and maybe the
# libraries it uses) leak memory or other resources.  On most systems, this
# isn't really needed, but a few (such as Solaris) do have notable leaks
# in the libraries.  For Win32, set this value to zero (unlimited)
# unless advised otherwise.
#
# NOTE: This value does not include keepalive requests after the initial
#       request per connection. For example, if a child process handles
#       an initial request and 10 subsequent "keptalive" requests, it
#       would only count as 1 request towards this limit.
#
MaxRequestsPerChild 0

#
# Number of concurrent threads (i.e., requests) the server will allow.
# Set this value according to the responsiveness of the server (more
# requests active at once means they're all handled more slowly) and
# the amount of system resources you'll allow the server to consume.
#
ThreadsPerChild 50

#
# Listen: Allows you to bind Apache to specific IP addresses and/or
# ports, in addition to the default. See also the <VirtualHost>
# directive.
#
#Listen 3000
#Listen 12.34.56.78:80

#
# BindAddress: You can support virtual hosts with this option. This directive
# is used to tell the server which IP address to listen to. It can either
# contain "*", an IP address, or a fully qualified Internet domain name.
# See also the <VirtualHost> and Listen directives.
#
#BindAddress *

#
# Apache Modules compiled into the standard Windows build
#
# The following modules are bound into the standard Apache binary distribution
# for Windows.  To change the standard behavior, uncomment the following lines
# and modify the list of those specific modules to be enabled in the server.
```



```
#
# WARNING: This is an advanced option that may render your server inoperable!
# Do not use these directives without expert guidance.
#
#ClearModuleList
#AddModule mod_so.c mod_mime.c mod_access.c mod_auth.c mod_negotiation.c
#AddModule mod_include.c mod_autoindex.c mod_dir.c mod_cgi.c mod_userdir.c
#AddModule mod_alias.c mod_env.c mod_log_config.c mod_asis.c mod_imap.c
#AddModule mod_actions.c mod_setenvif.c mod_isapi.c

#
# Dynamic Shared Object (DSO) Support
#
# To be able to use the functionality of a module which was built as a DSO you
# have to place corresponding `LoadModule' lines at this location so the
# directives contained in it are actually available _before_ they are used.
# Please read the file README.DSO in the Apache 1.3 distribution for more
# details about the DSO mechanism and run `apache -l' for the list of already
# built-in (statically linked and thus always available) modules in your Apache
# binary.
#
# Note: The order in which modules are loaded is important.  Don't change
# the order below without expert advice.
#
#LoadModule anon_auth_module modules/mod_auth_anon.so
#LoadModule dbm_auth_module modules/mod_auth_dbm.so
#LoadModule digest_auth_module modules/mod_auth_digest.so
#LoadModule cern_meta_module modules/mod_cern_meta.so
#LoadModule digest_module modules/mod_digest.so
#LoadModule expires_module modules/mod_expires.so
#LoadModule headers_module modules/mod_headers.so
#LoadModule proxy_module modules/mod_proxy.so
#LoadModule rewrite_module modules/mod_rewrite.so
#LoadModule speling_module modules/mod_speling.so
#LoadModule info_module modules/mod_info.so
#LoadModule status_module modules/mod_status.so
#LoadModule usertrack_module modules/mod_usertrack.so
LoadModule ssl_module modules/mod_ssl.so

#
# ExtendedStatus controls whether Apache will generate "full" status
# information (ExtendedStatus On) or just basic information (ExtendedStatus
# Off) when the "server-status" handler is called. The default is Off.
#
#ExtendedStatus On

### Section 2: 'Main' server configuration
#
# The directives in this section set up the values used by the 'main'
# server, which responds to any requests that aren't handled by a
# <VirtualHost> definition.  These values also provide defaults for
# any <VirtualHost> containers you may define later in the file.
#
# All of these directives may appear inside <VirtualHost> containers,
# in which case these default settings will be overridden for the
# virtual host being defined.
#

#
# Port: The port to which the standalone server listens.  Certain firewall
# products must be configured before Apache can listen to a specific port.
# Other running httpd servers will also interfere with this port.  Disable
# all firewall, security, and other services if you encounter problems.
# To help diagnose problems use the Windows NT command NETSTAT -a
#
#Port 80

<IfModule mod_ssl.c>
##
##  SSL Support
##
```



```
##  When we also provide SSL we have to listen to the
##  standard HTTP port (see above) and to the HTTPS port
##
#Listen 80
Listen 443
</IfModule>

#
# ServerAdmin: Your address, where problems with the server should be
# e-mailed.  This address appears on some server-generated pages, such
# as error documents.
#
ServerAdmin you@your.address

#
# ServerName allows you to set a host name which is sent back to clients for
# your server if it's different than the one the program would get (i.e., use
# "www" instead of the host's real name).
#
# Note: You cannot just invent host names and hope they work. The name you
# define here must be a valid DNS name for your host. If you don't understand
# this, ask your network administrator.
# If your host doesn't have a registered DNS name, enter its IP address here.
# You will have to access it by its address (e.g., http://123.45.67.89/)
# anyway, and this will make redirections work in a sensible way.
#
# 127.0.0.1 is the TCP/IP local loop-back address, often named localhost. Your
# machine always knows itself by this address. If you use Apache strictly for
# local testing and development, you may use 127.0.0.1 as the server name.
#
#ServerName new.host.name
ServerName @@ServerName@@

#
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.
#
DocumentRoot "@@ServerRoot@@/htdocs"

#
# Each directory to which Apache has access, can be configured with respect
# to which services and features are allowed and/or disabled in that
# directory (and its subdirectories).
#
# First, we configure the "default" to be a very restrictive set of
# permissions.
#
<Directory />
    Options FollowSymLinks
    AllowOverride AuthConfig
</Directory>

#
# Note that from this point forward you must specifically allow
# particular features to be enabled - so if something's not working as
# you might expect, make sure that you have specifically enabled it
# below.
#

#
# This should be changed to whatever you set DocumentRoot to.
#
<Directory "@@ServerRoot@@/htdocs">

#
# This may also be "None", "All", or any combination of "Indexes",
# "Includes", "FollowSymLinks", "ExecCGI", or "MultiViews".
#
# Note that "MultiViews" must be named *explicitly* --- "Options All"
# doesn't give it to you.
```



```
#
    Options Indexes FollowSymLinks MultiViews

#
# This controls which options the .htaccess files in directories can
# override. Can also be "All", or any combination of "Options", "FileInfo",
# "AuthConfig", and "Limit"
#
    AllowOverride None

#
# Controls who can get stuff from this server.
#
    Order allow,deny
    Allow from all
</Directory>

#
# UserDir: The name of the directory which is appended onto a user's home
# directory if a ~user request is received.
#
# Under Win32, we do not currently try to determine the home directory of
# a Windows login, so a format such as that below needs to be used.  See
# the UserDir documentation for details.
#
<IfModule mod_userdir.c>
UserDir "@@ServerRoot@@/users"
</IfModule>

#
# Control access to UserDir directories.  The following is an example
# for a site where these directories are restricted to read-only.
#
#<Directory "/Apache/users">
#    AllowOverride FileInfo AuthConfig Limit
#    Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
#    <Limit GET POST OPTIONS PROPFIND>
#        Order allow,deny
#        Allow from all
#    </Limit>
#    <LimitExcept GET POST OPTIONS PROPFIND>
#        Order deny,allow
#        Deny from all
#    </LimitExcept>
#</Directory>

#
# DirectoryIndex: Name of the file or files to use as a pre-written HTML
# directory index.  Separate multiple entries with spaces.
#
<IfModule mod_dir.c>
    DirectoryIndex index.html
</IfModule>

#
# AccessFileName: The name of the file to look for in each directory
# for access control information.
#
AccessFileName .htaccess

#
# The following lines prevent .htaccess files from being viewed by
# Web clients.  Since .htaccess files often contain authorization
# information, access is disallowed for security reasons.  Comment
# these lines out if you want Web visitors to see the contents of
# .htaccess files.  If you change the AccessFileName directive above,
# be sure to make the corresponding changes here.
#
# Also, folks tend to use names such as .htpasswd for password
# files, so this will protect those as well.
#
```



```
<Files ~ "^\.ht">
    Order allow,deny
    Deny from all
</Files>

#
# CacheNegotiatedDocs: By default, Apache sends "Pragma: no-cache" with each
# document that was negotiated on the basis of content. This asks proxy
# servers not to cache the document. Uncommenting the following line disables
# this behavior, and proxies will be allowed to cache the documents.
#
#CacheNegotiatedDocs

#
# UseCanonicalName:  (new for 1.3)  With this setting turned on, whenever
# Apache needs to construct a self-referencing URL (a URL that refers back
# to the server the response is coming from) it will use ServerName and
# Port to form a "canonical" name.  With this setting off, Apache will
# use the hostname:port that the client supplied, when possible.  This
# also affects SERVER_NAME and SERVER_PORT in CGI scripts.
#
UseCanonicalName On

#
# TypesConfig describes where the mime.types file (or equivalent) is
# to be found.
#
<IfModule mod_mime.c>
    TypesConfig conf/mime.types
</IfModule>

#
# DefaultType is the default MIME type the server will use for a document
# if it cannot otherwise determine one, such as from filename extensions.
# If your server contains mostly text or HTML documents, "text/plain" is
# a good value.  If most of your content is binary, such as applications
# or images, you may want to use "application/octet-stream" instead to
# keep browsers from trying to display binary files as though they are
# text.
#
DefaultType text/plain

#
# The mod_mime_magic module allows the server to use various hints from the
# contents of the file itself to determine its type.  The MIMEMagicFile
# directive tells the module where the hint definitions are located.
# mod_mime_magic is not part of the default server (you have to add
# it yourself with a LoadModule [see the DSO paragraph in the 'Global
# Environment' section], or recompile the server and include mod_mime_magic
# as part of the configuration), so it's enclosed in an <IfModule> container.
# This means that the MIMEMagicFile directive will only be processed if the
# module is part of the server.
#
<IfModule mod_mime_magic.c>
    MIMEMagicFile conf/magic
</IfModule>

#
# HostnameLookups: Log the names of clients or just their IP addresses
# e.g., www.apache.org (on) or 204.62.129.132 (off).
# The default is off because it'd be overall better for the net if people
# had to knowingly turn this feature on, since enabling it means that
# each client request will result in AT LEAST one lookup request to the
# nameserver.
#
HostnameLookups Off

#
# ErrorLog: The location of the error log file.
# If you do not specify an ErrorLog directive within a <VirtualHost>
# container, error messages relating to that virtual host will be
```



```
# logged here.  If you *do* define an error logfile for a <VirtualHost>
# container, that host's errors will be logged there and not here.
#
ErrorLog logs/error.log

#
# LogLevel: Control the number of messages logged to the error.log.
# Possible values include: debug, info, notice, warn, error, crit,
# alert, emerg.
#
LogLevel crit

#
# The following directives define some format nicknames for use with
# a CustomLog directive (see below).
#
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent

#
# The location and format of the access logfile (Common Logfile Format).
# If you do not define any access logfiles within a <VirtualHost>
# container, they will be logged here.  Contrariwise, if you *do*
# define per-<VirtualHost> access logfiles, transactions will be
# logged therein and *not* in this file.
#
#CustomLog logs/access.log common

#
# If you would like to have agent and referer logfiles, uncomment the
# following directives.
#
#CustomLog logs/referer.log referer
#CustomLog logs/agent.log agent

#
# If you prefer a single logfile with access, agent, and referer information
# (Combined Logfile Format) you can use the following directive.
#
#CustomLog logs/access.log combined

#
# Optionally add a line containing the server version and virtual host
# name to server-generated pages (error documents, FTP directory listings,
# mod_status and mod_info output etc., but not CGI generated documents).
# Set to "EMail" to also include a mailto: link to the ServerAdmin.
# Set to one of:  On | Off | EMail
#
ServerSignature On

#
# Apache parses all CGI scripts for the shebang line by default.
# This comment line, the first line of the script, consists of the symbols
# pound (#) and exclamation (!) followed by the path of the program that
# can execute this specific script.  For a perl script, with perl.exe in
# the C:\Program Files\Perl directory, the shebang line should be:

   #!c:/program files/perl/perl

# Note you _must_not_ indent the actual shebang line, and it must be the
# first line of the file.  Of course, CGI processing must be enabled by
# the appropriate ScriptAlias or Options ExecCGI directives for the files
# or directory in question.
#
# However, Apache on Windows allows either the Unix behavior above, or can
# use the Registry to match files by extention.  The command to execute
# a file of this type is retrieved from the registry by the same method as
# the Windows Explorer would use to handle double-clicking on a file.
# These script actions can be configured from the Windows Explorer View menu,
```



```
# 'Folder Options', and reviewing the 'File Types' tab.  Clicking the Edit
# button allows you to modify the Actions, of which Apache 1.3 attempts to
# perform the 'Open' Action, and failing that it will try the shebang line.
# This behavior is subject to change in Apache release 2.0.
#
# Each mechanism has it's own specific security weaknesses, from the means
# to run a program you didn't intend the website owner to invoke, and the
# best method is a matter of great debate.
#
# To enable the this Windows specific behavior (and therefore -disable- the
# equivilant Unix behavior), uncomment the following directive:
#
#ScriptInterpreterSource registry
#
# The directive above can be placed in individual <Directory> blocks or the
# .htaccess file, with either the 'registry' (Windows behavior) or 'script'
# (Unix behavior) option, and will override this server default option.
#

#
# Aliases: Add here as many aliases as you need (with no limit). The format is
# Alias fakename realname
#
<IfModule mod_alias.c>

    #
    # Note that if you include a trailing / on fakename then the server will
    # require it to be present in the URL.  So "/icons" isn't aliased in this
    # example, only "/icons/".  If the fakename is slash-terminated, then the
    # realname must also be slash terminated, and if the fakename omits the
    # trailing slash, the realname must also omit it.
    #
Alias /icons/ "@@ServerRoot@@/icons/"

<Directory "@@ServerRoot@@/icons">
        Options Indexes MultiViews
        AllowOverride None
        Order allow,deny
        Allow from all
    </Directory>

    #
    # ScriptAlias: This controls which directories contain server scripts.
    # ScriptAliases are essentially the same as Aliases, except that
    # documents in the realname directory are treated as applications and
    # run by the server when requested rather than as documents sent to the client.
    # The same rules about trailing "/" apply to ScriptAlias directives as to
    # Alias.
    #
ScriptAlias /cgi-bin/ "@@ServerRoot@@/cgi-bin/"

    #
    # "/Apache/cgi-bin" should be changed to whatever your ScriptAliased
    # CGI directory exists, if you have that configured.
    #
<Directory "@@ServerRoot@@/cgi-bin">
        AllowOverride None
        Options None
        Order allow,deny
        Allow from all
    </Directory>

</IfModule>
# End of aliases.

#
# Redirect allows you to tell clients about documents which used to exist in
# your server's namespace, but do not anymore. This allows you to tell the
# clients where to look for the relocated document.
# Format: Redirect old-URI new-URL
#
```



```
#
# Directives controlling the display of server-generated directory listings.
#
<IfModule mod_autoindex.c>

    #
    # FancyIndexing is whether you want fancy directory indexing or standard
    #
    # Note, add the option TrackModified to the IndexOptions default list only
    # if all indexed directories reside on NTFS volumes.  The TrackModified flag
    # will report the Last-Modified date to assist caches and proxies to properly
    # track directory changes, but it does _not_ work on FAT volumes.
    #
    IndexOptions FancyIndexing

    #
    # AddIcon* directives tell the server which icon to show for different
    # files or filename extensions.  These are only displayed for
    # FancyIndexed directories.
    #
    AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip

    AddIconByType (TXT,/icons/text.gif) text/*
    AddIconByType (IMG,/icons/image2.gif) image/*
    AddIconByType (SND,/icons/sound2.gif) audio/*
    AddIconByType (VID,/icons/movie.gif) video/*

    AddIcon /icons/binary.gif .bin .exe
    AddIcon /icons/binhex.gif .hqx
    AddIcon /icons/tar.gif .tar
    AddIcon /icons/world2.gif .wrl .wrl.gz .vrml .vrm .iv
    AddIcon /icons/compressed.gif .Z .z .tgz .gz .zip
    AddIcon /icons/a.gif .ps .ai .eps
    AddIcon /icons/layout.gif .html .shtml .htm .pdf
    AddIcon /icons/text.gif .txt
    AddIcon /icons/c.gif .c
    AddIcon /icons/p.gif .pl .py
    AddIcon /icons/f.gif .for
    AddIcon /icons/dvi.gif .dvi
    AddIcon /icons/uuencoded.gif .uu
    AddIcon /icons/script.gif .conf .sh .shar .csh .ksh .tcl
    AddIcon /icons/tex.gif .tex
    AddIcon /icons/bomb.gif core

    AddIcon /icons/back.gif ..
    AddIcon /icons/hand.right.gif README
    AddIcon /icons/folder.gif ^^DIRECTORY^^
    AddIcon /icons/blank.gif ^^BLANKICON^^

    #
    # DefaultIcon is which icon to show for files which do not have an icon
    # explicitly set.
    #
    DefaultIcon /icons/unknown.gif

    #
    # AddDescription allows you to place a short description after a file in
    # server-generated indexes.  These are only displayed for FancyIndexed
    # directories.
    # Format: AddDescription "description" filename
    #
    #AddDescription "GZIP compressed document" .gz
    #AddDescription "tar archive" .tar
    #AddDescription "GZIP compressed tar archive" .tgz

    #
    # ReadmeName is the name of the README file the server will look for by
    # default, and append to directory listings.
    #
    # HeaderName is the name of a file which should be prepended to
```



```
    # directory indexes.
    #
    # If MultiViews are amongst the Options in effect, the server will
    # first look for name.html and include it if found.  If name.html
    # doesn't exist, the server will then look for name.txt and include
    # it as plaintext if found.
    #
    ReadmeName README
    HeaderName HEADER

    #
    # IndexIgnore is a set of filenames which directory indexing should ignore
    # and not include in the listing.  Shell-style wildcarding is permitted.
    #
    IndexIgnore .??* *~ *# HEADER* README* RCS CVS *,v *,t

</IfModule>
# End of indexing directives.

#
# Document types.
#
<IfModule mod_mime.c>

    #
    # AddEncoding allows you to have certain browsers (Mosaic/X 2.1+) uncompress
    # information on the fly. Note: Not all browsers support this.
    # Despite the name similarity, the following Add* directives have nothing
    # to do with the FancyIndexing customization directives above.
    #
    AddEncoding x-compress Z
    AddEncoding x-gzip gz tgz
    #
    # AddLanguage allows you to specify the language of a document. You can
    # then use content negotiation to give a browser a file in a language
    # it can understand.
    #
    # Note 1: The suffix does not have to be the same as the language
    # keyword --- those with documents in Polish (whose net-standard
    # language code is pl) may wish to use "AddLanguage pl .po" to
    # avoid the ambiguity with the common suffix for perl scripts.
    #
    # Note 2: The example entries below illustrate that in quite
    # some cases the two character 'Language' abbreviation is not
    # identical to the two character 'Country' code for its country,
    # E.g. 'Danmark/dk' versus 'Danish/da'.
    #
    # Note 3: In the case of 'ltz' we violate the RFC by using a three char
    # specifier. But there is 'work in progress' to fix this and get
    # the reference data for rfc1766 cleaned up.
    #
    # Danish (da) - Dutch (nl) - English (en) - Estonian (ee)
    # French (fr) - German (de) - Greek-Modern (el)
    # Italian (it) - Korean (kr) - Norwegian (no)
    # Portugese (pt) - Luxembourgeois* (ltz)
    # Spanish (es) - Swedish (sv) - Catalan (ca) - Czech(cz)
    # Polish (pl) - Brazilian Portuguese (pt-br) - Japanese (ja)
    # Russian (ru)
    #
    AddLanguage da .dk
    AddLanguage nl .nl
    AddLanguage en .en
    AddLanguage et .ee
    AddLanguage fr .fr
    AddLanguage de .de
    AddLanguage el .el
    AddLanguage he .he
    AddCharset ISO-8859-8 .iso8859-8
    AddLanguage it .it
    AddLanguage ja .ja
    AddCharset ISO-2022-JP .jis
```



```
AddLanguage kr .kr
AddCharset ISO-2022-KR .iso-kr
AddLanguage no .no
AddLanguage pl .po
AddCharset ISO-8859-2 .iso-pl
AddLanguage pt .pt
AddLanguage pt-br .pt-br
AddLanguage ltz .lu
AddLanguage ca .ca
AddLanguage es .es
AddLanguage sv .se
AddLanguage cz .cz
AddLanguage ru .ru
AddLanguage tw .tw
AddLanguage zh-tw .tw
AddCharset Big5         .Big5    .big5
AddCharset WINDOWS-1251 .cp-1251
AddCharset CP866        .cp866
AddCharset ISO-8859-5   .iso-ru
AddCharset KOI8-R       .koi8-r
AddCharset UCS-2        .ucs2
AddCharset UCS-4        .ucs4
AddCharset UTF-8        .utf8

# LanguagePriority allows you to give precedence to some languages
# in case of a tie during content negotiation.
#
# Just list the languages in decreasing order of preference. We have
# more or less alphabetized them here. You probably want to change this.
#
<IfModule mod_negotiation.c>
    LanguagePriority en da nl et fr de el it ja kr no pl pt pt-br ru ltz ca es sv tw
</IfModule>

#
# AddType allows you to tweak mime.types without actually editing it, or to
# make certain files to be certain types.
#
# For example, the PHP 3.x module (not part of the Apache distribution - see
# http://www.php.net) will typically use:
#
#AddType application/x-httpd-php3 .php3
#AddType application/x-httpd-php3-source .phps
#
# And for PHP 4.x, use:
#
#AddType application/x-httpd-php .php
#AddType application/x-httpd-php-source .phps

AddType application/x-tar .tgz

#
# AddHandler allows you to map certain file extensions to "handlers",
# actions unrelated to filetype. These can be either built into the server
# or added with the Action command (see below)
#
# If you want to use server side includes, or CGI outside
# ScriptAliased directories, uncomment the following lines.
#
# To use CGI scripts:
#
#AddHandler cgi-script .cgi

#
# To use server-parsed HTML files
#
#AddType text/html .shtml
#AddHandler server-parsed .shtml

#
# Uncomment the following line to enable Apache's send-asis HTTP file
```



```
    # feature
    #
    #AddHandler send-as-is asis

    #
    # If you wish to use server-parsed imagemap files, use
    #
    #AddHandler imap-file map

    #
    # To enable type maps, you might want to use
    #
    #AddHandler type-map var

</IfModule>
# End of document types.

#
# Action lets you define media types that will execute a script whenever
# a matching file is called. This eliminates the need for repeated URL
# pathnames for oft-used CGI file processors.
# Format: Action media/type /cgi-script/location
# Format: Action handler-name /cgi-script/location
#

#
# MetaDir: specifies the name of the directory in which Apache can find
# meta information files. These files contain additional HTTP headers
# to include when sending the document
#
#MetaDir .web

#
# MetaSuffix: specifies the file name suffix for the file containing the
# meta information.
#
#MetaSuffix .meta

#
# Customizable error response (Apache style)
#  these come in three flavors
#
#    1) plain text
#ErrorDocument 500 "The server made a boo boo.
#  n.b.  the single leading (") marks it as text, it does not get output
#
#    2) local redirects
#ErrorDocument 404 /missing.html
#  to redirect to local URL /missing.html
#ErrorDocument 404 /cgi-bin/missing_handler.pl
#  N.B.: You can redirect to a script or a document using server-side-includes.
#
#    3) external redirects
#ErrorDocument 402 http://some.other_server.com/subscription_info.html
#  N.B.: Many of the environment variables associated with the original
#  request will *not* be available to such a script.

#
# Customize behaviour based on the browser
#
<IfModule mod_setenvif.c>

    #
    # The following directives modify normal HTTP response behavior.
    # The first directive disables keepalive for Netscape 2.x and browsers that
    # spoof it. There are known problems with these browser implementations.
    # The second directive is for Microsoft Internet Explorer 4.0b2
    # which has a broken HTTP/1.1 implementation and does not properly
    # support keepalive when it is used on 301 or 302 (redirect) responses.
    #
    BrowserMatch "Mozilla/2" nokeepalive
```



```
    BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0 force-response-1.0

    #
    # The following directive disables HTTP/1.1 responses to browsers which
    # are in violation of the HTTP/1.0 spec by not being able to grok a
    # basic 1.1 response.
    #
    BrowserMatch "RealPlayer 4\.0" force-response-1.0
    BrowserMatch "Java/1\.0" force-response-1.0
    BrowserMatch "JDK/1\.0" force-response-1.0

</IfModule>
# End of browser customization directives

#
# Allow server status reports, with the URL of http://servername/server-status
# Change the ".your_domain.com" to match your domain to enable.
#
#<Location /server-status>
#    SetHandler server-status
#    Order deny,allow
#    Deny from all
#    Allow from .your_domain.com
#</Location>

#
# Allow remote server configuration reports, with the URL of
# http://servername/server-info (requires that mod_info.c be loaded).
# Change the ".your_domain.com" to match your domain to enable.
#
#<Location /server-info>
#    SetHandler server-info
#    Order deny,allow
#    Deny from all
#    Allow from .your_domain.com
#</Location>

#
# There have been reports of people trying to abuse an old bug from pre-1.1
# days.  This bug involved a CGI script distributed as a part of Apache.
# By uncommenting these lines you can redirect these attacks to a logging
# script on phf.apache.org.  Or, you can record them yourself, using the script
# support/phf_abuse_log.cgi.
#
#<Location /cgi-bin/phf*>
#    Deny from all
#    ErrorDocument 403 http://phf.apache.org/phf_abuse_log.cgi
#</Location>

#
# Proxy Server directives. Uncomment the following lines to
# enable the proxy server:
#
#<IfModule mod_proxy.c>
#    ProxyRequests On

#    <Directory proxy:*>
#        Order deny,allow
#        Deny from all
#        Allow from .your_domain.com
#    </Directory>

    #
    # Enable/disable the handling of HTTP/1.1 "Via:" headers.
    # ("Full" adds the server version; "Block" removes all outgoing Via: headers)
    # Set to one of: Off | On | Full | Block
    #
#    ProxyVia On

    #
    # To enable the cache as well, edit and uncomment the following lines:
```



```
    # (no cacheing without CacheRoot)
    #
#    CacheRoot "/Apache/proxy"
#    CacheSize 5
#    CacheGcInterval 4
#    CacheMaxExpire 24
#    CacheLastModifiedFactor 0.1
#    CacheDefaultExpire 1
#    NoCache a_domain.com another_domain.edu joes.garage_sale.com

#</IfModule>
# End of proxy directives.

### Section 3: Virtual Hosts
#
# VirtualHost: If you want to maintain multiple domains/hostnames on your
# machine you can setup VirtualHost containers for them. Most configurations
# use only name-based virtual hosts so the server doesn't need to worry about
# IP addresses. This is indicated by the asterisks in the directives below.
#
# Please see the documentation at <URL:http://www.apache.org/docs/vhosts/>
# for further details before you try to setup virtual hosts.
#
# You may use the command line option '-S' to verify your virtual host
# configuration.

#
# Use name-based virtual hosting.
#
#NameVirtualHost *

#
# VirtualHost example:
# Almost any Apache directive may go into a VirtualHost container.
# The first VirtualHost section is used for requests without a known
# server name.
#
#<VirtualHost *>
#    ServerAdmin webmaster@dummy-host.example.com
#    DocumentRoot /www/docs/dummy-host.example.com
#    ServerName dummy-host.example.com
#    ErrorLog logs/dummy-host.example.com-error_log
#    CustomLog logs/dummy-host.example.com-access_log common
#</VirtualHost>

<IfModule mod_ssl.c>
##
##  SSL Global Context
##
##  All SSL configuration in this context applies both to
##  the main server and all SSL-enabled virtual hosts.
##

#
#   Some MIME-types for downloading Certificates and CRLs
#
AddType application/x-x509-ca-cert .crt
AddType application/x-pkcs7-crl    .crl

#   Pass Phrase Dialog:
#   Configure the pass phrase gathering process.
#   The filtering dialog program (`builtin' is a internal
#   terminal dialog) has to provide the pass phrase on stdout.
SSLPassPhraseDialog  builtin

#   Inter-Process Session Cache:
#   Configure the SSL Session Cache: First either `none'
#   or `dbm:/path/to/file' for the mechanism to use and
#   second the expiring timeout (in seconds).
#SSLSessionCache        none
#SSLSessionCache        shm:logs/ssl_scache(512000)
```



```
SSLSessionCache         dbm:logs/ssl_scache
SSLSessionCacheTimeout  300

#   Semaphore:
#   Configure the path to the mutual exclusion semaphore the
#   SSL engine uses internally for inter-process synchronization.
#SSLMutex  file:logs/ssl_mutex
SSLMutex  sem

#   Pseudo Random Number Generator (PRNG):
#   Configure one or more sources to seed the PRNG of the
#   SSL library. The seed data should be of good random quality.
#   WARNING! On some platforms /dev/random blocks if not enough entropy
#   is available. This means you then cannot use the /dev/random device
#   because it would lead to very long connection times (as long as
#   it requires to make more entropy available). But usually those
#   platforms additionally provide a /dev/urandom device which doesn't
#   block. So, if available, use this one instead. Read the mod_ssl User
#   Manual for more details.
SSLRandomSeed startup builtin
SSLRandomSeed connect builtin
#SSLRandomSeed startup file:/dev/random  512
#SSLRandomSeed startup file:/dev/urandom 512
#SSLRandomSeed connect file:/dev/random  512
#SSLRandomSeed connect file:/dev/urandom 512

#   Logging:
#   The home of the dedicated SSL protocol logfile. Errors are
#   additionally duplicated in the general error log file.  Put
#   this somewhere where it cannot be used for symlink attacks on
#   a real server (i.e. somewhere where only root can write).
#   Log levels are (ascending order: higher ones include lower ones):
#   none, error, warn, info, trace, debug.
#SSLLog      logs/ssl_engine_log
#SSLLogLevel info

##
## SSL Virtual Host Context
##

<VirtualHost _default_:443>

#  General setup for the virtual host
DocumentRoot "@@ServerRoot@@/htdocs"
ServerName @@ServerName@@
ServerAdmin webmaster@localhost
#ErrorLog logs/error_log
#TransferLog logs/access_log

#   SSL Engine Switch:
#   Enable/Disable SSL for this virtual host.
SSLEngine on

#   SSL Cipher Suite:
#   List the ciphers that the client is permitted to negotiate.
#   See the mod_ssl documentation for a complete list.
SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL

#   Server Certificate:
#   Point SSLCertificateFile at a PEM encoded certificate.  If
#   the certificate is encrypted, then you will be prompted for a
#   pass phrase.  Note that a kill -HUP will prompt again. A test
#   certificate can be generated with `make certificate' under
#   built time. Keep in mind that if you've both a RSA and a DSA
#   certificate you can configure both in parallel (to also allow
#   the use of DSA ciphers, etc.)
SSLCertificateFile "@@ServerRoot@@/conf/ssl.crt/server.crt"
#SSLCertificateFile "/Apache/conf/ssl.crt/server-dsa.crt"

#   Server Private Key:
#   If the key is not combined with the certificate, use this
```



```
#   directive to point at the key file.  Keep in mind that if
#   you've both a RSA and a DSA private key you can configure
#   both in parallel (to also allow the use of DSA ciphers, etc.)
SSLCertificateKeyFile "@@ServerRoot@@/conf/ssl.key/server.key"
#SSLCertificateKeyFile "/Apache/conf/ssl.key/server-dsa.key"

#   Server Certificate Chain:
#   Point SSLCertificateChainFile at a file containing the
#   concatenation of PEM encoded CA certificates which form the
#   certificate chain for the server certificate. Alternatively
#   the referenced file can be the same as SSLCertificateFile
#   when the CA certificates are directly appended to the server
#   certificate for convinience.
#SSLCertificateChainFile "/Apache/conf/ssl.crt/ca.crt"

#   Certificate Authority (CA):
#   Set the CA certificate verification path where to find CA
#   certificates for client authentication or alternatively one
#   huge file containing all of them (file must be PEM encoded)
#   Note: Inside SSLCACertificatePath you need hash symlinks
#         to point to the certificate files. Use the provided
#         Makefile to update the hash symlinks after changes.
#SSLCACertificatePath "/Apache/conf/ssl.crt"
#SSLCACertificateFile "/Apache/conf/ssl.crt/ca-bundle.crt"

#   Certificate Revocation Lists (CRL):
#   Set the CA revocation path where to find CA CRLs for client
#   authentication or alternatively one huge file containing all
#   of them (file must be PEM encoded)
#   Note: Inside SSLCARevocationPath you need hash symlinks
#         to point to the certificate files. Use the provided
#         Makefile to update the hash symlinks after changes.
#SSLCARevocationPath "/Apache/conf/ssl.crl"
#SSLCARevocationFile "/Apache/conf/ssl.crl/ca-bundle.crl"

#   Client Authentication (Type):
#   Client certificate verification type and depth.  Types are
#   none, optional, require and optional_no_ca.  Depth is a
#   number which specifies how deeply to verify the certificate
#   issuer chain before deciding the certificate is not valid.
#SSLVerifyClient require
#SSLVerifyDepth  10

#   Access Control:
#   With SSLRequire you can do per-directory access control based
#   on arbitrary complex boolean expressions containing server
#   variable checks and other lookup directives.  The syntax is a
#   mixture between C and Perl.  See the mod_ssl documentation
#   for more details.
#<Location />
#SSLRequire (    %{SSL_CIPHER} !~ m/^(EXP|NULL)/ \
#            and %{SSL_CLIENT_S_DN_O} eq "Snake Oil, Ltd." \
#            and %{SSL_CLIENT_S_DN_OU} in {"Staff", "CA", "Dev"} \
#            and %{TIME_WDAY} >= 1 and %{TIME_WDAY} <= 5 \
#            and %{TIME_HOUR} >= 8 and %{TIME_HOUR} <= 20       ) \
#           or %{REMOTE_ADDR} =~ m/^192\.76\.162\.[0-9]+$/
#</Location>

#   SSL Engine Options:
#   Set various options for the SSL engine.
#   o FakeBasicAuth:
#     Translate the client X.509 into a Basic Authorisation.  This means that
#     the standard Auth/DBMAuth methods can be used for access control.  The
#     user name is the `one line' version of the client's X.509 certificate.
#     Note that no password is obtained from the user. Every entry in the user
#     file needs this password: `xxj31ZMTZzkVA'.
#   o ExportCertData:
#     This exports two additional environment variables: SSL_CLIENT_CERT and
#     SSL_SERVER_CERT. These contain the PEM-encoded certificates of the
#     server (always existing) and the client (only existing when client
#     authentication is used). This can be used to import the certificates
```



```
#     into CGI scripts.
#   o StdEnvVars:
#     This exports the standard SSL/TLS related `SSL_*' environment variables.
#     Per default this exportation is switched off for performance reasons,
#     because the extraction step is an expensive operation and is usually
#     useless for serving static content. So one usually enables the
#     exportation for CGI and SSI requests only.
#   o CompatEnvVars:
#     This exports obsolete environment variables for backward compatibility
#     to Apache-SSL 1.x, mod_ssl 2.0.x, Sioux 1.0 and Stronghold 2.x. Use this
#     to provide compatibility to existing CGI scripts.
#   o StrictRequire:
#     This denies access when "SSLRequireSSL" or "SSLRequire" applied even
#     under a "Satisfy any" situation, i.e. when it applies access is denied
#     and no other module can change it.
#   o OptRenegotiate:
#     This enables optimized SSL connection renegotiation handling when SSL
#     directives are used in per-directory context.
#SSLOptions +FakeBasicAuth +ExportCertData +CompatEnvVars +StrictRequire
<Files ~ "\.(cgi|shtml|phtml|php3?)$">
    SSLOptions +StdEnvVars
</Files>
<Directory "@@ServerRoot@@/cgi-bin">
    SSLOptions +StdEnvVars
</Directory>

#   SSL Protocol Adjustments:
#   The safe and default but still SSL/TLS standard compliant shutdown
#   approach is that mod_ssl sends the close notify alert but doesn't wait for
#   the close notify alert from client. When you need a different shutdown
#   approach you can use one of the following variables:
#   o ssl-unclean-shutdown:
#     This forces an unclean shutdown when the connection is closed, i.e. no
#     SSL close notify alert is send or allowed to received.  This violates
#     the SSL/TLS standard but is needed for some brain-dead browsers. Use
#     this when you receive I/O errors because of the standard approach where
#     mod_ssl sends the close notify alert.
#   o ssl-accurate-shutdown:
#     This forces an accurate shutdown when the connection is closed, i.e. a
#     SSL close notify alert is send and mod_ssl waits for the close notify
#     alert of the client. This is 100% SSL/TLS standard compliant, but in
#     practice often causes hanging connections with brain-dead browsers. Use
#     this only for browsers where you know that their SSL implementation
#     works correctly.
#   Notice: Most problems of broken clients are also related to the HTTP
#   keep-alive facility, so you usually additionally want to disable
#   keep-alive for those clients, too. Use variable "nokeepalive" for this.
#   Similarly, one has to force some clients to use HTTP/1.0 to workaround
#   their broken HTTP/1.1 implementation. Use variables "downgrade-1.0" and
#   "force-response-1.0" for this.
SetEnvIf User-Agent ".*MSIE.*" \
         nokeepalive ssl-unclean-shutdown \
         downgrade-1.0 force-response-1.0

#   Per-Server Logging:
#   The home of a custom SSL log file. Use this when you want a
#   compact non-error SSL logfile on a virtual host basis.
#CustomLog logs/ssl_request_log \
#          "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"

</VirtualHost>
</IfModule>
```

