
*June 22, 2001*

# Configuring Apache *and* Java *for* SSL

## A White Paper

***Revision 0.2***

**Consolidated Solutions Unit Lab**
**Performance Advisor Team**
**Boise, ID**

**Hewlett-Packard**

# Introduction

SSL (Secure Sockets Layer) is an internet protocol that allows for trusted, encrypted communication between two computers (generally a client and a server). It has been in existence for a number of years, and has become the de facto standard for secure communications in web applications.

SSL performs its job in several steps. The process begins when one computer (the client) makes a request to send information to, or receive information from, another computer (the server). In the case of sending information to another computer, the client wants to verify that the server is a trustworthy recipient of the sensitive information (assumed to be sensitive because we are going to the trouble to encrypt it). The client, in essence, sends a request for identification to the server. The server returns a certificate to identify itself. The client verifies that the server is trusted in one of 2 ways.

1. The certificate the server sent exists in a list of trusted certificates the client keeps.
2. The certificate is signed by an authority that is recognized by the client as being trusted.

Once the server has been verified, all communications between the server and the client are encrypted.

It is also possible for authentication to go the other way (server requests verification from client), and 2-way (server and client request authentication from each other). This paper only covers configuration for the client-request model.

This paper covers configuration of Apache and the JRE for SSL communications, as well as creating and installing self-signed certificates. There will be 3 different computers referred to in this document: the server, the client (or host agent), and the certificate authority. It is possible to perform the certificate authority tasks on either of the other 2 computers.

## What you will need

There are a number of required elements to perform the steps contained in this document.

For the server

- Performance Advisor 1.03.00 or greater.

For the certificate authority

- OpenSSL version 0.9.6a source code.
- mod_ssl version 2.6.6-1.3.12 source code.
- Apache version 1.3.12 source code.
- Microsoft Visual C++ 6.0.
- Perl 5

For the client

- Java JRE version 1.2 or greater.
- Performance Advisor Host Agent or CLUI version 1.03.00 or greater.

# Compiling Apache with mod_ssl

The mod_ssl team doesn't fully support the Windows operating system. Because of this, no pre-compiled binaries of mod_ssl for Windows exist. It will be necessary for us to create them. To create mod_ssl, it is necessary to compile the OpenSSL source files and the Apache source files. These steps are done on the certificate authority computer to eliminate some duplication of effort.

### Compiling OpenSSL

These instructions are taken from the install.w32 file included with the OpenSSL 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.
   b. Create a directory under openssl called bin.
   c. Create a directory under openssl called lib.
   d. Create a directory under openssl called include.
   e. In the OpenSSL source directory, under the inc32 directory, is a directory called openssl, copy that directory to C:\openssl\include.
   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. Copy the apps directory from the OpenSSL source directory to C:\openssl

   Replace C: with the drive you wish to install OpenSSL to.

### Preparing 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-2.6.6-1.3.12***
3. Prepare the apache make files to compile both mod_ssl and apache with SSL hooks by typing: ***configure.bat --with-apache=C:\apache_1.3.12 --with-ssl=C:\openssl***

   where C: is where you unpacked the Apache 1.3.12 source files.

### Compile and Install the SSL-ready Apache

These instructions are taken from the install.w32 file included with the mod_ssl source files.

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.nt***
5. Group the Apache files into one directory by typing: ***nmake /f Makefile.nt installr***

   This creates a directory on your C: drive called Apache.
6. Copy the Apache directory to the server system, in the C:\HPSS directory (or the directory you installed the Performance Advisor product to). Click yes to all if you are warned that you will be overwriting the apache directory.

# Creating and Installing Self-Signed Certificates

In the following sections, we cover creating self-signed certificates using the OpenSSL package, and installing them for use by Java programs and Apache. If you are using a certificate signed by a valid Certificate Authority such as Thawte or Verisign, skip ahead to the section titled Configuring Apache to use Certificates and SSL.

**Note:** It is recommended that you purchase a certificate from a valid Certificate Authority, rather than creating a self-signed one, though the Performance Advisor software will work just as well with either.

## Creating Self-Signed Certificates

If you aren't using an already signed certificate from a valid Certificate Authority, it is necessary to create your own (a self-signed certificate). These steps are performed on the certificate authority computer.

1. Add C:\openssl\bin (replacing C:\openssl with the location you copied the OpenSSL files to) to your path.
2. Create an RSA Private Key by typing: ***openssl genrsa –rand –des3 –out server.key 1024***
   This will create a private key based on the triple DES encryption standard.
3. Create a self signed certificate using the key generated in step one by typing:
   ***openssl req –new –x509 –days 365 –key server.key –out server.crt –config c:\openssl\apps\openssl.cnf***

   This creates a self-signed certificate that will expire in 365 days. You will be prompted to provide certain information for the certificate. Most of the information is not required, but the common name field must be filled. Your common name is the name of the computer that has the management station installed on it, as it appears in the ServerName directive in the httpd.conf file. So if the ServerName directive reads somehost.domain.com, that would be your common name.

## Installing Certificates in the JRE Environment

If you are using a self-signed certificate, the client won't be able to validate the certificate sent to it by the server. This is because it doesn't recognize the authority that signed the certificate. Because of this, it is necessary to fool the client into thinking that it has already validated the certificate previously. To accomplish this, we install the certificate created in the previous section into the certificate storage area for the JRE. If you are using JRE 1.2, there is an additional set up step required.

These steps are performed on the computer where the Performance Advisor Host Agent or CLUI are installed.

1. If you are using JRE 1.3, skip to step 2. Otherwise, do the following.
   a. Add the following line to the file <jre_path>\lib\security\java.security:
      ***security.provider.2=com.sun.net.ssl.internal.ssl.Provider***

b. Copy the following jar files from the Host Agent install directory to <path_to_jre>\lib\ext
   i. jnet.jar
   ii. jsse.jar
   iii. jcert.jar

2. Copy the server.crt file to the Host Agent computer in the <jre_path>\lib\security directory.
3. Install the server.crt file into the default certificate store by typing: ***keytool –keystore cacerts –alias mycert –import –file server.crt***
   You will be prompted for a password. The default password for the cacerts keystore is ***changeit***.
4. Restart the Performance Advisor Host Agent (xpstart service on Windows NT or 2000, xppa.start on unix).

This imports the server.crt certificate file into the default certificate store for java, and allows the client application to verify the server.

**Configuring Apache to use Certificates and SSL**

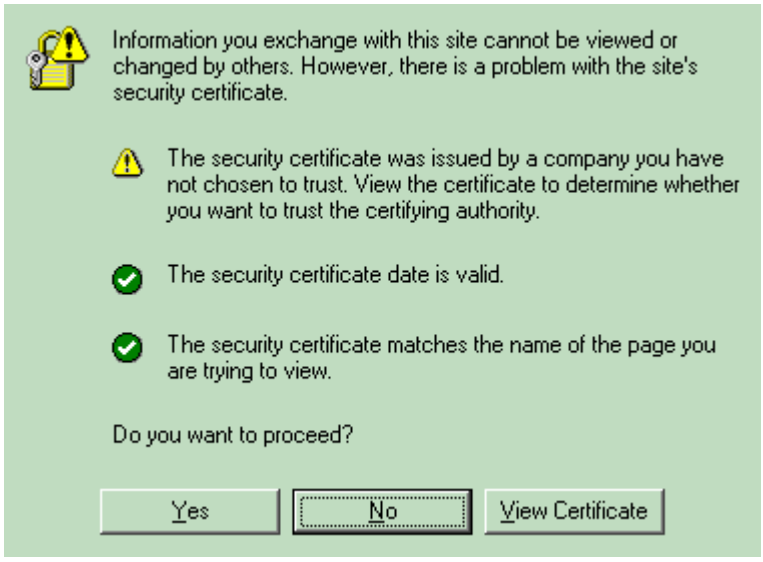
Now we have to install the certificate and its key on the server. Additionally, it is necessary to configure Apache to use mod_ssl.

1. Install the certificate and key using the following steps
   a. In the C:\HPSS\apache\conf directory, create 2 subdirectories: ssl.crt and ssl.key.
   b. Copy the server.crt file to the ssl.crt directory created in step a.
   c. Copy the sever.key file to the ssl.key directory created in step a.
2. Stop the Apache service.
3. Configure Apache to use mod_ssl and the certificate file through the following steps.
   a. Search the httpd.conf file for the word “LoadModule”.
   b. At the end of the LoadModule section type the following line:
      LoadModule ssl_module modules/ApacheModuleSSL.dll
   c. Before the last line in the httpd.conf file, add the following lines.

```
<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

AddType application/x-x509-ca-cert .crt
AddType application/x-pkcs7-crl    .crl
SSLPassPhraseDialog  builtin
SSLSessionCache        dbm:logs/ssl_scache
SSLSessionCacheTimeout  300
SSLMutex  sem
SSLRandomSeed startup builtin
SSLRandomSeed connect builtin
```

```
SSLLog      logs/ssl_engine_log
SSLLogLevel info

SSLEngine on

SSLCertificateFile "C:/hpss/Apache/conf/ssl.crt/server.crt"
SSLCertificateKeyFile "C:/hpss/Apache/conf/ssl.key/server.key"

<Files ~ "\.(cgi|shtml)$">
   SSLOptions +StdEnvVars
</Files>
<Directory "C:/hpss/Apache/cgi-bin">
   SSLOptions +StdEnvVars
</Directory>

SetEnvIf User-Agent ".*MSIE.*" nokeepalive ssl-unclean-shutdown

</IfModule>
```

4. Restart the Apache service.
   **Note:** This configuration requires that all applications using this web server be SSL Enabled.
5. Test this configuration by pointing your browser to the management stations web root (i.e. https://myserver/). The name of the server is the same as the name given on the ServerName directive in the httpd.conf file. A warning dialog should appear that looks like the one below, if you are using a self-signed certificate.




This indicates that the certificate was not signed by one of the major Certificate Authorities. By viewing the certificate, you can add it to your browser's key store, or you can click yes to proceed.