HP Secure Web Server for OpenVMS (based on Apache)
Version 1.3-1 Installation and Configuration Guide

January 2005

Version 1.3-1 for OpenVMS Alpha, based on Apache 1.3.26
CPQ-AXPVMS-CSWS-V0103-1-1.PCSI_SFX_AXPEXE

Version 1.3-1 for OpenVMS I64, based on Apache 1.3.26
HP-I64VMS-CSWS-V0103-1-1.PCSI_SFX_I64EXE

This document contains information about installing and configuring the HP Secure Web Server for OpenVMS Alpha and OpenVMS I64. It also includes information about running the web server, security information, and how to build and debug loadable Apache modules.

Contents

Chapter 1
Installation Requirements and Prerequisites

Before you can install the Secure Web Server for OpenVMS *(based on Apache)*, you should verify that your system meets the minimum hardware and software requirements described below.

1.1 Hardware Requirements

You can install the Secure Web Server for OpenVMS on any system running OpenVMS Alpha or OpenVMS I64.

1.2 Software Requirements

The Secure Web Server requires the following software:

- HP OpenVMS Alpha Version 7.3-1 or higher, or OpenVMS I64 Version 8.2 or higher (OpenVMS 7.3-1 requires the C run-time library patch: VMS731_ACRTL (VMS731_ACRTL-V0300 or later) available from IT Resource Center at http://www2.itrc.hp.com/
- HP TCP/IP Services for OpenVMS Version 5.3 or higher

1.2.1 MultiNet and TCPware Network Products

If you are using MultiNet or TCPware from Process Software Corporation, instead of HP TCP/IP Services for OpenVMS, you should be aware of the following information.

The Secure Web Server has been tested and verified using HP TCP/IP Services for OpenVMS. There are no known problems running the Secure Web Server with other TCP/IP network products such as MultiNet and TCPware, but HP has not formally tested and verified these other products.

MultiNet and TCPware require ECO kits for the Secure Web Server. These ECO kits are subject to change. For the latest ECO kit information, contact Process Software and ask for the ECO kits required to run the Secure Web Server for OpenVMS. Send network connectivity questions regarding the Secure Web Server on TCPware and MultiNet via email to **support@process.com**.

1.2.2 CSWS_JAVA

CSWS_JAVA includes the following Apache Jakarta technologies: Tomcat (JavaServer Pages 1.2, Java Servlet 2.3, MOD_JK, and MOD_JK2) and Ant. (Note: Ant is a partial implementation of the Jakarta Ant subproject and its use is limited to building the included sample web applications and simple user-written web applications for Tomcat.)

CSWS_JAVA has retired support for CSWS_JSERV. If you want to continue JSERV support, download CSWS_JAVA Version 1.1 from the CSWS_JAVA for HP Secure Web Server for OpenVMS web site at http://h71000.www7.hp.com/openvms/products/ips/apache/csws_java.html.

See the CSWS_JAVA for HP Secure Web Server for OpenVMS *Installation Guide and Release Notes* for CSWS_JAVA requirements.

1.2.3 CSWS_PHP

PHP is a server-side, cross-platform, HTML embedded scripting language that lets you create dynamic web pages. PHP-enabled web pages are treated the same as regular HTML pages, and you can create and edit them the way you normally create regular HTML pages.

See the CSWS_PHP for HP Secure Web Server for OpenVMS *Installation Guide and Release Notes* for CSWS_PHP requirements.

1.2.4 CSWS_PERL

Perl has become the premier scripting language of the Web, as most CGI programs are written in Perl. The Secure Web Server for OpenVMS supports an optional kit, CSWS_PERL. This kit includes MOD_PERL, an interface between Perl and the Secure Web Server which lets you write modules entirely in Perl.

See the CSWS_PERL for HP Secure Web Server for OpenVMS *Installation Guide and Release Notes* for CSWS_PERL requirements.

1.2.5 Building the Apache HTTP Server from Source Code

The Secure Web Server V1.3-1 kit is based on Apache 1.3.26. Source code and instructions for building an Apache HTTP server for OpenVMS can be found at the Secure Web Server for OpenVMS web site at http://h71000.www7.hp.com/openvms/products/ips/apache/csws_source.html.

Chapter 2
Installation and Configuration

Read this chapter to install and configure the Secure Web Server for OpenVMS. Installation and configuration consists of the following steps:

1. Read the release notes
2. Install the server and optional modules
3. Configure the server
4. Review the post configuration checklist
5. Test the installation

Detailed instructions for completing each of these steps are provided below.

2.1 Read the Release Notes

Before you begin the installation, you should read the *HP Secure Web Server for OpenVMS Release Notes*.

2.2 Install the Secure Web Server and Optional Modules

If you are upgrading to OpenVMS Alpha Version 8.2, the currently available CSWS_JAVA V2.1, CSWS_PERL V1.1, and Perl for OpenVMS V5.6-1 kits will work properly.

New kits are required for the Secure Web Server and CSWS_PHP on OpenVMS Alpha, and for the Secure Web Server and all of the optional kits on OpenVMS I64.

You can install the Secure Web Server by itself or with one or more of the optional modules. You can install the optional modules later, if you choose.

Before you begin, do the following:

1. Decide what you want to install.
2. Review the software requirements for the server and each optional module you are installing.
3. Decide where you want to install the kit.

Note

The Secure Web Server, CSWS_PERL, and CSWS_PHP must be installed in the same directory (required).

By default, the Secure Web Server, CSWS_PERL, and CSWS_PHP are installed in SYS$COMMON. However, HP recommends that you specify another location.

CSWS_JAVA can be installed into a different disk or directory from the Secure Web Server. (HP requires that you install CSWS_JAVA on an ODS-5 enabled disk. Your installation of the Secure Web Server can remain on an ODS-2 disk.)

HP recommends that you **shut down the Secure Web Server** (and Tomcat, which runs as a separate process) before installing a new version of any component: CSWS, CSWS_PHP, CSWS_Perl, or CSWS_JAVA (Tomcat).

Follow these instructions to install the Secure Web Server by itself or with the optional modules.

1. Make sure you are logged in as a privileged OpenVMS user (for example, SYSTEM).
2. Select UIC group and member numbers for the APACHE$WWW account that will be created by the installation procedure. HP recommends that you use an empty or new UIC group (without current members). Servers typically use the highest unused UIC group (for example, [370,1]).

   To ensure that the UIC you chose for APACHE$WWW has READ and WRITE access to the intended login device, use the SHOW DEVICE/FULL command.
3. Decompress the server kit with one of the following commands, depending on the platform on which you are installing the server kit:

   ```
   $ RUN CPQ-AXPVMS-CSWS-V0103-1-1.PCSI_SFX_AXPEXE   ! on Alpha
   $ RUN HP-I64VMS-CSWS-V0103-1-1.PCSI_SFX_I64EXE    ! on I64
   ```

   The files are expanded and are named `CPQ-AXPVMS-CSWS-V0103-1-1.PCSI` (on Alpha) and `HP-I64VMS-CSWS-V0103-1-1.PCSI` (on I64). Do not rename these files.
4. Start the installation with the PRODUCT INSTALL command. Use the /DESTINATION qualifier to specify a target device and directory for the installation. If you do not specify a destination, the software will be installed in SYS$COMMON. HP recommends that you specify another location.

Note

Once you enter a PCSI INSTALL CSWS/DESTINATION=[*destination*] command, you cannot change the installation location unless you remove CSWS and then reinstall it. To change the installation location when you upgrade to a new version of CSWS, you must first enter the PCSI REMOVE CSWS command, then enter PCSI INSTALL CSWS/DESTINATION=[*new-destination*].

Review the software requirements for the server and each optional module you are about to install. To prevent installation problems, make sure the required software is installed **before** you enter the PRODUCT INSTALL command.

To install the server, enter the following command:

```
$ PRODUCT INSTALL CSWS /DESTINATION=device:[directory-name]
```

To install the server and one or more of the optional modules, specify CSWS and the CSWS_*nnnn* kit name on the PRODUCT INSTALL command line, separated by commas. (You must have previously expanded the optional kit in order to install it with the server.)

For example, to install the server and CSWS_PHP, use the following command:

```
$ PRODUCT INSTALL CSWS, CSWS_PHP /DESTINATION=device:[directory-name]
```

The installation proceeds and displays product information as well as post-installation instructions. The installation is finished when you see the DCL prompt ($).

After the installation, you must configure the Secure Web Server.

Note

Do not attempt to start the server or configure any optional modules before you have configured the server.

2.2.2 Sample Installation

Following is an example of the Secure Web Server product installation.

```
$ PRODUCT INSTALL CSWS /DESTINATION=DKB300:[000000]

The following product has been selected:
    CPQ AXPVMS CSWS V1.3-1                  Layered Product

Do you want to continue? [YES]

Configuration phase starting ...

You will be asked to choose options, if any, for each selected product and for
any products that may be installed to satisfy software dependency requirements.

CPQ AXPVMS CSWS V1.3-1

    Hewlett-Packard Company & The Apache Software Foundation.

* This product does not have any configuration options.

Execution phase starting ...

The following product will be installed to destination:
    CPQ AXPVMS CSWS V1.3-1                  USER$DISK3:[000000.]

Portion done: 0%...10%...20%...30%...40%...50%...60%...70%...90%...100%

The following product has been installed:
    CPQ AXPVMS CSWS V1.3-1                  Layered Product

CPQ AXPVMS CSWS V1.3-1

    Release notes are available in SYS$HELP:CSWSxxx.RELEASE_NOTES.

    HP highly recommends that you read these release notes.

    For the most up-to-date documentation, including release notes,
    Frequently Asked Questions (FAQs), and information about configuring
    and running the HP Secure Web Server, please see the web pages at:

    http://h71000.www7.hp.com/openvms/products/ips/apache/csws.html

    Post-installation tasks are required for the HP Secure Web Server.

    The OpenVMS Installation and Configuration Guide gives detailed directions.
    This information is a brief checklist.

    Configure OpenVMS aspects of the HP Secure Web Server by:
```

```
    $ @SYS$MANAGER:APACHE$CONFIG

If the OpenVMS username APACHE$WWW does not exist, you will be
prompted to create that username.  File ownerships are set to UIC
[APACHE$WWW], etc.

After configuration, start the HP Secure Web Server manually by
entering:

    $ @SYS$STARTUP:APACHE$STARTUP

Check that neither SYLOGIN.COM nor the LOGIN.COM write any output to
SYS$OUTPUT:.  Look especially for a

    $ SET TERMINAL/INQUIRE.

Start the HP Secure Web Server at system boot time by adding the
following lines to SYS$MANAGER:SYSTARTUP_VMS.COM:

    $ file := SYS$STARTUP:APACHE$STARTUP.COM
    $ if f$search("''file'") .nes. "" then @'file'

Shutdown the Apache server at system shutdown time by adding the
following lines to SYS$MANAGER:SYSHUTDWN.COM:

    $ file := SYS$STARTUP:APACHE$SHUTDOWN.COM
    $ if f$search("''file'") .nes. "" then @'file'

Test the installation using your favorite Web browser.
Replace host.domain in the following URL (Uniform Resource Locator)
with the information for the HP Secure Web Server just installed,
configured, and started.


URL http://host.domain/ should display the standard introductory page
from the Apache Software Foundation. This has the bold text "It
Worked! The Apache Web Server is Installed on this Web Site!" at the
top  and the Apache server logo prominently displayed at the bottom.
If you do not see this page, check the HP Secure Web Server
release notes, particularly the Frequently Asked Questions section.

If you'd like to use secure connections with the HP Secure Web Server
then you'll need to create a server certificate.  We recommend that
you start by creating a 30 day self signed certificate using the
following certificate tool:

    $ @APACHE$COMMON:[OPENSSL.COM]OPENSSL_AUTO_CERT.COM

Once the certificate has been created you'll need to uncomment the
following directive in the APACHE$COMMON:[CONF]HTTPD.CONF file to
enable SSL.

    Include /apache$root/conf/ssl.conf

Thank you for using the HP Secure Web Server.
```

2.3 Configure the Secure Web Server

After you have installed the Secure Web Server, you are ready to configure it.

The installation wrote values, such as the name of the directory where the Secure Web Server is installed, to the file:

```
SYS$COMMON:[SYSMGR]APACHE$CONFIG_DEFAULT.DAT
```

The information stored in this file provides the default values you see during configuration. Do not try to modify the contents of this file.

The configuration procedure gives you the opportunity to separate the server components---server application, server system files, and server content files---and store them wherever it is most appropriate in your environment. By default, they are all configured in SYS$COMMON or the destination you specified on the PRODUCT INSTALL command line. During configuration you are asked if you would like to specify different locations.

If you have an OpenVMS Cluster, see Section 3.13 before you continue with the configuration.

2.3.1 Configuring a Single Server

Most users need only run a single server on a given system. When that server configuration is started, it usually exists as a main process and multiple child processes to handle multiple user requests. Those child processes may also generate subprocesses to handle certain types of requests (such as CGI scripts).

For information about configuring multiple servers, see Section 2.3.3.

You can define server startup, shutdown, and tag configuration elements while configuring a server using APACHE$CONFIG CONFIG. See Table 3-4 for the process logical names that you can specify.

To configure a single server, enter the following command:

```
$ @SYS$MANAGER:APACHE$CONFIG
```

This command procedure asks you a number of questions about the OpenVMS run-time environment for the Secure Web Server. These values are written into the following data file:

```
SYS$MANAGER:APACHE$CONFIG.DAT
```

2.3.2 Sample Configuration of a Single Server

This section shows a sample configuration dialog after you have completed the installation of the Secure Web Server.

This sample illustrates the following:

- A configuration for the default installation destination of DISK$DKA0:[APACHE].
- A configuration for the first time. When you configure the Secure Web Server for the first time, you are asked to provide account information. This allows the Secure Web Server to create an OpenVMS account (called APACHE$WWW) for the server to use. In this sample dialog, the account has a UIC group number of 377 and a member number of 1.
- When the user is asked for the device and directory where the installation procedure installed the software, the user chose the default, DISK$DKA0:[APACHE].
- When the user is asked for the device and directory for the system-specific files, the user chose the default for the name of the system: DISK$DKA0:[APACHE.SPECIFIC.MYHOST].
- The user chose to define the logical names APACHE$SPECIFIC, APACHE$COMMON, and APACHE$ROOT systemwide.
- The user specified the configuration file name APACHE$BX23. The full file specification for the APACHE$CONFIG.COM data file is SYS$MANAGER:APACHE$BX23.DAT.
- The user chose to enable MOD_SSL.
- The user chose not to enable suEXEC.
- The user chose to define server startup and shutdown procedures and chose a custom server name tag.

- The user chose not to set any command-line arguments for the server. For more information about command-line arguments, see the *Secure Web Server SSL User Guide.*
- The user chose to set the owner UIC on Secure Web Server files.

```
$ @SYS$MANAGER:APACHE$CONFIG CONFIG APACHE$BX23.DAT

               Secure Web Server V1.3-1 for OpenVMS
                           [based on Apache]

        This procedure helps you define the parameters and the
        operating environment required to run the  Secure Web
        Server on this system.

[Creating OpenVMS username "APACHE$WWW" ]
[Starting SYS$COMMON:[APACHE]APACHE$ADDUSER.COM ]

Press enter to continue...


PLEASE NOTE:

You will be prompted for the following information:

    Full name for APACHE$WWW: ! Full name of site server administrator/owner.

    Password:  ! Password for the APACHE$WWW account

    UIC Group Number:   ? ! Question mark will display a list of all
                          ! UIC groups currently in use. Quite useful.
                          ! Please pick a group separate from all other
                          ! usernames.
                          ! Servers are usually given the first unused group,
                          ! starting at [377,*] and working down.  DO _not_
                          ! go below SYSGEN parameter MAXSYSGROUP.

    UIC Member Number:  1 ! Question mark will display a list of all
                          ! UIC members currently in use in that group.
                          ! %UAF-W-BADSPC, no user matches specification
                          ! means the group is empty.

*********************************************************************
*  Creating a NEW user account...                                   *
*                                                                   *
*  If at ANY TIME you need help about a prompt, just type "?".      *
*********************************************************************


 *** Processing APACHE$WWW's account ***

Full name for APACHE$WWW:  Secure Web Server
Password (password is not echoed to terminal) [APACHE$WWW]:

UIC Group number [200]: 377

UIC Member number: 1

%UAF-I-ADDMSG, user record successfully added
%UAF-I-RDBADDMSGU, identifier APACHE$WWW value [000377,000001]
                   added to rights database
%UAF-I-RDBADDMSGU, identifier AP_HTTPD value [000377,177777]
                   added to rights database
%UAF-I-MDFYMSG, user record(s) updated
%UAF-I-MDFYMSG, user record(s) updated
%UAF-I-DONEMSG, system authorization file modified
%UAF-I-RDBDONEMSG, rights database modified

Check newly created account:


Username: APACHE$WWW                   Owner:   Secure WEB
```

```
SERVER
Account:  AP_HTTPD                       UIC:    [377,1] ([AP_HTTPD,APACHE$WWW])
CLI:      DCL                            Tables: DCLTABLES
Default:  APACHE$ROOT:[000000]
LGICMD:   LOGIN
Flags:  LockPwd DisNewMail DisMail DisReport
Primary days:   Mon Tue Wed Thu Fri
Secondary days:                     Sat Sun
Primary   000000000011111111112222  Secondary 000000000011111111112222
Day Hours 012345678901234567890123  Day Hours 012345678901234567890123
Network:  ##### Full access ######            ##### Full access ######
Batch:    -----  No access  ------            -----  No access  ------
Local:    -----  No access  ------            -----  No access  ------
Dialup:   -----  No access  ------            -----  No access  ------
Remote:   -----  No access  ------            -----  No access  ------
Expiration:            (none)    Pwdminimum:  6   Login Fails:     0
Pwdlifetime:         90 00:00    Pwdchange:      (pre-expired)
Last Login:            (none) (interactive),            (none)
(non-interactive)
Maxjobs:         0  Fillm:       300  Bytlm:       200000
Maxacctjobs:     0  Shrfillm:      0  Pbytlm:           0
Maxdetach:       0  BIOlm:       300  JTquota:       4096
Prclm:          20  DIOlm:       300  WSdef:        15000
Prio:            4  ASTlm:       610  WSquo:        30000
Queprio:         4  TQElm:       610  WSextent:     30000
CPU:        (none)  Enqlm:      2000  Pgflquo:     250000
Authorized Privileges:
  NETMBX       TMPMBX
Default Privileges:
  NETMBX       TMPMBX
%UAF-I-NOMODS, no modifications made to system authorization file
%UAF-I-RDBNOMODS, no modifications made to rights database

Please verify that this account does not validate any site-specific
security policy. This account will enabled and it will have no
expiration date.

Is everything satisfactory with the account [YES]:

PLEASE NOTE:

The APACHE$WWW account was created with the minimum SYSUAF quotas to
run the server.  On almost all systems, the server should start but
these parameters will need to be increased to improve performance or
to keep up with increased demands.

See Release notes for details.


Please provide the device and directory where the kit was installed.

Device and directory where the kit was installed? [DISK$DKA0:[APACHE]]

Please provide the device and the directory for this specific system.
Each system in a cluster has its own rooted logical name for writing
system-specific files (e.g. the LOG files).  The device and directory
specified here will be used exclusively by this system.  This command
procedure creates the system-specific directory and the necessary
subdirectories.  You do not need to create these directories yourself.

Device and directory for this system? [DISK$DKA0:[APACHE.SPECIFIC.MYHOST]]

You can specify an optional server startup procedure that executes
before the server image is started.

If you do not specify a startup procedure, the default startup
procedure, APACHE$ROOT:[000000]APACHE$SERVER_STARTUP.COM, will be
executed, if it exists.

No server startup is defined for this server.
```

```
Change server startup? [NO] y
Server startup procedure for this server (optional)? [NONE]
apache$specific:[000000]bx23_startup.com

You can specify an optional server shutdown procedure that executes
after the server image terminates.

If you do not specify a shutdown procedure, the default shutdown
procedure, APACHE$ROOT:[000000]APACHE$SERVER_SHUTDOWN.COM, will be
executed, if it exists.

No server shutdown is defined for this server.

Change server shutdown? [NO] y
Server shutdown procedure for this server (optional)? [NONE]
apache$specific:[000000]bx23_shutdown.com

The server tag is an optional name that can be associated with this
server. The tag is a string of up to 4 characters that appears in the
process name for server processes created in this environment.

No server tag is defined for this server.

Change server tag? [NO] y
Server tag for this server (optional)? [NONE] bx23

Do you want to define the logical names APACHE$SPECIFIC, APACHE$COMMON,
and APACHE$ROOT systemwide for this configuration?

If you are planning to run a single web server on this system, then
these logical names should be made systemwide (the default).

If you are planning to run multiple web servers on this system, only
one of the APACHE$CONFIG data files should be used to define the
systemwide logicals.

Define systemwide logicals? [YES]

Do you want to enable the security features provided by MOD_SSL?
If so, the server will support the HTTPS (HTTP over the Secure Socket
Layer) protocol.

Enable MOD_SSL? [YES]

Do you want to enable the impersonation features provided by SuEXEC?
If so, the server will support running CGIs using specified usernames.

Enable suEXEC? [NO]

You can specify optional command-line arguments for the server below.
(For example, specify "-D<name>" to define a name for the
<IfDefine>
directives or specify "-d<path>" to
specify the ServerRoot directory.)
Note that the optional arguments are case-sensitive.

There are currently no optional command-line arguments.

Change this value? [NO]

To operate successfully, the server processes must have read access
to the installed files and read-write access to certain other files
and directories. HP recommends that you use this procedure to
set the owner UIC on the CSWS files and directories to match the server.
You should do this each time the product is installed, but it only has
to be done once for each installation on a cluster.

WARNING: The owner UIC for some files in APACHE$COMMON:[000000...] is
not the server UIC (APACHE$WWW).  This may cause the web server process
to fail with a protection violation.
```

```
Set owner UIC on CSWS files? [YES]

Setting ownership on files.  This could take a minute or two.  . . .

Configuration is complete.  To start the server:

    $ @SYS$STARTUP:APACHE$STARTUP.COM START APACHE$BX23.DAT

$ @SYS$STARTUP:APACHE$STARTUP.COM START APACHE$BX23.DAT

%APACHE-S-PROC_ID, identification of created process is 0000CD46

$ SHOW SYSTEM /OWNER_UIC=[APACHE$WWW]
OpenVMS V7.3-1  on node MYHOST   5-SEP-2003 15:35:12.53  Uptime  34 04:28:25
  Pid    Process Name    State  Pri      I/O       CPU       Page flts  Pages
0000CD46 APACHE$BX23     LEF      6     1035   0 00:00:02.19       504    523
0000C547 APACHE$BX23000  LEF      6      845   0 00:00:01.32       402    482
0000CB48 APACHE$BX23001  LEF      6      849   0 00:00:01.27       410    482
0000C949 APACHE$BX23002  LEF      6      843   0 00:00:01.16       408    484
0000C74A APACHE$BX23003  LEF      6      842   0 00:00:01.37       408    482
0000C34B APACHE$BX23004  LEF      4      842   0 00:00:01.38       402    482
```

2.3.3 Configuring Multiple Servers

For some advanced configurations, it may be necessary to run two or more servers on the same system. For example, you may decide to run multiple virtual hosts on the same system and have each virtual host serviced by a separate server.

For each server process, run the APACHE$CONFIG.COM command procedure to define the OpenVMS operating environment for each server. In particular, each server must have its own APACHE$SPECIFIC directory into which it writes its output files.

To configure an additional server, enter the following command:

```
    $ @SYS$MANAGER:APACHE$CONFIG CONFIGURE filename
```

where *filename* is the file specification of the disk file where the APACHE$CONFIG.COM command procedure saves the values provided by the user.

2.3.4 Sample Configuration of Multiple Servers

This section shows a sample configuration dialog of a second server after you have completed the installation of the Secure Web Server and configured the first server.

This sample illustrates the following:

- A configuration for the default installation destination: DISK$DKA0:[APACHE].
- The user specified the file name APACHE$MYHOST_1. The full file specification for the APACHE$CONFIG.COM data file is SYS$MANAGER:APACHE$MYHOST_1.DAT.
- This configuration does not create a new account, because that operation is done only once.
- When the user is asked for the device and directory where the installation procedure installed the software, the user chose the default, DISK$DKA0:[APACHE].
- When the user is asked for the device and directory for the system-specific files, the user entered a new location for the system-specific files: DISK$DKA0:[APACHE.SPECIFIC.MYHOST_1].
- The user chose not to define the logical names APACHE$SPECIFIC, APACHE$COMMON, and APACHE$ROOT systemwide. The other server configuration defines these logical names. That APACHE$CONFIG.COM data file is essentially the default configuration. Therefore, this APACHE$CONFIG.COM configuration should not define these logical names systemwide.
- The user chose not to enable MOD_SSL.

- The user chose not to enable suEXEC.
- The user chose not to define server startup and shutdown procedures nor a custom server name tag.
- The user chose to add the command-line argument -DMYHOST_1. This will define the name MYHOST_1 so that the HTTPD.CONF can contain an <IfDefine MYHOST_1> directive. For more information about command-line arguments, see the Secure Web Server SSL User Guide
- The user chose not to set the owner UIC on Secure Web Server files. The owner UIC has already been set during an earlier run of the APACHE$CONFIG.COM procedure.
- The APACHE$CONFIG.COM procedure has created processwide logical names for APACHE$SPECIFIC, APACHE$COMMON, and APACHE$ROOT that match the APACHE$MYHOST_1 data file. Even though the systemwide logical name for APACHE$ROOT points to another location, the processwide definition for APACHE$ROOT is appropriate for this server. Therefore, the user can conveniently edit the APACHE$ROOT:[CONF]HTTPD.CONF file.
- The user must reference the APACHE$MYHOST_1 data file as the second parameter to the APACHE$STARTUP.COM command procedure.

```
$ @SYS$MANAGER:APACHE$CONFIG CONFIGURE APACHE$MYHOST_1

                  Secure Web Server V1.3-1 for OpenVMS
                           [based on Apache]

        This procedure helps you define the parameters and the
        operating environment required to run the  Secure Web
        Server on this system.

Please provide the device and directory where the kit was installed.

Device and directory where the kit was installed? [DISK$DKA0:[APACHE]]

Please provide the device and the directory for this specific system.
Each system in a cluster has its own rooted logical name for writing
system-specific files (e.g. the LOG files).  The device and directory
specified here will be used exclusively by this system.  This command
procedure creates the system-specific directory and the necessary
subdirectories.  You do not need to create these directories yourself.

Device and directory for this system? [DISK$DKA0:[APACHE.SPECIFIC.MYHOST]]
                                       DISK$DKA0:[APACHE.SPECIFIC.MYHOST_1]

You can specify an optional server startup procedure that executes
before the server image is started.

If you do not specify a startup procedure, the default startup
procedure, APACHE$ROOT:[000000]APACHE$SERVER_STARTUP.COM, will be
executed, if it exists.

No server startup is defined for this server.

Change server startup? [NO]
Server startup procedure for this server (optional)? [NONE]

You can specify an optional server shutdown procedure that executes
after the server image terminates.

If you do not specify a shutdown procedure, the default shutdown
procedure, APACHE$ROOT:[000000]APACHE$SERVER_SHUTDOWN.COM, will be
executed, if it exists.

No server shutdown is defined for this server.

Change server shutdown? [NO]
Server shutdown procedure for this server (optional)? [NONE]

The server tag is an optional name that can be associated with this
server. The tag is a string of up to 4 characters that appears in the
process name for server processes created in this environment.
```

```
No server tag is defined for this server.

Change server tag? [NO]
Server tag for this server (optional)? [NONE]

Do you want to define the logical names APACHE$SPECIFIC, APACHE$COMMON,
and APACHE$ROOT systemwide for this configuration?

If you are planning to run a single web server on this system, then
these logical names should be made systemwide (the default).

If you are planning to run multiple web servers on this system, only
one of the APACHE$CONFIG data files should be used to define the
systemwide logicals.

Define systemwide logicals? [YES] NO

Do you want to enable the security features provided by MOD_SSL?
If so, the server will support the HTTPS (HTTP over the Secure Socket
Layer) protocol.

Enable MOD_SSL? [YES] NO

Do you want to enable the impersonation features provided by SuEXEC?
If so, the server will support running CGIs using specified usernames.

Enable suEXEC? [NO]

You can specify optional command-line arguments for the server below.
(For example, specify -D<name> to define a name for the <IfDefine>
directives or specify -d<path> to specify the ServerRoot directory.)
Note that the optional arguments are case-sensitive.

There are currently no optional command-line arguments.

Change this value? [NO] YES
New command-line arguments? -DMYHOST_1

To operate successfully, the server processes must have read access
to the installed files and read-write access to certain other files
and directories. HP recommends that you use this procedure to
set the owner UIC on the CSWS files and directories to match the server.
You should do this each time the product is installed, but it only has
to be done once for each installation on a cluster.

Set owner UIC on CSWS files? [YES] NO

Configuration is complete.  To start the server:

        $ @SYS$STARTUP:APACHE$STARTUP.COM START APACHE$MYHOST_1

    $ SHOW LOGICAL /PROCESS APACHE$*

    (LNM$PROCESS_TABLE)

  "APACHE$COMMON" = "MYHOST$DKA0:[APACHE.]"
  "APACHE$ROOT" = "MYHOST$DKA0:[APACHE.SPECIFIC.MYHOST_1.]"
   = "APACHE$COMMON:"
  "APACHE$SERVER_PID" = "0000D755"
  "APACHE$SPECIFIC" = "MYHOST$DKA0:[APACHE.SPECIFIC.MYHOST_1.]"

    $ SHOW SYSTEM /OWNER_UIC=[APACHE$WWW]

OpenVMS V7.3-1  on node MYHOST   6-SEP-2003 12:01:34.25  Uptime  35 00:54:40
  Pid    Process Name    State  Pri      I/O       CPU       Page flts  Pages
0000CD46 APACHE$00       LEF      6     1052   0 00:00:02.59       505    524
0000C547 APACHE$00000    LEF      6      914   0 00:00:01.40       416    497
0000CB48 APACHE$00001    LEF      6      916   0 00:00:01.37       424    497
0000C949 APACHE$00002    LEF      6     1176   0 00:00:01.42       443    520
0000C74A APACHE$00003    LEF      6     1019   0 00:00:01.63       429    504
0000C34B APACHE$00004    LEF      6      909   0 00:00:01.47       416    497
```

```
0000C34C APACHE$00005     LEF       6       841   0 00:00:01.20       412    484
0000C056 APACHE$01        LEF       6       382   0 00:00:00.34       430    328
0000C457 APACHE$01000     LEF       6       252   0 00:00:00.26       322    310
0000C958 APACHE$01001     LEF       6       252   0 00:00:00.20       322    310
0000C959 APACHE$01002     LEF       6       252   0 00:00:00.29       318    310
0000AA5A APACHE$01003     LEF       6       256   0 00:00:00.26       320    3100000C95B
APACHE$01004    LEF       4      252   0 00:00:00.28       314    310
```

2.3.5 Specifying the READ Function

You can specify the READ function to the APACHE$CONFIG.COM procedure, as follows:

```
    $ @SYS$MANAGER:APACHE$CONFIG READ [filename]
```

where *filename* is an optional file specification of a Secure Web Server configuration file created by APACHE$CONFIG.COM (the default is SYS$MANAGER:APACHE$CONFIG.DAT).

The READ function parses the specified configuration file and sets the APACHE$SPECIFIC, APACHE$COMMON, and APACHE$ROOT processwide logical names to appropriate values for that configuration file.

If you want to edit the clusterwide server configuration file (HTTPD.CONF), you can define the APACHE$COMMON logical name and use the logical name to edit the file by entering:

```
     $ @SYS$MANAGER:APACHE$CONFIG READ
     $ EDIT APACHE$COMMON:[CONF]HTTPD.CONF
```

For example, if you have an alternate Secure Web Server configuration file for an additional server called APACHE$ALIAS_2.DAT, you can define the APACHE$SPECIFIC logical name and use it to examine the SSL key files for that server as follows:

```
    $ @SYS$MANAGER:APACHE$CONFIG READ APACHE$ALIAS_2.DAT
    $ DIRECTORY APACHE$SPECIFIC:[CONF.SSL_KEY]
```

Because it may be important to identify the running server process that corresponds to a particular configuration file, the READ function also defines a logical name called APACHE$SERVER_PID. This logical name is equivalent to the process ID of the running server, or a blank space (" ") if there is no running server for the specified configuration file.

When you use APACHE$STARTUP.COM, APACHE$SHUTDOWN.COM, or APACHE$CONFIG.COM on a Secure Web Server configuration data file, the process logical names are set to reflect that configuration.

For example, if you have an alternate Secure Web Server configuration file for an additional server called APACHE$ALIAS_2.DAT, and you have your process logical names set to reflect that configuration, you may decide to start your default configuration by entering:

```
    $ @SYS$STARTUP:APACHE$STARTUP
```

You have now changed the processwide logical names to reflect the default configuration: the processwide logical names will be the same as the systemwide logical names. To restore the processwide logical names, use the APACHE$CONFIG.COM READ function:

```
    $ @SYS$MANAGER:APACHE$CONFIG READ APACHE$ALIAS_2.DAT
```

2.3.6 Specifying the STATUS Function

You can specify the STATUS function to the APACHE$CONFIG.COM procedure, as follows:

```
    $ @SYS$MANAGER:APACHE$CONFIG STATUS [filename]
```

If a configuration data file is specified, the STATUS function reads the specified configuration data file and displays status information about the running server.

If a configuration data file is not specified, the default is to show the status of the default server process. If you specify an asterisk instead of a data file, the status for all running servers is shown. For example:

```
Process ID:         00003B3B
Process name:       APACHE$00
Started:            12-SEP-2004 11:06:51.49
Specific directory:
MYHOST$DKA0:[SYS0.SYSCOMMON.APACHE.SPECIFIC.MYHOST.]
Common directory:   MYHOST$DKA0:[SYS0.SYSCOMMON.APACHE.]
Config file:        SYS$MANAGER:APACHE$CONFIG.DAT
  00003B3B APACHE$00       LEF      5      820   0 00:00:07.06
  0000CE41 APACHE$00005    LEF      6      757   0 00:00:00.88
  00011245 APACHE$00000    LEF      6     1109   0 00:00:01.24
  00011A47 APACHE$00001    LEF      6     1191   0 00:00:01.38
  00011A49 APACHE$00002    LEF      6     1115   0 00:00:01.42
  0000E14A APACHE$00003    LEF      6     1031   0 00:00:01.21
  0001244B APACHE$00004    LEF      6     1015   0 00:00:01.25
  00013462 APACHE$00006    LEF      6      627   0 00:00:00.69
  00010164 APACHE$00007    LEF      6      641   0 00:00:00.72
  00010165 APACHE$00008    LEF      6      635   0 00:00:00.75

Process ID:         00009254
Process name:       APACHE$03
Started:            12-SEP-2004 11:12:02.23
Specific directory:
MYHOST$DKA0:[SYS0.SYSCOMMON.APACHE.SPECIFIC.MYHOST_3.]
Common directory:   MYHOST$DKA0:[SYS0.SYSCOMMON.APACHE.]
Config file:        SYS$SYSROOT:[SYSMGR]MYHOST_3.DAT;
  00009254 APACHE$03       LEF      5      955   0 00:00:06.89
  0000F759 APACHE$03000    LEF      6      368   0 00:00:00.45
  0000F75A APACHE$03001    LEF      6      367   0 00:00:00.43
  0000F75B APACHE$03002    LEF      6      367   0 00:00:00.37
  0000F35C APACHE$03003    LEF      6      368   0 00:00:00.39
  0001265D APACHE$03004    LEF      6      366   0 00:00:00.40

Process ID:         00009642
Process name:       APACHE$01
Started:            12-SEP-2004 11:08:49.63
Specific directory:
MYHOST$DKA0:[SYS0.SYSCOMMON.APACHE.SPECIFIC.MYHOST_1.]
Common directory:   MYHOST$DKA0:[SYS0.SYSCOMMON.APACHE.]
Config file:        SYS$SYSROOT:[SYSMGR]MYHOST_1.DAT;
  00009642 APACHE$01       LEF      5     1118   0 00:00:07.02
  0001154D APACHE$01000    LEF      6      372   0 00:00:00.40
  0001154F APACHE$01001    LEF      6      367   0 00:00:00.47
  00011550 APACHE$01002    LEF      6      367   0 00:00:00.44
  0000BE51 APACHE$01003    LEF      6      367   0 00:00:00.43
  0000C352 APACHE$01004    LEF      6      367   0 00:00:00.39

Process ID:         0000E648
Process name:       APACHE$02
Started:            12-SEP-2004 11:11:20.68
Specific directory:
MYHOST$DKA0:[SYS0.SYSCOMMON.APACHE.SPECIFIC.MYHOST_2.]
Common directory:   MYHOST$DKA0:[SYS0.SYSCOMMON.APACHE.]
Config file:        SYS$SYSROOT:[SYSMGR]MYHOST_2.DAT;
  0000E648 APACHE$02       LEF      5     1181   0 00:00:07.67
  00010B53 APACHE$02000    LEF      6      370   0 00:00:00.44
  0000E455 APACHE$02001    LEF      6      368   0 00:00:00.36
  0000E456 APACHE$02002    LEF      6      367   0 00:00:00.49
  0000E457 APACHE$02003    LEF      6      367   0 00:00:00.41
  0000F758 APACHE$02004    LEF      6      367   0 00:00:00.42
```

2.3.7 Managing Multiple Servers

This section describes how to manage multiple web servers. For example, consider the case where a site is using three servers, as follows:

- SYS$MANAGER:APACHE$CONFIG.DAT is the default configuration file for the host MYHOST. The system-specific directory is MYHOST$DKA0:[APACHE.SPECIFIC.MYHOST.] and the server is started with the command-line argument -DMYHOST.
- SYS$MANAGER:APACHE$MYHOST_1.DAT is the first additional configuration file. The system-specific directory is MYHOST$DKA0:[APACHE.SPECIFIC.MYHOST_1.] and the server is started with the command-line argument -DMYHOST_1.
- SYS$MANAGER:APACHE$MYHOST_2.DAT is the second additional configuration file. The system-specific directory is MYHOST$DKA0:[APACHE.SPECIFIC.MYHOST_2.] and the server is started with the command-line argument -DMYHOST_2.

The following sections discuss the issues you may encounter when managing the preceding three servers.

2.3.7.1 HTTPD.CONF

Because there are multiple servers at work, there must be some differences in the HTTPD.CONF file for each server. You can maintain the HTTPD.CONF file in two different ways: maintain multiple HTTPD.CONF files or a single HTTPD.CONF file.

To create and maintain multiple HTTPD.CONF files, you rely on the fact that each server has a separate configuration-specific root directory. You can use the APACHE$CONFIG READ command (described in the previous section) to set the processwide logical name APACHE$SPECIFIC to the configuration-specific directory. You then edit the file APACHE$SPECIFIC:[CONF]HTTPD.CONF.

To maintain a single clusterwide HTTPD.CONF file, you can can use the -D command-line argument to enable optional sections of the HTTPD.CONF file. For example, each server has to listen on a different port. The default HTTPD.CONF file contains both a PORT (old) and LISTEN (new) directive pointing to port 80. You should comment out the PORT 80 directive and use the following directives to set up your LISTEN ports. The clusterwide HTTPD.CONF file might therefore contain the following lines:

```
<IfDefine MYHOST>
Listen 80
</IfDefine>

<IfDefine MYHOST_1>
Listen 8010
</IfDefine>

<IfDefine MYHOST_2>
Listen 8020
</IfDefine>
```

2.3.7.2 Processes for Multiple Servers

To start the multiple server configuration described in the previous section, you need to use multiple startup commands:

```
$ @SYS$STARTUP:APACHE$STARTUP START
$ @SYS$STARTUP:APACHE$STARTUP START APACHE$MYHOST_1
$ @SYS$STARTUP:APACHE$STARTUP START APACHE$MYHOST_2
```

After the servers have been started, you may want to write procedures to monitor the status of the servers. To examine the main server process for each configuration data file, you can use the APACHE$CONFIG READ command to define the APACHE$SERVER_PID logical name for each data file. If the logical name is a blank space, that means that the server process is down. Otherwise, the logical name is the process identifier of the parent process.

If the parent process is known, you can get the name of the process and look for all processes that begin with the parent process' name. For example, a DCL procedure might look similar to the following:

```
$ @SYS$MANAGER:APACHE$CONFIG READ APACHE$MYHOST_1
$ parent_pid = f$trnlnm("apache$server_pid")
$ if (parent_pid .eqs. " ")
$ then
    [server is not running]
$ endif

    . . .
$ parent_prcnam = f$getjpi(parent_pid,"prcnam")
$ parent_uic = f$getjpi(parent_pid,"uic")
$ context = ""
$ t1 = f$context("process", context, "uic", parent_uic, "eql")
$find_next_process:
$ pid = f$pid(context)
$ if (pid .nes. "")
$ then
$   t1 = f$getjpi(pid,"prcnam")
$   if (f$locate(parent_prcnam,t1) .eq. 0)
$   then

        . . .

$   endif
$   goto find_next_process
$ endif
```

2.3.7.3 LOGIN.COM

Each server process runs the optional APACHE$ROOT:[000000]LOGIN.COM. Sometimes it is necessary for the LOGIN.COM procedure to be interpretted differently for one set of web server processes than another set of processes. There are two ways to accomplish this.

One method is to build and maintain a separate LOGIN.COM file for a given configuration and place it in the configuration-specific directory (APACHE$SPECIFIC:[000000]LOGIN.COM). All of the web server processes associated with the related configuration data file will use that LOGIN.COM file. All other processes will use some other LOGIN.COM.

Another method is to use a single, clusterwide LOGIN.COM (APACHE$COMMON:[000000]LOGIN.COM), but include conditional statements to test for known values for the APACHE$SPECIFIC logical name, or check the APACHE$SERVER_TAG logical name for your server tag. For example:

```
$ t1 = f$trnlnm("apache$specific")
$ t2 = "MYHOST$DKA0:[APACHE.SPECIFIC.MYHOST_1.]"
$ if (t1 .eqs. t2) then . . .
```

2.3.8 Viewing the Certificate

You need a valid server certificate to run the Secure Web Server in SSL mode. Configuration creates a self-signed certificate and installs it. If you want to view the certificate before starting the server, use the OpenSSL Certificate Tool as described in the *Secure Web Server SSL User Guide.*

After configuring the Secure Web Server, do not start the server. Follow the instructions in the Section 2.4.

2.4 Post Configuration Checklist

After you configure the Secure Web Server, perform the following tasks to ensure a successful startup:

1. Configure CSWS_JAVA, if you have just installed it.
2. You can (optionally) check the CSWS_PERL configuration now or later.
3. You can (optionally) check the CSWS_PHP configuration now or later.
4. Run AUTOGEN.
5. Check disk quota.
6. Check for SET TERMINAL/INQUIRE.

Each of these tasks is explained below. Once you have completed them, you can test the installation by starting the Secure Web Server.

2.4.1 Configure CSWS_JAVA

If you installed the CSWS_JAVA module, you must configure it before you can start the server. For instructions, see the *CSWS_JAVA for HP Secure Web Server for OpenVMS Installation Guide and Release Notes.*

2.4.2 Check the CSWS_PERL Configuration

You are not required to configure CSWS_PERL before starting the server. CSWS_PERL is preconfigured with default values that let you execute PERL scripts. If you want to change the default configuration, you should do so now, before you start the server. To change the default configuration, edit APACHE$ROOT:[CONF]MOD_PERL.CONF. For more information, see the *CSWS_PERL for HP Secure Web Server for OpenVMS Installation Guide and Release Notes*.

2.4.3 Check the CSWS_PHP Configuration

You are not required to configure CSWS_PHP before starting the server. CSWS_PHP is preconfigured with default values. If you want to change the default configuration, you should do so now, before you start the server. To change the default configuration, edit APACHE$ROOT:[CONF]MOD_PHP.CONF. For more information, see the *CSWS_PHP for HP Secure Web Server for OpenVMS Installation Guide and Release Notes*.

2.4.4 Run AUTOGEN

After the installation, run SYS$UPDATE:AUTOGEN.COM (AUTOGEN) to evaluate your system parameters and make adjustments based on your hardware configuration and system workload. On the Secure Web Server for OpenVMS, AUTOGEN will probably increase the page file size and the number of swap file pages.

2.4.5 Check Disk Quota

If the disk quota is too low, the Secure Web Server will not start. Either raise the disk quota for the user account APACHE$WWW, or grant the account the EXQUOTA privilege, thus allowing it to bypass disk quota restrictions. Use the following commands:

```
$ SHOW QUOTA/USER=[server-uic]/DISK=device-name

$ SET PROCESS/PRIVILEGES=EXQUOTA node-name::APACHE$WWW
```

2.4.6 Check for SET TERMINAL/INQUIRE

When the Secure Web Server for OpenVMS is started, the following login files are executed:

- SYLOGIN.COM (system login file)
- LOGIN.COM (login file for APACHE$WWW)

Check these files to make sure that any SET TERMINAL/INQUIRE statements are executed only in INTERACTIVE mode. For example:

```
$ IF F$MODE() .eqs "INTERACTIVE" then $ SET TERMINAL/INQUIRE
```

Failure to do so might result in ill-formed HTML intermittently being returned to clients. This problem might also appear when executing CGI scripts.

2.5 Test the Installation

Now you will manually start the Secure Web Server to verify the installation and configuration of the server. Enter the following command:

```
$ @SYS$STARTUP:APACHE$STARTUP
```

2.5.1 Browser Test

You can test the installation using your web browser. Replace *host.domain* in the following URL with the information for the Secure Web Server you just installed:

```
HTTP://host.domain/
```

If this is a new installation, the browser should display the standard introductory page with the following bold text at the top:

```
"Hey, it worked !
 The SSL/TLS-aware Apache webserver was
 successfully installed on this website."
```

The Apache logo is displayed at the bottom.

2.5.2 TELNET Test

You can also use TELNET on the local host to test the installation. Use the following procedure to test the installation:

1. Enter the following command:

   ```
   $ TELNET 0 80
   ```

2. The following text is displayed:

   ```
   %TELNET-I-TRYING, Trying ... 127.0.0.1
    %TELNET-I-SESSION, Session 01, host localhost, port 80
    -TELNET-I-ESCAPE, Escape character is ^]
   ```

3. Press ENTER and enter the following HTTP command:

   ```
   HEAD / HTTP/1.0
   ```

4. Press ENTER **twice**.
   Text similar to the following is displayed:

```
HTTP/1.1 200 OK
Date: Tue, 21 September 2004 17:05:05 GMT
Server: Apache/1.3.26 (OpenVMS)
Last-Modified: Mon, 20 September 2004 15:33:27 GMT
ETag: "33dfec-681-39295347"
Accept-Ranges: bytes
Content-Length: 1665
Connection: close
Content-Type: text/html

%TELNET-S-REMCLOSED, Remote connection closed
-TELNET-I-SESSION, Session 01, host localhost, port 80
```

You should receive several lines of text from the Secure Web Server.

2.5.3 Troubleshooting

If you do not receive a response from the Secure Web Server, check the following:

- Look in your SYLOGIN.COM file and make sure there is no SET TERMINAL/INQUIRE statement for NETWORK processes.
- Make sure the APACHE$WWW account exists and is not disabled.
- Look for the following files:
  - APACHE$ROOT:[000000]APACHE$$SERVER.LOG
  - APACHE$ROOT:[LOGS]ERROR_LOG.

2.6 What's Next

After you have successfully tested the installation, perform any of the following tasks that are relevant for you:

- If you are upgrading from a previous version or beta kit of the Secure Web Server, you can merge the previous versions of files commonly modified by system administrators with the newly installed versions of these files. See Section 2.7.

- If you enabled MOD_SSL, follow the instructions for verifying SSL in the *Secure Web Server SSL User Guide.*
- Read Chapter 3 for information on starting and stopping the server, using HTTPD.CONF to customize the server environment, and other OpenVMS specific topics.

2.7 Merge Changes to Files You Have Customized (Upgrade Customers Only)

If you have installed a previous version or beta kit of the Secure Web Server, it is removed automatically before the new kit is installed.

When the previous version of the Secure Web Server is removed, the PCSI utility removes only the files and directories it installed. Any files you have created are not affected.

Note

Files installed by the Secure Web Server that are commonly modified by system administrators are *not removed*. However, the new kit contains updated versions of these files. Be sure to transfer any edits you made to the previous versions of these files to the new versions.

These commonly modified files are as follows:

- [APACHE]LOGIN.COM
- [APACHE.HTDOCS]INDEX.HTML
- [APACHE.CONF]HTTPD.CONF

If you modified the file [APACHE.CONF]MIME.TYPES, you need to copy the file to another location before you begin the installation. This file is removed during the installation. (HP recommends that you use the AddTypes directive instead of modifying the MIME.TYPES file.)

The new kit contains an updated version of this file. After you save your current version, restore the file and incorporate your local modifications with the new version.

Chapter 3
Running the Secure Web Server on OpenVMS

In general, you can run the Secure Web Server on OpenVMS as you would run Apache with MOD_SSL on any platform. However, there are some exceptions. This chapter describes the functions that behave differently or are not available, as well as any enhancements that are specific to OpenVMS.

3.1 Starting and Stopping the Server

Starting and stopping the Secure Web Server requires enhanced privileges (DETACH, SYSNAM, WORLD, etc.). Start and stop the server from a privileged account such as SYSTEM.

3.1.1 Starting the Server

Start the Secure Web Server with the following command:

```
    $ @SYS$STARTUP:APACHE$STARTUP [startup-value] [configuration-file]
```

Startup-value is optional and can have the following values:

Value	Description
START	Creates the Secure Web Server as a detached network process; default value
GRACEFUL	Sends a restart signal to the server, but existing client connections are not interrupted. Idle child processes are immediately deleted and replaced. Busy child processes are replaced when the connection is terminated
RESTART	Sends a restart signal to the server to have it reread APACHE$ROOT:[CONF]HTTPD.CONF
RUN	Runs the server on the current process

Configuration-file is an optional file specification of a configuration file built and maintained by APACHE$CONFIGURE.COM. The default value is SYS$MANAGER:APACHE$CONFIG.DAT.

To automate the startup of the Secure Web Server when the system is booted, add the following commands to the SYS$MANAGER:SYSTARTUP_VMS.COM file:

```
    $ FILE := SYS$STARTUP:APACHE$STARTUP.COM
    $ IF F$SEARCH("''FILE'") .NES. "" THEN @'FILE'
```

3.1.2 Stopping the Server

You can shut down the Secure Web Server with the following command:

```
    $ @SYS$STARTUP:APACHE$SHUTDOWN [startup-value] [configuration-file]
```

Startup-value is optional and can have the following values:

Value	Description
GRACEFUL	Sends a restart signal to the server, but existing client connections are not interrupted. Idle child processes are immediately deleted and replaced. Busy child processes are replaced when the connection is terminated
RESTART	Sends a restart signal to the server to have it reread APACHE$ROOT:[CONF]HTTPD.CONF
SHUTDOWN	Stops the detached network process; default value
STOP	Same as SHUTDOWN

Configuration-file is an optional file specification of a configuration file built and maintained by APACHE$CONFIGURE.COM. The default value is SYS$MANAGER:APACHE$CONFIG.DAT.

To automate the shutdown of the Secure Web Server when the system is shut down, add the following commands to the SYS$MANAGER:SYSHUTDOWN.COM file:

```
$ FILE := SYS$STARTUP:APACHE$SHUTDOWN.COM
$ IF F$SEARCH("''FILE'") .NES. "" THEN @'FILE'
```

Note

The Secure Web Server will not shut down as long as the APACHE$WWW process is running. If you have a problem with shutting down the server, use the following command to see if APACHE$WWW processes are still running:

```
$ SHOW SYSTEM/OWNER_UIC=[APACHE$WWW]
```

Server processes can have two types of process name: APACHE$nn, where *nn* is an integer number, or APACHE$ssss, where *ssss* is a user-defined server tag. Similarly, child processes can have two types of process name: APACHE$nnmmm, where *APACHE$nn* is the server and *mmm* is an integer number, or APACHE$ssssmmm, where *ssss* is a user-defined server tag and *mmm* is an integer number.

If APACHE$WWW is still running, use the following command to stop it. You should then be able to shut down the server.

```
$ STOP PROCESS/ID=<apache-pid>
```

3.2 Server Log File

The server log file for APACHE$WWW is written to:

```
APACHE$SPECIFIC:[000000]APACHE$$SERVER.LOG
```

3.3 Performance Considerations

You should have prior experience tuning the performance of the OpenVMS operating system. For general information on OpenVMS performance, see the *OpenVMS Performance Management Manual* in the OpenVMS documentation website.

Recommendations for improving performance on a Secure Web Server are provided in the following sections.

3.3.1 Limits and Quotas

The following table shows sample values for the APACHE$WWW system user account (SYSUAF) from a working and exercised Secure Web Server with a light to moderate load. These values are presented as an example of a system performing well within its context. If you should experience performance difficulties, refer to this table for guidelines in making adjustments. For heavier loads, we point out which values, in our experience, need to be increased as load increases. Keep in mind that no one set of values will be appropriate for all situations.

Table 3-1 Sample Values for the APACHE$WWW SYSUAF

Parameter	Default	On Secure Web Server
ASTLM (NonPooled) Total number of asynchronous system trap (AST) operations and scheduled wake-up requests the user can have queued at one time	250	610 Or BIOLM + DIOLM + 10
BIOLM (NonPooled) Number of outstanding buffered I/O operations permitted for a user's process	150	300 You might also need to increase the SYSGEN parameter CHANNELCNT because it limits BIOLM,DIOLM, and FILLM.
BYTLM (Pooled) Amount of buffer space a user's process can use	64000	200000 Increase this value for a heavy load.
DIOLM (NonPooled) Number of outstanding direct I/O operations permitted to a user's process	150	300 You might also need to increase the SYSGEN parameter CHANNELCNT because it limits BIOLM,DIOLM, and FILLM.
ENQLM (Pooled) Specifies the lock queue limit	2000	2000
FILLM (Pooled) Number of files a user's process can have opened at one time. Includes the number of network logical links that can be active at the same time	100	300 Increase this value for a heavy load. You might also need to increase the SYSGEN parameter CHANNELCNT because it limits BIOLM,DIOLM, and FILLM.
JTQUOTA (Pooled) Byte quota for the jobwide logical name table	4096	8192
PGFLQUO (Pooled) Number of pages the user's process can use in the system page file	50000	250000 If you increase PGFLQUO, you should monitor the free size of the system page and swap files; they may need to be increased.
PRCLM (Pooled) Number of subprocesses a user's process can	8	20 You should increase this value for a heavy load.

create		
TQELM (Pooled) Number of entries a user's process can have in the timer queue or the number of temporary common event flag clusters a user's process can have	10	610 Or BIOLM + DIOLM + 10

To change the quotas for the APACHE$WWW SYSUAF, use the system manager account and run the AUTHORIZE utility. For example:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
UAF> SHOW APACHE$WWW
Username: APACHE$WWW                    Owner: APACHE WEBSERVER
   ...
Maxjobs:       0 Fillm:     100 Bytlm:      64000
Maxacctjobs:   0 Shrfillm:    0 Pbytlm:         0
Prclm:         8 DIOlm:     150 WSdef:       2000
   ...
UAF>  MODIFY APACHE$WWW/FILLM=300/PRCLM=20
%UAF-I-MDFYMSG, user record(s) updated
UAF>  EXIT
$
```

3.3.2 Server Experiencing Medium to High Usage

After you install the server and have been running it, look in the log file for errors of the "cannot open" variety. Errors of this type often indicate you need to modify system parameters. Try the following:

- Set FILLM to limit the number of files a user's process can have open.
- Set the SYSGEN parameter CHANNELCNT to 1024 (unless it is already set to a higher value).

Note

Whenever you change system parameters, you must reboot the system to enable the new settings.

3.3.3 Global Pages and Global Sections

If a browser installation stalls, this could be an indication that the number of global pages or global sections is too low. Run AUTOGEN to evaluate the number of global pages and global sections you need. Some browsers might need more.

3.3.4 Excessive File Build Up

A large number of .LOG and .PID files can amass over time in the directories APACHE$ROOT:[000000] and APACHE$ROOT:[LOGS]. Purging these files can become a burden on application or system managers.

System managers should manually use explicit SET DIRECTORY/VERSION commands on these two directories.

3.4 Customizing the Server Environment with HTTPD.CONF

The installation procedure creates a file named HTTPD.CONF and places it in APACHE$ROOT:[CONF]. The HTTPD.CONF file stores information that the Secure Web Server uses to set up the server environment. HTTPD.CONF has been tailored to use OpenVMS syntax, but its overall functionality is essentially identical to HTTPD.CONF on the UNIX platform.

HTTPD.CONF contains an explanation for each line that it can execute. You can refer to these explanations when customizing the file for your environment. You can also refer to any generally available Apache documentation on HTTPD.CONF.

Note the following about HTTPD.CONF on OpenVMS:

- No directives have been deleted or added to the Apache template except an Include directive for MOD_SSL. Installing CSWS_JAVA, CSWS_PHP, or CSWS_PERL will also append Include directives specific to these modules.
- MOD_OSUSCRIPT has been added to enable CGI scripts originally written for the OSU server.
- MOD_AUTH_OPENVMS enables authentication using OpenVMS usernames and passwords.
- UNIX style path names are recognized by OpenVMS. You can use either UNIX style or OpenVMS style path names in the configuration file. However, you cannot intermix the two styles within a specification. HP recommends UNIX style path names.
- In an OpenVMS cluster, you can specify either clusterwide or system-specific files. For more information, see Individual System vs. Clusterwide Definition.

Note

Beginning with Version 1.3-1, the _alpha suffix has been removed from image filename extensions. These filenames are referenced in the *.conf files read during Apache startup. Because you are allowed to customize the *.conf files, the CSWS installation procedure does not replace the files during an upgrade. **You must manually edit any *.conf procedure following the CSWS V1.3-1 upgrade and change the *.exe_alpha references to *.exe.**

Examples of filenames in Version 1.3 and earlier:

```
apache$common:[modules]mod_auth_openvms.exe_alpha
apache$common:[modules]mod_dav.exe_alpha
apache$common:[modules]mod_ssl.exe_alpha
```

Examples of filenames in Version 1.3-1 and later:

```
apache$common:[modules]mod_auth_openvms.exe
apache$common:[modules]mod_dav.exe
apache$common:[modules]mod_ssl.exe
```

3.5 Running Multiple Servers

If your web server consistently experiences high load, you may need to run multiple servers (which is running more than one server process on a given system).

3.5.1 Defining Multiple Configurations

To configure and start multiple servers, you must first create and maintain multiple APACHE$CONFIG.COM configuration files.

The DCL command procedure APACHE$CONFIG.COM accepts two optional parameters. The first parameter is the verb, and the default is CONFIGURE. The second parameter is a configuration file, and the default is SYS$MANAGER:APACHE$CONFIG.DAT. To create a new APACHE$CONFIG.COM configuration file, specify some other file specification for the second parameter.

For example, the server defined by the file SYS$MANAGER:APACHE$CONFIG.DAT for the system called MYHOST, will, by default, write its system-specific files in the directory:

```
device:[directory.APACHE.SPECIFIC.MYHOST]
```

where the CSWS kit was installed in *device:[directory]*.

To create another server configuration, perform the following steps.

- Run APACHE$CONFIG.COM by entering a command similar to the following:

  ```
  $ @SYS$MANAGER:APACHE$CONFIG CONFIGURE
  ```

- Specify a new system-specific directory for that configuration:

  ```
  device:[directory.APACHE.SPECIFIC.MYHOST_1]
  ```

- Edit the HTTPD.CONF configuration file for that system-specific directory:

  ```
  $ EDIT APACHE$ROOT:[CONF]HTTPD.CONF
  ```

  (For example, this server must listen on a different port number than the main server for that system.)

APACHE$CONFIG.COM creates the system-specific directory and the necessary subdirectories (such as [.HTDOCS] and [.CONF]). You do not need to create these directories yourself.

3.5.2 Multiple Installations

During the configuration dialog, the Secure Web Server assumes that the same directory will be used for the clusterwide files. In other words, all of the servers should use the images from the same location. HP does not recommend installing different versions of the Secure Web Server in different locations and then running two or more versions of the software on the same system at the same time.

The configuration and startup procedures must install certain shareable image libraries systemwide. If different versions of the Secure Web Server are started on the same system, some server processes will fail by attempting to run with incompatible shareable images.

3.5.3 Starting and Stopping Multiple Servers

To start the server for this configuration, enter the following command:

```
$ @SYS$STARTUP:APACHE$STARTUP START SYS$MANAGER:APACHE$MYHOST_1.DAT
```

To shutdown this server, enter:

```
$ @SYS$STARTUP:APACHE$SHUTDOWN SHUTDOWN SYS$MANAGER:APACHE$MYHOST_1.DAT
```

If the system has three such servers, the system startup file might include the following:

```
$ @SYS$STARTUP:APACHE$STARTUP
$ @SYS$STARTUP:APACHE$STARTUP START SYS$MANAGER:APACHE$MYHOST_1.DAT
$ @SYS$STARTUP:APACHE$STARTUP START SYS$MANAGER:APACHE$MYHOST_2.DAT
```

Similarly, if you want to shut down the Secure Web Server, you need to stop all three servers, as follows:

```
$ @SYS$STARTUP:APACHE$SHUTDOWN
$ @SYS$STARTUP:APACHE$SHUTDOWN SHUTDOWN SYS$MANAGER:APACHE$MYHOST_1.DAT
$ @SYS$STARTUP:APACHE$SHUTDOWN SHUTDOWN SYS$MANAGER:APACHE$MYHOST_2.DAT
```

3.6 Modules and Directives

3.6.1 Apache Modules

Following is a list of the modules included in the Secure Web Server kit. The list shows the directives supported in each module. All supported modules and directives function as documented by the Apache Software Foundation. The server documentation from the Apache Software Foundation provides the information needed to use the modules and directives.

HTTP_CORE.C

AccessConfig
AccessFileName
AllowOverride
AuthName
AuthType
BindAddress
CoreDumpDirectory
DefaultType
<Directory>
<DirectoryMatch>
DocumentRoot
ErrorDocument
ErrorLog
<Files>
<FilesMatch>
HostnameLookups
IdentityCheck
<IfDefine>
<IfModule>
Include
KeepAlive
KeepAliveTimeout
<Limit>
<LimitExcept>
LimitRequestBody
LimitRequestFields
LimitRequestLine
Listen
ListenBacklog
<Location>
<LocationMatch>

LogLevel
MaxClients
MaxKeepAliveRequests
MaxRequestPerChild
MaxSpareServers
MinSpareServers
NameVirtualHost
Options
PidFile
Port
Require
ResourceConfig
Satisfy
SendBufferSize
ServerAdmin
ServerAlias
ServerName
ServerPath
ServerRoot
ServerSignature
ServerTokens
ServerType
StartServers
TimeOut
UseCanonicalName
VirtualHost

MOD_ACCESS.C

allow
deny
order

MOD_ACTIONS.C

Action
Script

MOD_ALIAS.C

Alias
AliasMatch
Redirect
RedirectMatch
RedirectTemp
RedirectPermanent
ScriptAlias
ScriptAliasMatch

MOD_ASIS.C

MOD_AUTH.C

AuthGroupFile
AuthUserFile

MOD_AUTH_OPENVMS.C *OpenVMS specific*

MOD_AUTOINDEX.C

AddAlt
AddAltByEncoding
AddAltyByType
AddDescription
AddIcon
AddIconByEncoding
AddIconByType
DefaultIcon
FancyIndexing
HeaderName
IndexIgnore
IndexOptions
IndexOrderDefault
ReadmeName

MOD_CGI.C

ScriptLog
ScriptLogBuffer
ScriptLogLength

MOD_DAV.C

DAV
DAVLockDB
DAVMinTimeout
DAVDepthInfinity
DAVParam
LimitXMLRequestBody

MOD_DEFINE.C

DefineTagOpenVMS *OpenVMS specific. Allows you to redefine the Define tag to another character (for example, an ampersand (&)) instead of the default dollar sign ($).*

MOD_DIR.C

DirectoryIndex

MOD_ENV.C

SetEnv
UnsetEnv

MOD_IMAP.C

ImapBase
ImapDefault
ImapMenu

MOD_INCLUDE.C

MOD_INFO.C

AddModuleInfo

MOD_LOG_CONFIG.C

CustomLog
LogFormat
TransferLog

MOD_MIME.C

AddCharset
AddEncoding
AddHandler
AddLanguage
AddType
DefaultLanguage
ForceType
RemoveHandler
SetHandler
TypesConfig

MOD_NEGOTIATION.C

CacheNegotiatedDocs
LanguagePriority

MOD_PROXY.C

ProxyRequests
ProxyRemote
ProxyPass
ProxyPassReverse
AllowConnect
ProxyBlock
ProxyReceiveBufferSize
NoProxy
ProxyDomain
ProxyVia
CacheRoot
CacheSize
CacheMaxExpire
CacheDefaultExpire
CacheLastModifiedFactor
CacheGcInterval
CacheDirLevels
CacheDirLength
CacheForceCompletion
NoCache

MOD_REWRITE.C

RewriteEngine
RewriteOptions
RewriteLog

RewriteLogLevel
RewriteLock
RewriteMap
RewriteBase
RewriteCond
RewriteRule

MOD_OSUSCRIPT.C *OpenVMS specific*

MOD_SETENVIF.C

BrowserMatch
BrowserMatchNoCase
SetEnvIf
SetEnvIfNoCase

MOD_SO.C

LoadModule

MOD_STATUS.C

ExtendedStatus

MOD_UNIQUE_ID.C

MOD_USERDIR.C

UserDir

3.7 Supported and Unsupported Features

The server documentation from the Apache Software Foundation provides most of the information needed to run your Secure Web Server for OpenVMS. Information specific to the OpenVMS operating system is provided below.

3.7.1 Modules Not Included

The following modules are **not** included in this version of the Secure Web Server for OpenVMS.

MOD_AUTH_ANON
MOD_AUTH_DB
MOD_AUTH_DBM
MOD_AUTH_DIGEST
MOD_CERN_META
MOD_DIGEST
MOD_EXAMPLE
MOD_EXPIRES
MOD_HEADERS
MOD_ISAPI
MOD_LOG_AGENT
MOD_LOG_REFERER
MOD_MIME_MAGIC
MOD_MMAP_STATIC
MOD_SPELING
MOD_USERTRACK
MOD_VHOST_ALIAS

3.7.2 Unsupported Directives

The following directives are **not** supported.

AgentLog
Anonymous
Anonymous_Authoritative
Anonymous_LogEmail
Anonymous_MustGiveEmail
Anonymous_NoUserID
Anonymous_VerifyEmail
AuthDBAuthoritative
AuthDBGroupFile
AuthDBMAuthoritative
AuthDBMGroupFile
AuthDBUserFile
AuthDBMUserFile
AuthDigestFile
CheckSpelling
CookieExpires
CookieTracking
Example
ExpiresActive
ExpiresByType
ExpiresDefault
Header
Metadir
MetaFiles
MetaSuffix
MimeMagicFile
MMapFile
RefererIgnore
RefererLog
RLimitCPU
RLimitMEM
RLimitNPROC
ScriptInterpreterSource
VirtualDocumentRoot
VirtualDocumentRootIP
VirtualScriptAlias
VirtualScriptAliasIP

3.7.3 Command Line Options

This section describes the HTTPD command line options supported in the Secure Web Server. You must define HTTPD as a symbol before you can use the HTTPD command line options, as shown below. (If needed, you can define HTTPD in SYS$MANAGER:LOGIN.COM.)

```
    $ HTTPD :== $APACHE$ROOT:[000000]APACHE_HTTPD.EXE
```

Then you can use the following format to enter a command line option:

```
    $ HTTPD -option
```

where *-option* is one of the following:

Table 3-2 HTTPD Command Line Options

Option	Description

-v	Display the HTTPD version and its build date
-“V”	Display the HTTPD base version, its build date, and a list of compile settings that influence the behavior and performance of the server
-h	Display a list of the HTTPD options
-l	Display a list of all modules compiled into the server
-“L”	Display a list of directives with expected arguments and places where the directive is valid

The following example shows how to enter the L option to list the available configuration directives:

```
$ httpd -“L”
```

3.7.4 Virtual Host Support

The term *virtual host* refers to the practice of maintaining a single server to serve pages for multiple virtual hosts. Both IP-based and name-based virtual host support are available in the Secure Web Server for OpenVMS.

Note

On OpenVMS, the security profile of the running server is the same on all virtual hosts.

For more information, see Apache Virtual Host Documentation at http://httpd.apache.org/docs/vhosts/index.html.

3.7.5 Dynamic Shared Object Support

Dynamic shared object support provides a way to format code so that it will load into the address space of an executable program at run time. This functionality is supported on OpenVMS.

For more information, see Dynamic Shared Object (DSO) Support at http://httpd.apache.org/docs/dso.html.

3.7.6 File Handlers

The Secure Web Server for OpenVMS supports the ability to use file handlers explicitly.

For more information, see Apache's Handler Use Documentation at http://httpd.apache.org/docs/handler.html.

3.7.7 Content Negotiation

The MOD_NEGOTIATION module provides content negotiation. This module lets you specify language variants of HTML files. To specify language variants on OpenVMS, use an underscore instead of a period before the language extension.

For example:

- On UNIX, *filename.html.fr* is the French variant of *filename.html*.
- On OpenVMS, *filename.html_fr* is the French variant of *filename.html*.

For more information, see Content Negotiation Documentation at http://httpd.apache.org/docs/content-negotiation.html.

3.7.8 Apache API

You can use the standard Apache API to write your own modules that will run on the Secure Web Server for OpenVMS. For more information, see Apache API Documentation at http://httpd.apache.org/docs/misc/API.html.

3.7.9 MOD_DAV (Distributed Authoring and Versioning) Support

The Secure Web Server for OpenVMS includes MOD_DAV to provide DAV (Distributed Authoring and Versioning) capabilities.

DAV is a set of extensions to the HTTP protocol that allows users to collaboratively edit and manage files on remote web servers using DAV-enabled client applications. MOD_DAV is an Apache module that provides DAV capabilities (RFC 2518) for the Apache web server.

For more information, see the MOD_DAV website at http://www.webdav.org/mod_dav/.

3.7.9.1 MOD_DAV Hardware Requirements

DAV file names and collection names use an extended character-set name-space. Therefore, an ODS-5 disk is required for DAV storage.

3.7.9.2 Record Attributes on DAV-Served Files

Only STREAM-LF files are supported by DAV.

3.7.9.3 Configuring MOD_DAV

To use DAV on OpenVMS, you must configure MOD_DAV and enable extended file support for the Secure Web Server.

3.7.9.3.1 Enabling Extended Filename Support

Enable extended file specification features in the C Run-Time Library for the Secure Web Server by adding the following lines to APACHE$ROOT:[000000]LOGIN, above the "$ exit" statement.

```
$ define decc$argv_parse_style enable
$ define decc$efs_case_preserve enable
$ define decc$efs_charset enable
$ define decc$efs_case_special enable
$ define decc$enable_getenv_cache enable
$ define decc$posix_seek_stream_file enable
```

3.7.9.3.2 Configuration Examples

Following is a simple MOD_DAV configuration using the Location container to control DAV operations in a URL directory named webdav, which is located under the server's document directory.

```
LoadModule dav_module /apache$common/modules/mod_dav.exe

DAVLockDB /apache$root/var/davlock
```

```
DAVMinTimeout 600

<Location /webdav>
      DAV On
</Location>
```

For this configuration, you must create two directories owned by APACHE$WWW (or allowing read/write access by APACHE$WWW):

```
$ create/dir/own=apache$www apache$common:[var]
$ create/dir/own=apache$www apache$common:[htdocs.webdav]
```

Following is a configuration that requires OpenVMS user authentication before write access is granted (mod_auth_openvms must be loaded):

```
LoadModule dav_module /apache$common/modules/mod_dav.exe

DAVLockDB /apache$root/var/davlock
DAVMinTimeout 600

<Location /webdav>
 DAV on
 AllowOverride None
 <Limit PUT POST DELETE PROPFIND PROPPATCH MKCOL COPY MOVE LOCK UNLOCK>
        AuthType Basic
        AuthName "WebDAV"
        AuthOpenVMSAuthoritative On
        Require valid-user
 </Limit>
</Location>
```

3.7.9.4 Testing DAV Operation

You can test DAV operation using Microsoft Internet Explorer.

Recent versions of Windows come DAV-enabled. Older versions, such as Windows NT 4.0, require Internet Explorer 5.5 or later. You may need to DAV-enable Internet Explorer 5.5 using the Control Panel application's Add/Remove Programs option. Double-click on Microsoft Internet Explorer 5.5 and Internet Tools, and select Add Component and click OK. Scroll down to Web Publishing Components and select Web Folders and complete the configuration.

Run Internet Explorer and select File/Open. Select "Open as web folder" and enter the URL of the DAV folder (for example: http://your-server/webdav/). Internet Explorer will open your DAV folder and display its contents. In addition, Windows Explorer may now show a Web Folders folder that you can use like a local or network drive.

Note

Not all Windows applications are DAV-enabled. Recent versions of Microsoft Word and Microsoft Office are DAV-enabled, which may allow you to open and edit a DAV document directly in the application. For other applications, you may need to copy the DAV file to a local directory in Windows, edit the document and copy it back to the DAV folder using Windows Explorer.

3.7.10 suEXEC Support

The suEXEC feature provides the ability to run CGI programs under user IDs different from the user ID of the calling web server.

This feature is implemented in the Secure Web Server beginning with the Version 1.3 release.

For more information about suEXEC, see http://httpd.apache.org/docs/suexec.html.

See Section 4.7 for suEXEC configuration information.

3.5.11 Running MOD_OSUSCRIPT

The Secure Web Server for OpenVMS provides a CGI script environment. However, it also includes MOD_OSUSCRIPT, an optional module that enables the server to run scripts that were written for the OSU http server's script environment (which is not CGI).

Note

In Secure Web Server Version 1.1 for OpenVMS and higher, the Apache logical names must be defined *systemwide* (not processwide) in order for MOD_OSUSCRIPT to work properly.

MOD_OSUSCRIPT does not need to communicate with a running OSU server to work properly. It needs only the following OSU http distribution files:

- WWWEXEC.COM
- CLI_ENV_RM.COM
- CGI_SYMBOLS.EXE
- MINIPERL.EXE

The WWW_SYSTEM:MINIPERL.EXE image is required unless you use a different Perl interpreter. To use the Perl interpreter provided with CSWS, define the following foreign command in APACHE$ROOT:[000000]LOGIN.COM:

```
$ PERL :== $PERL_ROOT:[000000]PERL.EXE
```

You can download and install these files to run and test OSU scripts, as follows:

1. Create a directory for the OSU script files with the following command. The APACHE$WWW username must be able to read this directory and its files.

   ```
   $ CREATE/DIRECTORY device:[directory.BIN]
   ```

2. Create a logical name pointing to the OSU script root directory, as follows:

   ```
   $ DEFINE/SYSTEM WWW_ROOT device:[directory.]/TRANS=CONCEAL
   ```

3. Copy CLI_ENV_RM.COM and CGI_SYMBOLS.EXE from an OSU system to the following location. (CLI_ENV_RM.COM can be extracted from the OSU source code distribution found at http://www.er6.eng.ohio-state.edu/www/targazer/, but CGI_SYMBOLS.EXE must be compiled and linked.)

```
WWW_ROOT:[BIN]
```

4. Copy WWWEXEC.COM from an OSU system to the following location, or extract it from the OSU source code distribution*:

```
APACHE$ROOT:[000000]
```

Note

The standard version of WWWEXEC.COM does not support the use of Alias/<Directory> directives. You must use a <Location> container with mod_osuscript.

---

5. Use the SHOW SECURITY command to ensure that APACHE$WWW can read these files. If needed, use the SET SECURITY command to change the security settings.

6. Edit HTTPD.CONF to add the following lines:

```
<Location /htbin>
       SetHandler osuscript-handler
       OSUscript 0::"0=WWWEXEC" www_root:[bin]
       Order allow,deny
       Allow from all
</Location>
```

7. Enter the following commands to add the DECnet proxy. Replace *node* with your DECnet Phase IV node name.

```
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
UAF>  ADD/PROXY node::APACHE$WWW APACHE$WWW/DEFAULT
UAF>  EXIT
```

If you are running DECnet-Plus, replace *namespace:.abc.xyz* with your system's full name, for example, DEC:.ZKO.NODE22::APACHE$WWW.

```
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
UAF> ADD/PROXY namespace:.abc.xyz::APACHE$WWW APACHE$WWW/DEFAULT
UAF> EXIT
```

8. To use MOD_OSUSCRIPT, you must load the image during Apache startup by uncommenting the following line in the APACHE$COMMON:[CONF]HTTPD.CONF file supplied by the installation (or add the uncommented line to your existing configuration file):

```
#LoadModule osuscript_module modules/mod_osuscript.exe
```

9. To test, execute the simple OSU script CLI_ENV_RM.COM. To do this, replace *myhostname* in the following URL with your server's domain name:

```
HTTP://myhostname/htbin/CLI_ENV_RM.COM
```

MOD_OSUSCRIPT does the following:

1. Opens a DECnet connection using task 0 for the APACHE$WWW username entered in the proxy command.
2. Executes WWWEXEC.COM using the default HTTPD.CONF directives.
3. Searches the WWW_ROOT:[BIN] directory for the script name entered in the URL.

MOD_OSUSCRIPT supports all of the script server protocol commands, except the following:

- <DNETREUSE> Reuse logical link for subsequent scripts.
- <DNETINVCACHE> Invalidate internal cache (the Secure Web Server does not have a cache).
- <DNETMANAGE> Send management command, OSU server specific.
- <DNETFORCEKA> Put client link in keep-alive mode.

3.8 File Formats

All file formats are supported. However, the Web browser status bar will not show page loading progress for Variable or VFC format files. Page loading progress relies on an accurate byte count. Accurate byte count is not readily available for files in Variable or VFC format.

3.9 File Naming Conventions

In general, users running the Secure Web Server for OpenVMS can specify either UNIX style file names or OpenVMS style file names. The Secure Web Server usually displays UNIX style file names.

The ODS-5 volume structure, introduced in OpenVMS Alpha Version 7.2, supports long file names, allows the use of a wider range of characters within file names, and preserves case within file names. However, the DEC C RTL shipped with OpenVMS Alpha Version 7.2 does not provide full support for extended file names on ODS-5 devices. This lack of full support imposes certain restrictions on users running the Secure Web Server for OpenVMS.

Because mixed UNIX and OpenVMS style extended file names are not yet supported by the DEC C RTL, you might be required to use UNIX style syntax when interacting with the Secure Web Server. An example would be appending additional directories or a file name to a root.

The following examples illustrate mixed UNIX and OpenVMS style file names that are not supported in OpenVMS Alpha Version 7.2:

```
 doc/foo.bar.bar

./tmp/foo.bar.b^_ar

 ~foo^.bar
```

You can, however, modify the last example so that it will work as an OpenVMS extended file name that has a tilde (~) as the first character. Precede the leading tilde (~) with the Extended File Specifications escape character (^). For example:

```
^~foo^.bar
```

Mixed UNIX and OpenVMS style file names will be supported in a future release of the DEC C RTL for OpenVMS.

3.9.1 ODS-5 Recommendations

There are several modules that behave differently on ODS-2 and ODS-5 disks. If you are using an ODS-5 disk for your web pages, HP recommends that you define the following logical names before starting the Secure Web Server. They can be placed in APACHE$ROOT:[000000]LOGIN.COM:

```
        $ define decc$efs_case_preserve enable
        $ define decc$efs_charset enable
        $ define decc$efs_case_special enable
```

These logical names affect mod_dav, mod_index, and http_core directives.

3.10 Managing File and Directory Access Controls

The OpenVMS operating system controls read/write/execute/delete access to files and directories at two levels. (See the *OpenVMS Guide to System Security* for more details on the OpenVMS security model.) These two levels are as follows:

- System:Owner:Group:World (SOGW) protection through user/group identifier codes (UIC) based on file ownership. This is conceptually similar to standard UNIX file protection.
- User-id/access pairs stored as Access Control Entries (ACE) within a file's (or directory's) Access Control List (ACL). ACLs can be used to provide fine-grained access control to files and directories that are not owned by the user who is attempting access.

When the Secure Web Server is installed, all files and directories under the APACHE$ROOT:[*...] directory tree are owned by the Secure Web Server UIC [AP_HTTPD,APACHE$WWW]. Most of these files and directories are given the SOGW protection of (S:RWED,O:RWED,G,W), which allows read+write+execute+delete access to system and owner UICs ([SYSTEM] and [AP_HTTPD,APACHE$WWW]). This allows the Secure Web Server server processes to access their own files and directories.

There are instances when you will need to modify file and directory protections. In most cases, you should use ACLs to grant or deny access, because ACLs provide you better control and security. In general, you should not modify file and directory UICs unless there is good reason to do so.

The following sections explain situations when you need to modify file and directory protections.

3.7.1 Outbound Access to Non-CSWS Files and Directories

If, for example, you allow users to host document files from their home directories (see the UserDir directive for more information), and you have the following directive defined in your HTTPD.CONF file:

```
    UserDir public_html
```

In this case, a user is allowed to define a subdirectory called *public_html* under their default login directory (for example, [JOE.PUBLIC_HTML]). For Secure Web Server to serve those documents from that directory, it needs access to that directory and its contents, as well as all upper-level directories leading to it.

The correct way to accomplish this for user "Joe" is as follows:

```
    $ SET SECURITY/ACL=(IDENT=APACHE$WWW,ACCESS=READ) -
    _$ dev:[000000]JOE.DIR
    $ SET SECURITY/ACL=(IDENT=APACHE$WWW,ACCESS=READ+EXECUTE) -
    _$ dev:[JOE]PUBLIC_HTML.DIR
    $ SET SECURITY/ACL=(IDENT=APACHE$WWW,ACCESS=READ+EXECUTE) -
    _$ dev:[JOE.PUBLIC_HTML]*.*
```

3.7.2 Inbound Access to CSWS Files and Directories

When accessing Secure Web Server files and directories from another user account (UIC), make sure the ACL for the target directory and file allows access by that UIC.

For example, if you are performing a File Transfer Process (FTP) operation to transfer new files to a CSWS directory, protect the files as follows:

```
    $ SET SECURITY/ACL=(IDENTIFIER=yourFTPname,ACCESS=READ+WRITE) -
    _$ [directory]
    $ SET SECURITY/ACL=(IDENTIFIER=yourFTPname,ACCESS=READ+WRITE) -
    _$ [directory]*.*
```

3.11 Logical Names

The Secure Web Server for OpenVMS creates the following logical names.

Table 3-3 System Defined Logical Names

Logical Name	Description
APACHE$FIXBG	System executive mode logical name pointing to installed, shareable images. Not intended to be modified by the user.
APACHE$INPUT	Used by CGI programs for PUT/POST methods of reading the input stream.
APACHE$PLV_ENABLE_<username>	System executive mode logical name defined during startup and used to control access to the services provided by the APACHE$PRIVILEGED image. Not intended to be modified by the user.
APACHE$PRIVILEGED	System executive mode logical name pointing to installed, shareable images. Not intended to be modified by the user.

Table 3-4 Process Logical Names

Logical Name	Description
APACHE$COMMON	Concealed logical name that defines clusterwide files in APACHE$ROOT (device:[APACHE]).
APACHE$SPECIFIC	Concealed logical name that defines system-specific files in APACHE$ROOT (device:[APACHE.SPECIFIC.node-name]).
APACHE$ROOT	System executive mode logical name defined during startup that points to the top-level directory. (device:[APACHE], device:[APACHE.SPECIFIC.node-name]).
APACHE$SERVER_PID	Created by the APACHE$CONFIG.COM READ function to identify the running server process that corresponds to a particular configuration file. This logical name is equivalent to the process ID of the running server, or a blank space (" ") if there is no running server for the specified configuration file.
APACHE$SERVER_STARTUP	File specification of a user-defined procedure to execute before activating the Apache server image. This procedure runs after APACHE$ROOT:[000000]LOGIN.COM with APACHE$COMMON and APACHE$SPECIFIC logical names defined. If no startup procedure is defined, the default startup procedure is APACHE$ROOT:[000000]APACHE$SERVER_STARTUP.COM.
APACHE$SERVER_SHUTDOWN	File specification of a user-defined procedure to execute after the Apache

	server image has terminated. If no shutdown procedure is defined, the default shutdown procedure is APACHE$ROOT:[000000]APACHE$SERVER_SHUTDOWN.COM.
APACHE$SERVER_TAG	A string of up to four characters, beginning with a letter, which is used to associate a name with a particular configuration. The string appears in the process names of the Apache server processes. For example, in the process name APACHE$xxxxyyy, xxxx is the tag string and yyy is the 3-digit child server number assigned by Secure Web Server. If no tag is defined, the Apache server process name is APACHE$xxyyy, where xx is the 2-digit server number assigned by Secure Web Server.

Note

In the Secure Web Server V1.1 for OpenVMS and higher, the APACHE$COMMON, APACHE$SPECIFIC, and APACHE$ROOT logical names are no longer defined systemwide. They are process logical names defined for each server process and its child processes.

Because the Secure Web Server supports multiple server processes on the same system, each server process can have its own definitions for these logical names. Therefore, these logical names cannot be made systemwide.

The process logical names are made by APACHE$CONFIG.COM, APACHE$STARTUP.COM, and APACHE$SHUTDOWN.COM. You can edit a clusterwide version of HTTPD.CONF by entering the following command:

```
    $ EDIT APACHE$COMMON:[CONF]HTTPD.CONF
```

If you attempt to use the preceding command on a process that has not run APACHE$CONFIG.COM, APACHE$STARTUP.COM, or APACHE$SHUTDOWN.COM, the command will fail because the logical name APACHE$COMMON is not defined for that process.

Note

There are several modules that behave differently on ODS-2 and ODS-5 disks. If you are using an ODS-5 disk for your web pages, HP recommends that you define the following logical names before starting the Secure Web Server. They can be placed in APACHE$ROOT:[000000]LOGIN.COM:

```
         $ define decc$efs_case_preserve enable
         $ define decc$efs_charset enable
         $ define decc$efs_case_special enable
```

These logical names affect mod_dav, mod_index, and http_core directives.

Table 3-5 User Defined Logical Names

Logical Name	Description
APACHE$CGI_BYPASS_OWNER_ CHECK	If defined to any value, this logical name causes the Secure Web Server to bypass the file owner check of the CGI script file. The

	default is to enforce the owner check on CGI script files for security purposes.
APACHE$CGI_MODE	System logical name that controls how CGI environment variables are defined in the executing CGI process. There are three different options. Note that only one option is available at a time. 0 — Default. Environment variables are defined as local symbols and are truncated at 970 (limitable with DEC C). 1 — Environment variables are defined as local symbols unless they are greater than 970 characters. If the environment value is greater than 970 characters, it is defined as a multi-item logical. 2 — Environment variables are defined as logicals. If the environment value is greater than 255 characters, it is defined as a multi-item logical.
APACHE$CREATE_SYMBOLS_ GLOBAL	If defined, this system logical name causes CGI environment symbols to be defined globally. They are defined locally by default.
APACHE$DEBUG_DCL_CGI	If defined, this system logical name enables APACHE$VERIFY_DCL_CGI and APACHE$SHOW_CGI_SYMBOL.
APACHE$DL_FORCE_UPPERCASE	If defined to be true (1, T, or Y), this system logical name forces case-sensitive dynamic image activation symbol lookups. By default, symbol lookups are first done in a case-sensitive manner and then, if failed, a second attempt is made using case-insensitive symbol lookups. This fallback behavior can be disabled with APACHE$DL_NO_UPPERCASE_FALLBACK.
APACHE$DL_NO_UPPERCASE_ FALLBACK	If defined to be true (1, T, or Y), this system logical name disables case-insensitive symbol name lookups whenever case-sensitive lookups fail. See APACHE$DL_FORCE_UPPERCASE.
APACHE$PREFIX_DCL_CGI_ SYMBOLS_WWW	If defined, this system logical name prefixes all CGI environment variable symbols with "WWW_". By default, no prefix is used.
APACHE$SHOW_CGI_SYMBOL	If defined, this system logical name provides information for troubleshooting the CGI environment by dumping all of the symbols and logicals (job/process) for a given CGI. Use with APACHE$DEBUG_DCL_CGI.
APACHE$USER_HOME_PATH_ UPPERCASE	If defined to be true (1, T, or Y), this system logical name uppercases device and directory components for user home directories when matching pathnames in <DIRECTORY> containers. This provides backward compatibility for sites that specify these components in uppercase within <DIRECTORY> containers. See the UserDir directive in Modules and Directives section for more information.
APACHE$VERIFY_DCL_CGI	If defined, this system logical name provides information for troubleshooting DCL command procedure CGIs by forcing a SET VERIFY before executing any DCL CGI. Use with APACHE$DEBUG_DCL_CGI.

3.12 Redefining Logical Names

You cannot manually redefine these logical names defined during configuration:

- APACHE$ROOT
- APACHE$SPECIFIC
- APACHE$COMMON

If you need to change the definition of these logical names, rerun the configuration with the following command:

```
    $ @SYS$MANAGER:APACHE$CONFIG
```

3.13 OpenVMS Cluster Considerations

An OpenVMS Cluster is a group of OpenVMS systems that work together as one virtual system. The Secure Web Server runs in an OpenVMS Cluster so you can take advantage of the resource sharing that increases the availability of services and data. Keep the following points in mind:

- The Secure Web Server is supported on OpenVMS Alpha Version 7.3-1 or higher, or OpenVMS I64 Version 8.2 or higher.
- The Secure Web Server runs in an Alpha or an I64 cluster.

Note

The Secure Web Server V1.3-1 does not support mixed-architecture clusters. Support for mixed-architecture cluster operation will be included in a future release.

The configuration procedure lets you specify where you want to store the server software, the server system files (configuration, startup, and shutdown files), and your HTML files (content). By default, everything will go in SYS$COMMON or the device and directory you specified with the PRODUCT INSTALL command.

Where you put each server component depends on your OpenVMS cluster environment and how much you want to integrate or segregate the Secure Web Server and its activities. You can install the server once on a disk that is visible to multiple systems in the cluster. You have the option of:

- Using one configuration for all systems, or
- Configuring individual systems, provided they all share a common system disk.

In the latter case, a common system disk is needed so that all the systems have access to the server system files on the system disk. If a system has access to a clusterwide directory where the Secure Web Server is installed, but does not share a system disk with the other systems, you might need an additional installation.

If you have more than one installation, you must make sure that the APACHE$WWW account has the same UIC across the entire cluster. The installation procedure automatically assigns the APACHE$WWW UIC to all the files under [APACHE], regardless of their actual physical locations. One UIC definition for APACHE$WWW for all installations ensures that all files are visible at all times.

3.13.1 Individual System vs. Clusterwide Definition

To define clusterwide vs. individual configuration files, APACHE$ROOT uses the following concealed logical names:

- APACHE$COMMON defines clusterwide files.
- APACHE$SPECIFIC defines system-specific files.

When reading a file, the server first looks for a system-specific version of the file in APACHE$SPECIFIC:[directory]. If it does not find one, it looks for a clusterwide file in APACHE$COMMON:[directory].

To avoid confusion, always use the appropriate concealed logical name to specify the file you want to edit. For example, to edit a clusterwide version of HTTPD.CONF, refer to:

```
    $ EDIT APACHE$COMMON:[CONF]HTTPD.CONF
```

If you referred to:

```
    $ EDIT APACHE$ROOT:[CONF]HTTPD.CONF
```

the server would open the clusterwide file but save it as a system-specific version. The latest version of HTTPD.CONF would then be visible only to the individual node it was saved on.

Within HTTPD.CONF itself, you should make this distinction whenever you refer to a path or file location. This improves performance and ensures the server will return a complete directory listing. For example, you should specify APACHE$COMMON or APACHE$SPECIFIC (instead of APACHE$ROOT) with Directory directives.

The following extract, from the HTTPD.CONF file distributed with the OpenVMS kit, refers to APACHE$COMMON because the content for the default web page is in the clusterwide directories.

```
DocumentRoot "/apache$common/htdocs"

        . . .

    <Directory "/apache$common/htdocs">
        Options Indexes FollowSymLinks Multiviews
        AllowOverride None
        Order allow,deny
        Allow from all
    </Directory>
```

If there were content for one specific node in a cluster, the APACHE$SPECIFIC logical name would be used.

3.14 Common Gateway Interface (CGI)

Common Gateway Interface (CGI) programs execute within the DCL shell on the Secure Web Server for OpenVMS. Please note the following OpenVMS specific information.

3.10.1 CGI Environment Variables

By default, an environment variable symbol takes the form designated by the name of the environment variable. You can determine how environment variables are set when the server executes a CGI program.

You can define the APACHE$PREFIX_DCL_CGI_SYMBOLS_WWW logical name to prefix all environment variable symbols with "WWW_". By default, no prefix is used.

The APACHE$CGI_MODE logical name controls how CGI environment variables are defined in the executing CGI program, as follows:

```
    APACHE$CGI_MODE  option
```

where *option* can have **one** of the following values at a time:

0	Default. Environment variables are defined as local symbols and are truncated at 970 (limitable with DEC C).
1	Environment variables are defined as local symbols unless they are greater than 970 characters. If the environment value is greater than 970 characters, it is defined as a multi-item logical.
2	Environment variables are defined as logicals. If the environment value is greater than 255 characters, it is defined as a multi-item logical.

APACHE$DCL_ENV is a foreign symbol that lets you define CGI environment variables, as follows:

```
    APACHE$DCL_ENV [-c] [-d] [-e env-file] [-l]
```

where:

-c	Default. Indicates create environment variables.
-d	Indicates delete environment variables.
-e env-file	Specifies an alternate environment file. The environment file does not need to be specified by the caller because the parent derives it (it is easily be determined by default).
-l	Allows you to list the contents of the environment file. This option is used to display the enviroment variables when you specify the APACHE$DEBUG_DCL_CGI and the APACHE$SHOW_CGI_SYMBOLS logical names.

An example of the output of an environment file is as follows:

```
(CGI Environment Variables)

 "DOCUMENT_ROOT" = "/apache$common/htdocs"
 "HTTP_ACCEPT" = "image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*"
 "HTTP_ACCEPT_LANGUAGE" = "en-us"
 "HTTP_CONNECTION" = "Keep-Alive"
 "HTTP_HOST" = "husky2.zko.dec.com"
```

The following example deletes the environment and then recreates it:

```
  Example:  diff_mode_cgi.com
  $ APACHE$DCL_ENV -d
  $ Define APACHE$PREFIX_DCL_CGI_SYMBOLS_WWW 1
  $ APACHE$DCL_ENV -c
```

3.14.2 Referencing Input

CGI scripts that reference input to the Secure Web Server must refer to APACHE$INPUT.

3.10.3 Executing CGI

On OpenVMS, CGI images execute within a DCL process. You cannot execute CGI images directly.

3.10.4 Logicals for Debugging CGI Scripts

Use the following logical to debug CGI scripts.

Logical Name	Description
APACHE$DEBUG_DCL_CGI	If defined, this system logical name enables APACHE$VERIFY_DCL_CGI and APACHE$SHOW_CGI_SYMBOL.
APACHE$VERIFY_DCL_CGI	If defined, this system logical name provides information for troubleshooting DCL command procedure CGIs by forcing a SET VERIFY before executing any DCL CGI. Enabled by APACHE$DEBUG_DCL_CGI.
APACHE$SHOW_CGI_SYMBOL	If defined, this system logical name provides information for troubleshooting the CGI environment by dumping all of the symbols and logicals (job/process) for a given CGI. Enabled by APACHE$DEBUG_DCL_CGI.

3.10.5 Displaying Graphics with CGI Command Procedures

To display a graphics file with a CGI command procedure, use the APACHE$DCL_BIN foreign symbol in the following format:

```
APACHE$DCL_BIN [-s bin-size] bin-file
```

where:

-s bin-size	Specifies the actual or approximate file size in bytes. Bin-size is automatically determined if the image file is larger than 32768K (default value). If the image file is smaller than 32768K, you can provide an approximate (or actual) size (this will boost performance).
bin-file	Specifies the file to be displayed.

For example:

```
$ APACHE$FLIP_CCL
$ WRITE SYS$OUTPUT F$FAO("!as!/!/","CONTENT-TYPE: IMAGE/GIF")
$ APACHE$DCL_BIN APACHE$ROOT:[ICONS]APACHE_PB.GIF
$ EXIT
```

Chapter 4
Security Information

The Secure Web Server for OpenVMS is a non-privileged, user-mode, socket-based network application. TMPMBX and NETMBX are the only privilege requirements. The server runs under its own unique UIC and user account (APACHE$WWW).

4.1 Process Model

The Secure Web Server runs as a single job which consists of:

- A master process (APACHE$ssss)
  and
- Several detached processes

Server processes can have two types of process name: APACHE$nn, where *nn* is an integer number, or APACHE$ssss, where *ssss* is a user-defined server tag. Similarly, child processes can have two types of process name: APACHE$nnmmm, where *APACHE$nn* is the server and *mmm* is an integer number, or APACHE$ssssmmm, where *ssss* is a user-defined server tag and *mmm* is an integer number. The CSWS_JAVA Java servlet engine creates a process (APACHE$TOMCAT) to execute Java programs. CSWS_PERL does not create any processes.

The OpenVMS security profile for each process is identical and no enhanced mechanism is required for these processes to communicate with one another. Resource utilization is controlled by a single user account (APACHE$WWW) where pooled quotas are defined.

4.2 Privileged Images

The Secure Web Server performs three operations that require additional privilege:

- Binding to a port below 1024 (privileged ports)
  By default, the server binds to port 80 (HTTP) and 443 (HTTPS) which are privileged ports.
- Fetching path information for other users
  The server provides a replacement for the getpwnam C RTL routine to allow the server to fetch default path information for other users (required by MOD_UTIL and MOD_USERDIR).
- The server validates OpenVMS passwords required by MOD_AUTH_OPENVMS.
- Changing the "carriage-control" attribute on socket (BG) devices
  The server also enables/disables the carriage-control attribute on BG (socket) devices for certain stream operations.

Two protected shareable images are installed at startup to allow the server to perform these functions:

- APACHE$PRIVILEGED.EXE (exec-mode services)
- APACHE$FIXBG.EXE (kernel-mode services)

The APACHE$PRIVILEGED.EXE image provides exec-mode services for binding to privileged sockets and fetching a user's default path information. Access to these services is limited to processes running under the APACHE$WWW username and is controlled by the APACHE$PLV_ENABLE_APACHE$WWW logical name. This logical name is defined as:

```
APACHE$PLV_ENABLE_APACHE$WWW" = "7,80,1023"
```

The "7,80,1023" string represents three parameters where:

- The first parameter (7) is a bit-mask which enables/disables the two services:
    - Bit 0 controls binding to privileged ports.
    - Bit 1 controls fetching user default path information and validating OpenVMS passwords.
    - Bit 2 controls the creation of Galaxy Global Sections for the shared memory SSL session cache.
- The second and third parameters are the minimum and maximum port allowed to be bound.

When a call to either service is made, the service code:

1. Temporarily enables the privileges SYSPRV, OPER, SYSNAM, and NETMBX.
2. Performs the function.
3. Restores the process' original privileges.

The APACHE$FIXBG.EXE image provides a kernel-mode service for manipulating the carriage-control attribute for BG devices owned by the calling process. There is no special access control on this service. This function can also be performed using a setsocketopt C RTL run-time call, but it is not supported by all TCP/IP stack vendors, which is the reason this service exists. This service does not enable privileges, but executes in kernel mode.

4.3 Privileges Required to Start and Stop the Server

The Secure Web Server runs under the APACHE$WWW username and UIC and is started as a detached, network process. During startup, protected images are installed and logical names are placed in the system logical name table. Shutdown is accomplished by sending a KILL signal to the master process and its subprocess.

These actions require enhanced privileges (DETACH, SYSNAM, WORLD, etc.) and are usually performed from a suitably privileged account.

4.4 File Ownership and Protection

All of the server's files reside under its root directories pointed to by the APACHE$ROOT logical name. During installation, file protection is set to (S:RWED, O:RWED, G, W). During configuration, all files are set to be owned by APACHE$WWW.

4.5 Authentication Using OpenVMS Usernames and Passwords (MOD_AUTH_OPENVMS)

The MOD_AUTH_OPENVMS module supports authentication using the usernames and passwords contained in the system authorization file (SYS$SYSTEM:SYSUAF.DAT). You can optionally load the new module at startup. The module is located in:

```
APACHE$COMMON:[MODULES]MOD_AUTH_OPENVMS.EXE
```

Note

The HTTP password authentication protocol transmits username and password information in the clear. When you enable mod_auth_openvms, you are potentially exposing your user's passwords to password sniffing attacks. If your application is intended to be used over external network connections, consider placing your password-protected pages within an SSL-protected directory so usernames and passwords are encrypted before they are transmitted.

To enable mod_auth_openvms, edit the HTTPD.CONF file to include the following directive:

```
LoadModule auth_openvms_module
/apache$common/modules/mod_auth_openvms.exe
```

This module supports the following directives:

- `AuthOpenVMSUser {On,Off}`
  This directive controls whether user authentication and authorization using MOD_AUTH_OPENVMS are enabled or disabled (On or Off, respectively). By default, AuthOpenVMSUser is set to On if the MOD_AUTH_OPENVMS module is loaded.

  The syntax is similar to the AuthUserFile directive provided by MOD_AUTH and the AuthDBUserFile directive provided by MOD_AUTH_DB.

  Note

  Versions of the Secure Web Server prior to V1.3 defined the AuthUserOpenVMS directive, which is equivalent to AuthOpenVMSUser. The AuthUserOpenVMS directive is still supported; however, HP recommends that you use the new syntax. By default, the module is enabled once it is loaded.

- `AuthOpenVMSGroup {On,Off}`

  This directive controls whether group authorization using MOD_AUTH_OPENVMS is enabled or disabled (On or Off, respectively). By default, AuthOpenVMSGroup is set to On if the MOD_AUTH_OPENVMS module is loaded. (If AuthOpenVMSUser is disabled, then AuthOpenVMSGroup is also disabled, even if AuthOpenVMSGroup On is specified.)
  The syntax is similar to the AuthGroupFile directive provided by MOD_AUTH and the AuthDBGroupFile directive provided by MOD_AUTH_DB.

For example, if a directory is protected using an .HTACCESS file, the contents of that file might contain:

```
AuthType Basic
AuthName "OpenVMS authentication"
AuthOpenVMSUser On
require valid-user
```

When a user seeks to open a file in that directory, the user will be prompted for a username and password. That username and password must match entries in the SYSUAF.DAT file. Furthermore, the SYSUAF.DAT entry must allow a network login for that username at the time of the request. If an invalid authentication request occurs, an intrusion record is written.

An authentication request can be rejected for the following reasons:

- Username is blank or invalid.
- Password is blank or invalid. (Note: The Secure Web Server does not allow blank or null passwords, even though blank or null passwords are allowed by OpenVMS.)
- Password is valid, but target account is flagged as an intruder. (Account is flagged after 5 consecutive failures in a given time period; see the DCL commands SHOW INTRUSION and DELETE INTRUSION.)
- Target account has a secondary password defined. (HTTP password protocol does not support secondary passwords.)

- Account is marked for external authentication (see AUTHORIZE /FLAGS=EXTAUTH qualifier).
- Password has expired (see AUTHORIZE /FLAGS=PWD_EXPIRED qualifier).
- Account has expired (see AUTHORIZE /EXPIRATION qualifier).
- Account is disabled (see AUTHORIZE /FLAGS=DISUSER qualifier).
- Access restrictions prevent a network access for this time of day (see AUTHORIZE /ACCESS and /PRIMEDAYS qualifiers).

4.5.1 The require group Directive

When you use MOD_AUTH_OPENVMS for authentication, the software accepts one or more require group directives. Each require group directive can contain one or more group names or identifiers. (Note that the software treats the values specified in the require group directives in a case-insensitive manner.)

After a user has been authenticated using the specified password, MOD_AUTH_OPENVMS compares the UIC of the user to determine if the account is a member of a group that was included in the list of require group directives. If so, the user is allowed access. If not, the module looks to see if the SYSUAF account has been granted any of the specified identifiers.

4.5.2 The require user Directive

For simplicity and for consistency with the require group directive, MOD_AUTH_OPENVMS uses a case-insensitive comparison when processing values specified by the require user directive.

4.5.3 Hiding Accounts

System administrators can specify the accounts that MOD_AUTH_OPENVMS will use for authentication and authorization. In particular, system administrators may want to prevent users from attempting to authenticate using certain accounts.

By default, any account in the SYSUAF file can be used by the MOD_AUTH_OPENVMS module. However, if the system administrator uses the AUTHORIZE utility to create an identifier called APACHE$MOD_AUTH_OPENVMS_ENABLE, then MOD_AUTH_OPENVMS only utilizes accounts that have APACHE$MOD_AUTH_OPENVMS_ENABLE granted to them. Any account that does not have that identifier granted to it is effectively hidden.

If the APACHE$MOD_AUTH_OPENVMS_ENABLE identifier has not been defined, then the system administrator can define an identifier called APACHE$MOD_AUTH_OPENVMS_DISABLE. If that identifier exists, then any account in the SYSUAF file can be used by MOD_AUTH_OPENVMS provided that account does not hold the APACHE$MOD_AUTH_OPENVMS_DISABLE identifier. Any account that has APACHE$MOD_AUTH_OPENVMS_DISABLE granted to it is hidden. This identifier might be used in cases where the system administrator wants to use MOD_AUTH_OPENVMS for all accounts except certain specific ones.

If both identifiers are defined, only APACHE$MOD_AUTH_OPENVMS_ENABLE is significant: APACHE$MOD_AUTH_OPENVMS_DISABLE is ignored. For example, if the system administrator has been using the APACHE$MOD_AUTH_OPENVMS_ENABLE identifier to specify individual accounts that can be used by MOD_AUTH_OPENVMS, then the administrator will have to remove that identifier from all accounts and remove the definition of that identifier before the administrator can define the APACHE$MOD_AUTH_OPENVMS_DISABLE identifier. Otherwise the software will see the definition of APACHE$MOD_AUTH_OPENVMS_ENABLE and will require that that identifier be granted for any account that can be used by MOD_AUTH_OPENVMS.

An account that is hidden is treated by MOD_AUTH_OPENVMS as if the account does not exist. If AuthOpenVMSAuthoritative is on, a security request for a hidden account is rejected. If

AuthOpenVMSAuthoritative is off, the request is declined, and some other security facility can be used to process the request.

4.5.4 MOD_AUTH_OPENVMS Security Considerations

Installing a privileged shareable image library such as APACHE$PRIVILEGED adds some risk to system security. In particular, with APACHE$PRIVILEGED installed, certain users will be able to tell which accounts are valid and will be able to check to see if certain accounts hold certain identifiers. However, only users that are identified by the system wide logical name APACHE$PLV_ENABLE_<user> are allowed to run these routines. Frequently, this means that only the Secure Web Server server process account (APACHE$WWW) is allowed to call these privileged functions.

Furthermore, the system administrator can use the APACHE$MOD_AUTH_OPENVMS_ENABLE or APACHE$MOD_AUTH_OPENVMS_DISABLE identifier to hide some accounts, particularly privileged accounts, from the Secure Web Server software.

4.5.5 MOD_AUTH_OPENVMS Examples

The following directives demonstrate MOD_AUTH_OPENVMS authentication and authorization.

In the following portion of an HTTPD.CONF file, MOD_AUTH_OPENVMS is used for authentication and authorization:

```
<Location /private/projects>
  AuthType      Basic
  require       user    j_smith m_jones
  require       group   db_read_ident db_write_ident
  require       group   db_maint db_systems db_admin
  require       group   testing customer_support
  require       group   accounting travel
</Location>
```

If user "m_jones" has been authenticated, then that user is allowed access. If the authenticated user has an account that has a DB_READ_IDENT or DB_WRITE_IDENT identifier granted to it, then access is allowed. The user may also have a UIC that is assigned one of the specified groups. For example, if the ASCII representation of the UIC is [DB_ADMIN,ME], then access is allowed.

Some of the specified user names and groups names may not be defined in the SYSUAF file. Instead, the user may be in the PASSWD.DAT file. If so, authentication will be performed by MOD_AUTH using the groups defined in the GROUPS.DAT file.

The following is an unusual example showing MOD_AUTH_OPENVMS used for authentication but not authorization. In other words, once a user has been authenticated, MOD_AUTH is used to handle the require user and require group directives. In particular, the group names used by the require group directives must be contained in GROUPS.DAT:

```
<Location /private/projects>
  AuthType      Basic
  AuthUserFile  /private/passwd.dat
  AuthGroupFile /private/groups.dat
  AuthOpenVMSGroup Off
  require       user    j_smith m_jones
  require       group   db_maint db_systems db_admin
  require       group   testing customer_support
  require       group   accounting travel
</Location>
```

4.6 Server Extensions (CGI Scripts, PHP Scripts, Perl Modules)

Server extensions, such as CGI scripts, PHP scripts, and Perl modules, run within the context of the Secure Web Server's process or its subprocesses. These extensions have complete control over the server environment. You can configure the server to allow execution of arbitrary user scripts, but standard practice is to limit such activity to scripts written by completely trusted users. The Secure Web Server includes directives that allow a web administrator to control script execution and client access. The use of these directives is described in numerous books and is not duplicated here.

4.7 suEXEC in the Secure Web Server

The following sections discuss the implementation of suEXEC in the Secure Web Server.

4.7.1 suEXEC Security Model

suEXEC in the Secure Web Server uses rights identifiers to indicate authorized users to run suEXEC as well as users to be run via suEXEC.

The Secure Web Server does not use UID/GID minimums to determine the validity of the calling user. The use of the SETUID/SETGID restrictions on the invoked CGI or SSI program is currently not implemented.

suEXEC in the Secure Web Server supports the use of the User and UserDir directives within virtual hosts, and also supports the EXEC CGI mod_include directive.

There are no restrictions on OpenVMS account privileges or MAXSYSGROUP for suEXEC programs.

4.7.2 Configuring suEXEC

You can configure suEXEC using the configuration utility provided with the installation (SYS$MANAGER:APACHE$CONFIG.COM). This utility allows you to enable or disable the suEXEC feature for a given server.

Note

Before you enable suEXEC, be sure that the user accounts that are to be run via suEXEC have been created.

To enable suEXEC, run SYS$MANAGER:APACHE$CONFIG.COM and answer Yes to the question about enabling the suEXEC feature.

The suEXEC image is installed with privileges.

When you enable suEXEC, the following occur:

- The APACHE$SUEXEC_SRVR and APACHE$SUEXEC_USER rights identifiers are created in the rights database, if they do not already exist.
- The APACHE$SUEXEC_SRVR rights identifier is granted to the server account, and the user is prompted to enter user accounts that are to be run via suEXEC. These user accounts are granted the APACHE$SUEXEC_USER rights identifier.

- An suEXEC directory is created within the htdoc root (APACHE$COMMON:[HTDOCS.SUEXEC]) and set with the appropriate default ACEs that allow the Apache server read access to the suEXEC CGI/SSI programs.

After you have enabled suEXEC, manually perform the following steps:

- For each user account to be run via suEXEC, create a directory owned by that user under the suEXEC directory. For example, if you create a directory named *user*, it will be located in APACHE$COMMON:[HTDOCS.SUEXEC.*user*].
- Within each virtual host configuration, use the Alias or ScriptAlias directive to define a location for the suEXEC CGI/SSI programs to be used.

To disable suEXEC, run SYS$MANAGER:APACHE$CONFIG.COM and answer No to the question about enabling the suEXEC feature.

When you disable suEXEC, the following occur:

- The suEXEC ACEs are removed from all files within the Apache root.
- The APACHE$SUEXEC_SRVR rights identifier is revoked from the server account (APACHE$WWW) and the user is prompted about whether to disable all suEXEC servers.
- The user is prompted about whether to disable all suEXEC users.
- If no server accounts remain enabled, the APACHE$SUEXEC_SRVR rights identifier is removed from the rights database.
- If no user accounts remain enabled, the APACHE$SUEXEC_USER rights identifier is removed from the rights database.

4.8 Protecting Server Certificate Keys

The Secure Web Server's certificate keys must be protected against disclosure. The keys, by default, are owned by APACHE$WWW, and protected as (S:RWED, O:RWED, G, W). As an additional measure of security, the keys can be encrypted. When using encrypted keys, a password must be entered during startup to decrypt the keys.

Keys must be kept out of directories accessible by clients (such as document directories!).

A second threat for key disclosure exists during script execution because scripts run in the context of the server and have complete access to key files no matter where they exist (as long as they exist in a directory accessible to APACHE$WWW). Therefore, it is not advisable to allow the execution of arbitrary user scripts when using SSL.

Chapter 5
Building and Debugging Loadable Apache Modules for the Secure Web Server

The Secure Web Server for OpenVMS is ported from the Apache Web Server and includes all of the standard Apache modules as well as several optional modules. The Apache Web Server design architecture allows new modules to be added to the server at the following times:

- When the server is built
- Dynamically at run-time using the Apache Dynamic Shared Object (DSO) feature

On OpenVMS, the DSO function is performed by the LIB$FIND_IMAGE_SYMBOL run-time library routine. When the server encounters a LoadModule directive, it calls LIB$FIND_IMAGE_SYMBOL to load a shareable image.

For example:

```
    LoadModule rewrite_module /apache$common/modules/mod_rewrite.exe
```

This directive directs the server to activate the shareable image mod_rewrite.exe using the universal symbol "rewrite_module" to locate the Apache module data structure describing the module's internal routine entry points.

Modules that have been contributed by various authors, but are not part of the Apache source code distribution are listed in the Apache module registry at http://httpd.apache.org/

5.1 The Apache API, Run-Time Library, and HTTP Request Processing

A module must conform to the Apache API, which dictates how the server and module communicate with one another and how HTTP requests are represented and processed. The Apache module data structure is an important component of this communication because it provides information to the server about the module and its capabilities.

The server also supplies a run-time library that provides various services frequently needed by module writers (such as memory pool allocation).

An HTTP request is processed in several phases. Each phase performs a specific task, such as URI-to-filename translation, auth-id checking, send-response, logging, and so on. A module indicates its desire to participate in one or more of these phases by loading a routine address into the appropriate entry of the module data structure.

A good reference is *Writing Apache Modules with Perl and C: The Apache API and mod_perl* by Lincoln Stein and Doug MacEachern (O'Reilly).

5.2 Building a Module

The natural language for building Apache modules is C. This is the only language for which Apache provides data structure definitions and function prototypes. You can write an Apache module in any language you choose, but you will need to provide your own language-specific header files or C-wrappers. In the rest of this section, we use C as the implementation language.

5.2.1 Defining Your Apache Module Data Structure Symbol

Your module must include the Apache module data structure pointed to by a universal symbol of your choice. During startup, the server dynamically activates your module by calling LIB$FIND_IMAGE_SYMBOL with the name of your module's Apache Module Data Structure (obtained from the LoadModule directive in your httpd.conf file).

By default, the server calls LIB$FIND_IMAGE_SYMBOL using a case-sensitive symbol lookup. If the lookup fails, the server calls LIB$FIND_IMAGE_SYMBOL using a case-insensitive lookup. (This behavior can be changed. See Logical Names for the description of APACHE$DL_NO_UPPERCASE_FALLBACK and APACHE$DL_FORCE_UPPERCASE logical name controls.)

5.2.2 Compiling a Module

The following compiler switches are used to build the server:

```
/DEFINE=(EAPI)
/POINTER_SIZE=32
```

You should use the same compiler switches when building your module.

EAPI

Specify the EAPI compile-time macro when you compile your module, because the Secure Web Server is built from Apache source code with mod_ssl server core patches applied. When EAPI is used during compilation, the mod_ssl server core patches are activated and certain changes are made to structures and function prototypes.

One change of interest to module writers is the expansion of the module structure (this structure contains information about the phases of interest to the module) and module magic number (this number describes the version of the module structure). If you attempt to load a module that was not compiled with EAPI, the server logs an entry in the error log file indicating this, but will proceed to load the module. Depending on which features of the Apache API the module uses, this may have no effect or produce unpredictable results.

Note

Specify the EAPI macro if you are using the pre-built Secure Web Server supplied by HP. If you have built it yourself, you need to specify EAPI if that is how you built the server.

5.2.3 Linking a Module

A loadable module is implemented as an OpenVMS shareable image. Link your module with the /SHARE qualifier.

Most modules reference at least one routine from the Apache run-time library. To resolve these references at link-time, use a linker options file to specify the APACHE$HTTPD_SHR shareable image that contains the Apache run-time library routines.

Note

Your shareable image must not contain any linker warnings or errors in order to be properly loaded at run-time by LIB$FIND_IMAGE_SYMBOL.

For guidelines on how to design and write a shareable image, see the *OpenVMS Linker Utility Manual* and *Guide to Creating OpenVMS Modular Procedures*.

5.2.4 Example: mod_rewrite

mod_rewrite provides powerful URL-rewriting capabilities. (See http://httpd.apache.org/docs/mod/mod_rewrite.html for a description of mod_rewrite capabilities). You can include mod_rewrite with Secure Web Server by building it as a DSO module from sources.

This example module was built using hp C V6.4-008 on OpenVMS Alpha V7.2-2 and the Secure Web Server Version 1.1 for OpenVMS.

```
$! How-to-Build: mod_rewrite
$!
$! Copy the following files from the CSWS source kit to your default directory:
$!
$! mod_rewrite.c
$! mod_rewrite.h
$!
$! Copy rewriteguide.html from the CSWS source kit to
$! apache$common:[htdocs.manual.misc].
$!
$! Once mod_rewrite.exe is built, copy it to apache$common:[modules] and
$! add the following line to your httpd.conf file:
$!
$! LoadModule rewrite_module /apache$common/modules/mod_rewrite.exe
$!
$! Restart the server.
$!
$ cc/lis/define=eapi/prefix=all -
/include=(apache$common:[src.include],apache$common:[src.os.openvms]) -
mod_rewrite
$!
$ link/share/map mod_rewrite,sys$input:/option
!
! Linker options
!
GSMATCH=LEQUAL,1,0

SYMBOL_VECTOR=(REWRITE_MODULE=DATA)

! Run SYS$STARTUP:APACHE$CONFIG to setup the following logical name.
!
APACHE$HTTPD_SHR/SHARE
$!
$ exit
```

5.2.5 Debugging a User-Built Apache Module

This section describes two methods for debugging a user-built Apache DSO module using the standard server provided with the Secure Web Server. The instructions assume the Secure Web Server Version 1.1 or later, but the same techniques can be used with earlier versions with slight modifications.

There are two methods of running the server for debugging purposes:

- Running the server image in a detached process (normal)
- Running the server image directly

The first method requires a separate terminal device to send and receive debugger input/output. A DECterm device is used in this example. If DECwindows is not available, any terminal device that allows read/write access to APACHE$WWW can be used (such as a logged-in TELNET terminal), but APACHE$WWW will need SHARE privilege to access the device if it is already assigned to another process.

The second method does not require a separate terminal device. This technique can only be used to debug the main process, so you must run the server with the "-X" option. If this context is not appropriate for the problem you are debugging, the other method of debugging can be used (a separate terminal window will be required for each child process).

5.2.5.1 Preparing to debug your module

Before you begin debugging your module, perform the following steps:

1. Insert the following code sequence as the first lines of your module's initialization routine to activate the OpenVMS debugger at run-time:

```
    #ifdef __VMS
    #include <ssdef.h>
    #endif

    extern void lib$signal (int);
    lib$signal (SS$_DEBUG);
```

2. Build a debug version of your module. Compile with /DEBUG=(TRACEBACK,SYMBOLS) and link with /DEBUG, then copy the image to an appropriate directory. For example:

```
    apache$common:[modules]mod_rewrite.exe_debug
```

You may also want to copy the source code files here so that the debugger can find them (DBG> set source/latest apache$common:[modules]).

3. In httpd.conf, load the debug version of the module. For example:

```
    LoadModule rewrite_module
/apache$common/modules/mod_rewrite.exe_debug
```

4. Allow interactive logins to APACHE$WWW:

```
    UAF> mod apache$www/inter/pass=<password>/nopwdexpir
```

5.2.5.2 Debugging your module

Choose one of the following two methods to debug your module, depending on how you run the Apache server. These two methods are as follows:

- Method A: Normal detached server process (Apache server runs in its normal, detached context)
- Method B: Direct execution of server image (runs the Apache server image in your current process)

Method A - Normal Detached Server Process

This method allows the Apache server to run in its normal, detached context. The only difference from the standard configuration is that the APACHE$WWW process possesses GRPNAM privilege (in order to define DBG$INPUT and DBG$OUTPUT logical names in the LNM$GROUP table). In this example, the "-X" option is used so that no child processes are created.

To debug a normal detached server process, perform the following steps:

1. Grant GRPNAM privilege and allow interactive logins to APACHE$WWW:

```
UAF> mod apache$www/priv=grpnam/inter
```

2. From the APACHE$WWW account, run:

```
$ set proc/priv=grpnam
$ set display/create/node=<domain-name-of-debug-window-client>/trans=tcpip
$ create/term/noprocess/define=(table=lnm$group,dbg$input,dbg$output)
```

3. From SYSTEM (or suitably privileged) account, run:

```
$ @sys$startup:apache$config ! (specify "-X" for server options)
$ @sys$startup:apache$startup
```

Note

If the problem you are debugging requires child processes, you will need a separate terminal device for each process with process-specific dbg$input and dbg$output logical names defined for each child process. One way to do this is to create one DECterm window for each child process (in this example, MaxSpareServers is set to 1):

```
$ create/term/noprocess/define=(table=lnm$group,apache$00_dbg)
$ create/term/noprocess/define=(table=lnm$group,apache$00000_dbg)
$ create/term/noprocess/define=(table=lnm$group,apache$00001_dbg)
```

In APACHE$ROOT:[000000]LOGIN.COM, define dbg$input and dbg$output:

```
$ define/job dbg$input 'f$process()'_dbg
$ define/job dbg$output 'f$process()'_dbg
```

Method B - Direct Execution of Server Image

This method runs the Apache server image in your current process. One advantage of this method is that GRPNAM is not required nor is a separate debugger window needed.

To debug for direct execution of a server image, perform the following steps:

1. From the SYSTEM (or suitably privileged) account:

   ```
   $ @sys$startup:apache$config ! (specify system-wide logicals)
   ```

2. From the APACHE$WWW account, run:

   ```
   $ mcr apache$root:[000000]apache_httpd.exe "-X"
   ```

Debugger Session

Use the debugger as usual:

```
DBG> set image mod_rewrite
DBG> set module/all
DBG> set source/latest apache$root:[modules]mod_rewrite.c
```

Chapter 6
Open Source Licenses

This chapter provides open source license acknowledgements and license references.

Apache

This product includes software developed by the Apache Software Foundation.

Apache Software License at http://h71000.www7.hp.com/openvms/products/ips/apache/apache_license.txt

CSWS_JAVA

This product includes software developed by the Jakarta (Java Apache) Project and is covered under the Apache License for use in the Apache Tomcat project.

Apache Public License at http://h71000.www7.hp.com/openvms/products/ips/apache/apache_license.txt

Mod_SSL

This product includes software developed by Ralf S. Engelschall <rse@engelschall.com> for use in the Mod_SSL Project.

Mod SSL License at http://h71000.www7.hp.com/openvms/products/ips/apache/modssl_license.txt

OpenSSL

This product includes software developed by the OpenSSL Project. This product includes cryptographic software written by Eric Young.

OpenSSL License at http://h71000.www7.hp.com/openvms/products/ips/apache/apache_license.txt

MOD_PERL

This product includes software developed by the Apache/Perl Integration Project.

Mod_Perl License at http://h71000.www7.hp.com/openvms/products/ips/apache/modperl_license.txt

Perl

This product includes software developed by the Perl Project.

Perl License at http://h71000.www7.hp.com/openvms/products/ips/apache/perl_license.txt

MOD_PHP