HP Secure Web Server for OpenVMS (based on Apache) Version 2.1-1 Installation and Configuration Guide

June 2007

Version 2.1-1 for OpenVMS Alpha, based on Apache 2.0.52
CPQ-AXPVMS-CSWS-V0201-1-1.PCSI_SFX_AXPEXE

Version 2.1-1 for OpenVMS I64, based on Apache 2.0.52
HP-I64VMS-CSWS-V0201-1-1.PCSI_SFX_I64EXE

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

Software Version
Secure Web Server for OpenVMS Version 2.1-1

Hewlett-Packard Company
Palo Alto, California

Contents

Chapter 1 Installation Requirements and Prerequisites

1.1 Hardware Requirements
1.1.1 ODS-5 Disk
1.1.2 Disk Space
1.1.3 Stream_LF File Format No Longer Required
1.2 Software Requirements
1.2.1 MultiNet and TCPware Network Products
1.2.2 CSWS_JAVA Requirements
1.2.3 CSWS_PHP Requirements
1.2.4 CSWS_PERL Requirements
1.2.5 Building the Apache HTTP Server from Source Code

Chapter 2 Installation and Configuration

2.1 Read the Release Notes
2.2 Install the Secure Web Server
2.2.1 Sample Installation
2.3 Configure the Secure Web Server
2.3.1 Configuration Menu
2.3.2 Configuring a Single Server
2.3.3 Sample Configuration of a Single Server
2.3.4 Configuring Multiple Servers
2.3.5 Sample Configuration of Multiple Servers
2.3.6 Delete Server Instance
2.3.7 Managing suEXEC
2.3.8 Running the OpenSSL Certificate Tool
2.3.9 Converting Files to Stream_LF
2.3.10 Starting and Stopping the Secure Web Server
2.3.11 Showing the Status of an Apache Instance
2.3.12 Adding a Node to CSWS in a Cluster Environment
2.3.13 Managing Multiple Servers
2.3.13.1 HTTPD.CONF
2.3.13.2 APACHE$SETUP.COM and LOGIN.COM
2.3.14 Viewing the OpenSSL Certificate
2.4 Post Configuration Checklist
2.4.1 Configure CSWS_JAVA
2.4.2 Check the CSWS_PERL Configuration
2.4.3 Check the CSWS_PHP Configuration
2.4.4 Run AUTOGEN
2.4.5 Check Disk Quota
2.4.6 Check for SET TERMINAL/INQUIRE
2.5 Test the Installation
2.5.1 Browser Test
2.5.2 TELNET Test
2.5.3 Troubleshooting
2.6 What's Next
2.7 Merge Changes to Files You Have Customized
2.8 Installing Optional Modules at a Later Time

Chapter 3 Running the Secure Web Server on OpenVMS

3.1 Starting and Stopping the Server
3.1.1 Starting the Server
3.1.2 Stopping the Server
3.1.2.1 Stopping the Server Using the Server Process Name
3.2 Server Log File

3.3 Performance Considerations
3.3.1 Limits and Quotas
3.3.2 Server Experiencing Medium to High Usage
3.3.3 Global Pages and Global Sections
3.3.4 Excessive File Build Up
3.4 Customizing the Server Environment
3.5 Modules and Directives
3.5.1 Apache Modules
3.5.2 Apache 1.3 Modules Not Included
3.5.3 OpenVMS Directives
3.5.4 Command Line Options
3.5.5 Virtual Host Support
3.5.6 Dynamic Shared Object Support
3.5.7 File Handlers
3.5.8 Content Negotiation
3.5.9 Apache API
3.5.10 WebDAV (Distributed Authoring and Versioning) Support
3.5.10.1 Testing DAV Operation
3.5.11 suEXEC Support
3.5.12 MOD_SSL
3.5.12.1 Setting up a Galaxy Shared-Memory SSL Session Cache
3.5.13 Running MOD_OSUSCRIPT
3.6 File Formats
3.7 Managing File and Directory Access Controls
3.7.1 Outbound Access to Non-CSWS Files and Directories
3.7.2 Inbound Access to SWS Files and Directories
3.8 Logical Names
3.9 OpenVMS Cluster Considerations
3.9.1 Individual System vs. Clusterwide Definition
3.9.2 Mixed-Architecture (Alpha and VAX) Cluster
3.10 Common Gateway Interface (CGI)
3.10.1 CGI Environment Variables
3.10.2 Referencing Input
3.10.3 Executing CGI
3.10.4 Logicals for Debugging CGI Scripts
3.10.5 Displaying Graphics with CGI Command Procedures

Chapter 4 Security Information

4.1 Process Model
4.2 Privileges Required to Start and Stop the Server
4.3 File Ownership and Protection
4.4 Authentication Using OpenVMS Usernames and Passwords (MOD_AUTH_OPENVMS)
4.4.1 The require group Directive
4.4.2 The require user Directive
4.4.3 Hiding Accounts
4.4.4 MOD_AUTH_OPENVMS Security Considerations
4.4.5 MOD_AUTH_OPENVMS Examples
4.5 Server Extensions (CGI Scripts, PHP Scripts, Perl Modules)
4.6 suEXEC in the Secure Web Server
4.6.1 suEXEC Security Model
4.6.2 Configuring suEXEC
4.6.2.1 Using Paths with Logicals in UserDir Directive
4.6.2.2 Using Paths with Device names in UserDir Directive
4.7 Protecting Server Certificate Keys

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

5.1 The Apache API, Run-Time Library, and HTTP Request Processing
5.2 Building a Module

5.2.1 Defining Your Apache Module Data Structure Symbol
5.2.2 Compiling a Module
5.2.3 Linking a Module
5.2.4 Example: mod_rewrite
5.2.5 Debugging a User-Built Apache Module
5.2.5.1 Preparing to debug your module
5.2.5.2 Debugging your module

Chapter 6 Open Source Licenses

Chapter 1
Installation Requirements and Prerequisites

Before you can install the Secure Web Server for OpenVMS *(based on Apache)*, 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 AlphaServer system running OpenVMS Version 7.3-2 or higher, or any Integrity server system running OpenVMS I64 Version 8.2 or higher.

1.1.1 ODS-5 Disk

HP requires that you install the Version 2.1-1 kit on an ODS-5 enabled disk.

Important

You must install the V2.1-1 kit on an ODS-5 target volume. If you attempt to install this kit on an ODS-2 volume, the installation will fail. If you had an existing CSWS V1.3 installation, the failed operation will leave it in a corrupt state.

Verify that the destination device is an ODS-5 volume by entering a command similar to the following, where *DISK$DKA200* is the disk where you want to install the Secure Web Server:

```
$ SHOW DEV DISK$DKA200/FULL

Disk VARMIT$DKA200:, device type COMPAQ BB00923468, is online, mounted,
file-oriented device, shareable, available to cluster, error logging is
enabled.

Volume Status:  ODS-5, subject to mount verification, file high-water
marking, write-back caching enabled.
```

1.1.2 Disk Space

The Secure Web Server for OpenVMS Alpha compressed file contains 13,743 blocks. The expanded PCSI file requires approximately 42,000 blocks of working disk space to install.

The Secure Web Server for OpenVMS I64 compressed file contains 17,134 blocks. The expanded PCSI file requires approximately 59,000 blocks of working disk space to install.

1.1.3 Stream_LF File Format No Longer Required

The Secure Web Server no longer requires that all served files must be in Stream_LF format. See Converting Files to Stream_LF for information about a command procedure included in the kit that automatically converts your files if you choose to do so.

1.2 Software Requirements

The Secure Web Server requires the following software:

- HP OpenVMS Alpha Version 7.3-2 or higher
  – or –
  HP OpenVMS I64 Version 8.2 or higher

  Be sure that your system has the latest required ECOs, available from HP Services OpenVMS Software Patches, or use the HP Services OpenVMS FTP site.

- HP TCP/IP Services for OpenVMS Version 5.4 or higher (for SWS on OpenVMS Alpha Version 7.3-2)
  – or –
  HP TCP/IP Services for OpenVMS Version 5.5 or higher (for SWS on OpenVMS Alpha and OpenVMS I64 Version 8.2 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.

Note

MultiNet and TCPware currently support IPv4 only. If you want to take advantage of the IPv6 support in the Secure Web Server, you must use HP TCP/IP Services for OpenVMS Version 5.3 or higher.

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 Requirements

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 V3.0 provides Java Servlet 2.4 and JSP 2.0 technology, while CSWS_JAVA V2.x provides Java Servlet 2.3 and JSP 1.2 technology.

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 web site for CSWS_JAVA requirements.

1.2.3 CSWS_PHP Requirements

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 web site at http://h71000.www7.hp.com/openvms/products/ips/apache/csws_php.html for CSWS_PHP requirements.

1.2.4 CSWS_PERL Requirements

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 Version 2.1-1 kit is based on Apache 2.0.52. The Secure Web Server source code 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.

The build instructions are guidelines only and will require modification and customization for the OpenVMS environment used for the build. Other build prerequisites, such as compilers, might also exist.

Chapter 2
Installation and Configuration

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 in the following sections.

2.1 Read the Release Notes

Before you begin the installation, you should read the *HP Secure Web Server for OpenVMS Release Notes* available at http://h71000.www7.hp.com/openvms/products/ips/apache/csws_relnotes_21.html.

2.2 Install the Secure Web Server and Optional Modules

The following kits are intended to work together:

- Secure Web Server for OpenVMS (SWS) Version 2.1 and higher – or – 1.3-1
- CSWS_PHP Version 1.3 and higher
- CSWS_PERL Version 2.1 and higher and PERL for OpenVMS Version 5.8.6 and higher
- CSWS_JAVA Version 3.0 and higher

Note

Earlier versions of these optional kits will not work with the Secure Web Server V2.1-1. Current versions of these optional kits will not work with the Secure Web Server V2.0.

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 and CSWS_PHP must be installed in the same directory (required).

By default, the Secure Web Server 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 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. The Secure Web Server for OpenVMS kit is provided as a compressed, self-extracting file. To download it from the OpenVMS web site, fill out and submit the registration form at Secure Web Server for OpenVMS web site at http://h71000.www7.hp.com/openvms/products/ips/apache/csws.html.

Download any optional modules you want to install.

Download CSWS_JAVA from
http://h71000.www7.hp.com/openvms/products/ips/apache/csws_java_relnotes.html

Download CSWS_PHP from
http://h71000.www7.hp.com/openvms/products/ips/apache/csws_php_relnotes.html

Download CSWS_PERL from
http://h71000.www7.hp.com/openvms/products/ips/apache/csws_modperl_relnotes.html

Download PERL for OpenVMS from
http://h71000.www7.hp.com/openvms/products/ips/apache/csws_perl_relnotes.html

2. Log in as a privileged OpenVMS user (for example, SYSTEM).

3. 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.

   In the SHOW DEVICE/FULL output, the important piece of information is towards the bottom of the output: Vol Prot.

The APACHE$WWW UIC must have RWD access to the volume as indicated in the Vol Prot field and at least E access to the destination directory on that volume where Apache is installed.

For example, if Apache is installed to DISK$APPS:[000000], then APACHE$WWW must have at least RWD access to the DISK$APPS disk volume and at least E access to the DISK$APPS:[000000]000000.DIR directory.

For example:

```
$ SHOW DEVICE/FULL DKB0:

Disk $DKB0:, device type COMPAQ BD03664545, is online, mounted, file-oriented
device, shareable, available to cluster, error logging is enabled

Owner process                  ""    Owner UIC                       [SYSTEM]
Owner process ID        00000000    Dev Prot            S:RWPL,O:RWPL,G:R,W
Reference count               29    Default buffer size                512
Total blocks            71132000    Sectors per track                  254
Total cylinders            14003    Tracks per cylinder                 20

Volume label             "BUILD1"    Relative volume number               0
Cluster size                    3    Transaction count                   25
Free blocks              52293678    Maximum files allowed           8891500
Extend quantity                 5    Mount count                           1
Mount status               System    Cache name     "_ALPHA$DKA300:XQPCACHE"
Extent cache size              64    Maximum blocks in extent cache  5229367
File ID cache size             64    Blocks in extent cache             2703
Quota cache size                0    Maximum buffers in FCP cache       1730
Volume owner UIC          [SYSTEM]    Vol Prot    S:RWCD,O:RWCD,G:RWCD,W:RWCD

Volume Status:  ODS-5, subject to mount verification, write-back caching enabled,
access dates enabled, hard links enabled.
```

4. Decompress the server kit with one of the following command, depending on which platform on which you will install the kit:

```
$ RUN CPQ-AXPVMS-CSWS-V0201-1-1.PCSI_SFX_AXPEXE ! for Alpha
$ RUN HP-I64VMS-CSWS-V0201-1-1.PCSI_SFX_I64EXE  ! for I64
```

   The system expands the file and names it `CPQ-AXPVMS-CSWS-V0201-1-1.PCSI` or `HP-I64VMS-CSWS-V0201-1-1.PCSI`. Do not rename the file.

5. If you are upgrading from a previous version of the Secure Web Server and you modified the file [APACHE.CONF]MIME.TYPES, 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.) See the Merge Changes to Files You Have Customized section for more information.

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 PRODUCT 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 PRODUCT REMOVE CSWS command, then enter PRODUCT 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. 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.

2.2.1 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 V2.1-1                  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 V2.1-1

    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 V2.1-1                  USER$DISK3:[000000.]

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

The following product has been installed:
    CPQ AXPVMS CSWS V2.1-1                  Layered Product

CPQ AXPVMS CSWS V2.1-1

    Release notes are available in SYS$HELP:CSWS0201.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 configuration tool ensures that a user account is available to run the server and that all of the files are owned by that user. It also allows the system manager flexiblity in defining options for the installation.

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 OpenVMS Cluster Considerations before you continue with the configuration.

2.3.1 Configuration Menu

Version 2.1-1 of the Secure Web Server includes a configuration menu that allows you to choose configuration functions. All of the functions listed can be run independently or through the menu driven procedure. The menu also allows you to start and stop instances of the Secure Web Server.

To run the configuration menu, enter the following command:

```
$ @APACHE$COMMON:[000000]APACHE$MENU
```

Following is an example of the configuration menu:

```
            Apache$Menu

     1. Configure the Secure Web Server
     2. Create an Apache instance
     3. Delete an Apache instance
     4. Manage suEXEC users
     5. Run OpenSSL Certificate tool
     6. Convert directory tree to Stream_LF
     7. Start up an Apache instance
     8. Shut down an Apache instance
     9. Show status of an Apache instance
    10. Add a node to CSWS in a cluster environment
    11. Exit

  Enter Menu Choice:
```

The menu choices correspond to running the following procedures or commands from the DCL command line:

1. SYS$MANAGER:APACHE$CONFIG.COM
2. APACHE$COMMON:[000000]APACHE$CREATE_ROOT.COM
3. APACHE$COMMON:[000000]APACHE$DELETE_ROOT.COM
4. APACHE$COMMON:[000000]APACHE$MANAGE_SUEXEC.COM
5. APACHE$COMMON:[000000]APACHE$CERT_TOOL.COM
6. APACHE$COMMON:[000000]APACHE$CONVERT_STREAMLF.COM
7. SYS$STARTUP:APACHE$STARTUP.COM
8. SYS$STARTUP:APACHE$SHUTDOWN.COM
9. SHOW SYSTEM/PROCESS=APACHE$*tag*
10. APACHE$COMMON:[000000]APACHE$ADDNODE.COM

2.3.2 Configuring a Single Server

Choosing Option 1 from the Secure Web Server Configuration Menu starts the following command procedure, which configures a single server:

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

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 the Configuring Multiple Servers section.

To configure a single server, enter one of the following commands:

```
$ @SYS$MANAGER:APACHE$CONFIG
$ @APACHE$COMMON:[000000]APACHE$MENU and select Option 1
```

2.3.3 Sample Configuration of a Single Server

This section shows a sample configuration dialog.

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

                HP Secure Web Server for OpenVMS
                       [based on Apache]

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

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]

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

Enable suEXEC? [YES]

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

Enabling suEXEC configuration.  This could take a minute or two.  . . .

      APACHE$MANAGE_SUEXEC

 This procedure allows the system manager to grant
 users the ability to utilize the suEXEC feature of
 the Secure Web Server.  Users will be granted/revoked
 VMS rights identifiers to allow access.

Continue [YES]?

Enter '?' for help

Manage suEXEC user accounts (SHOW/GRANT/REVOKE/DONE/?): [DONE] GRANT
Enter Username: USER1
%UAF-I-GRANTMSG, identifier APACHE$SUEXEC_USER granted to USER1

Manage suEXEC user accounts (SHOW/GRANT/REVOKE/DONE/?): [DONE] GRANT
Enter Username: USER2
%UAF-I-GRANTMSG, identifier APACHE$SUEXEC_USER granted to USER2

Manage suEXEC user accounts (SHOW/GRANT/REVOKE/DONE/?): [DONE]

Configuration is complete.  To start the server:

     $ @SYS$STARTUP:APACHE$STARTUP.COM
```

2.3.4 Configuring Multiple Servers

Choosing Option 2 from the Secure Web Server Configuration Menu starts the following command procedure, which creates a new server root:

```
APACHE$COMMON:[000000]APACHE$CREATE_ROOT.COM
```

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

The Secure Web Server supports running multiple servers monitoring different ports. This is accomplished by defining an alternate root that is defined as APACHE$SPECIFIC. These instances are created by entering one of the following commands:

```
$ @APACHE$COMMON:[000000]APACHE$CREATE_ROOT
$ @APACHE$COMMON:[000000]APACHE$MENU and select Option 2
```

This procedure defines a new server root with a different user that runs the server, monitors a different port, and has unique startup and shutdown procedures. The instance can also be restricted as to what kinds of privileged routines can be accessed. The user that is defined is automatically granted access to the APACHE$COMMON area.

2.3.5 Sample Configuration of Multiple Servers

Following is an example of creating a new server root:

```
$ @APACHE$COMMON:[000000]APACHE$CREATE_ROOT

  APACHE$CREATE_ROOT

Create a set of directories and files where a Secure
Web Server can run.  You will be prompted for the
location of the root, the user to run under, the
TCP/IP port to monitor, the unique server tag, the
privileged routines the user will be allowed to use,
and optional startup and shutdown procedures.

Continue [YES]?

Root location:  Give the location of where to create the directory
tree and configuration template file for the new instance of the server.

   e.g.  USER2:[SMITH.CSWS]

This will create a series of directories under the USER2:[SMITH.CSWS]
directory.  This will become the new APACHE$SPECIFIC location.

   $ DIRECTORY USER2:[SMITH.CSWS]

   Directory USER2:[SMITH.CSWS]

   BIN.DIR;1          CGI-BIN.DIR;1      CONF.DIR;1         HTDOCS.DIR;1
   ICONS.DIR;1        KIT.DIR;1          LOGS.DIR;1         MODULES.DIR;1
   OPENSSL.DIR;1

   Total of 9 files.
Root Location: DISK1:[JOE.APACHE]
```

```
 Username: Enter the user that will own and control the content of
 this root.  The ownership of the directories and files will be set
 to the given user.  The user must be a valid user in the SYSUAF.

Username: JOE

 The Secure Web Server has several privileged routines to allow the
 server to run in a basic fashion.  These routines can be blocked from
 other users of the web server to run in a more restrictive mode.
 These routines are protected by a series of rights identifiers:

    APACHE$APR_ALL           Allow access to all of the protected routines
    APACHE$APR_CREMBX        Allow access to create a groupwide mailbox
    APACHE$APR_GETPWNAM      Allow access to other user's information
    APACHE$APR_SETSOCKOPT    Allow user to set socket options
    APACHE$APR_SOCKET        Allow creation of a privileged socket
    APACHE$APR_AUTH_OPENVMS  Allow user to authorize using SYSUAF
    APACHE$APR_GALAXY_GBLSEC Allow user to manage galactic memory sections
 Grant access to ALL routines? n
 Grant access to CreMbx? y
 Grant access to GetPwNam? n
 Grant access to SetSockOpt? n
 Grant access to Create a Priveleged Socket? y
 Grant authorization via SYSUAF? y
 Grant user ability to access galactic sections? y
 Each instance of the Secure Web Sever must have a unique TCP/IP port
 to monitor as it runs.  If you have not granted this user the Socket
 privilege, then the port must be greater than 1024 (non-privileged).
 Note that this routine does not keep track of previously specified
 ports to other instances.  It is the system manager's responsibility to
 maintain this information.
 Port number: 81

 Each instance of the Secure Web Server must have a unique tag associated
 with it on the system.  The tag is 1 to 4 characters (A-Z, 0-9).
 Unique Tag: Tst

 The instance of Secure Web Server can have a startup and a shutdown
 command procedure defined to run accordingly.

 Define a startup or shutdown procedure? y
 Startup procedure filename [NONE]: DISK1:[JOE.APACHE]Test_Start.com
 Shutdown procedure filename [NONE]:

 Granting rights to JOE UAF account...
 Creating directory tree under DISK1:[JOE.APACHE]
 Generating Apache configuration file DISK1:[JOE.APACHE.CONF]httpd.conf
 Updating the configuration database
 Root created: DISK1:[JOE.APACHE]
 Template server configuration file created:
 DISK1:[JOE.APACHE.CONF]httpd.conf
 Please review this file for accuracy.
```

The output of this procedure is a directory tree that matches that found under APACHE$COMMON. Populated in that tree are three files:

- a template configuration file
- a mime type file
- an SSL configuration file

These are all placed in the configuration directory and should be reviewed before attempting to start the server.

2.3.6 Delete Server Instance

Choosing Option 3 from the Secure Web Server Configuration Menu deletes an instance of the server. Choosing this option starts the following command procedure:

```
APACHE$COMMON:[000000]APACHE$DELETE_ROOT.COM
```

An example of the output from this procedure is as follows:

```
$ @APACHE$COMMON:[000000]APACHE$DELETE_ROOT

    APACHE$DELETE_ROOT

 Deletes a previously defined set of directories and
 all files contained therein.  Also revokes all user
 rights granted when the root was created.

Continue [YES]?

     Apache Instances available for deletion

         1. Tst     DISK1:[JOE.APACHE.CONF]httpd.conf
         2. Exit

     Choice: 1


Revoking rights from JOE UAF account...
Deleting all files under DISK1:[JOE.APACHE...]
Updating the configuration database...
Root deleted: DISK1:[JOE.APACHE]
```

2.3.7 Managing suEXEC

Choosing Option 4 from the Secure Web Server Configuration Menu starts the suEXEC management command procedure:

```
APACHE$COMMON:[000000]APACHE$MANAGE_SUEXEC.COM
```

Part of the Secure Web Server configuration involves enabling the suEXEC feature. If you enable suEXEC during configuration, the accounts using the feature are managed from APACHE$MANAGE_SUEXEC.COM. The APACHE$CONFIG.COM procedure automatically calls the suEXEC management procedure.

This procedure can also be run directly (by choosing Option 4) to add or subtract users with the ability to use suEXEC.

See the suEXEC in the Secure Web Server section for more information.

2.3.8 Running the OpenSSL Certificate Tool

Choosing Option 5 from the Secure Web Server Configuration Menu starts the following command procedure, which runs the OpenSSL certificate tool:

```
APACHE$COMMON:[000000]APACHE$CERT_TOOL.COM
```

The OpenSSL certificate tool enables you to view certificates and certificate requests, create certificate requests, sign your own certificate, create your own certificate authority, and sign client certificate requests. Additional hash functions are included.

The OpenSSL certificate tool is described in the HP Secure Web Server SSL User Guide.

2.3.9 Converting Files to Stream_LF

Choosing Option 6 from the Secure Web Server Configuration Menu starts the following command procedure, which converts your files to Stream_LF format:

```
APACHE$COMMON:[000000]APACHE$CONVERT_STREAMLF.COM
```

Note

The Secure Web Server Version 2.1-1 **no longer requires** that all served files be in Stream_LF format.

The EnableMMAP directive must be set to OFF to lift the Stream_LF restriction. In V2.1-1, EnableMMAP is set to OFF by default. (In V2.0, the default for EnableMMAP was ON.)

If you want your files in STREAM_LF format, you can use the APACHE$CONVERT_STREAMLF command procedure to recursively search down a directory tree for sequential files and convert them to Stream_LF. The command procedure excludes some sequential files; in particular, it ignores directory files, executable files (such as command procedures, OpenVMS images, CGI, PHP, and Perl scripts), object files, indexed files, and relative files.

The procedure is non-destructive and will create a newer version of the converted file and leave the original in place. The results of the conversion are stored in SYS$SCRATCH:CONVERT_DIR.LOG.

Note

The APACHE$CONVERT_STREAMLF command procedure converts all sequential files (with the exceptions listed above) to Stream_LF format, including sequential files currently in Stream format. After you run the procedure, be sure to check the SYS$SCRATCH:CONVERT_DIR.LOG file for files that should not be in Stream_LF format, and delete the newest version of those files.

The following is an example output of the convert utility:

```
$ @APACHE$COMMON:[000000]APACHE$CONVERT_STREAMLF
Top Directory: USER1:[APACHE.HTDOCS]

Starting conversion of USER1:[APACHE.HTDOCS...]
```

```
   This could take a while...

   Conversions complete.
   See SYS$SCRATCH:Convert_Dir.Log for a log of transactions.
```

2.3.10 Starting and Stopping the Secure Web Server

Choosing Option 7 and Option 8 from the Secure Web Server Configuration Menu runs the following command procedures which start up and shut down instances of the Secure Web Server:

```
SYS$STARTUP:APACHE$STARTUP.COM
SYS$STARTUP:APACHE$SHUTDOWN.COM
```

See Starting and Stopping the Server for more information.

2.3.11 Showing the Status of an Apache Instance

Choosing Option 9 from the Secure Web Server Configuration Menu runs the following command:

```
$ SHOW SYSTEM/PROCESS=APACHE$tag
```

Server processes have a process tag of the form APACHE$ssss, where *ssss* is up to four alphanumeric characters defined in the VMSServerTag directive. The default is APACHE$SWS.

Similarly, child processes have a process name of the form APACHE$ssssnnnn, where *APACHE$ssss* is the server name and *nnnn* is the child server process number represented as a hex value.

The SHOW SYSTEM/PROCESS=APACHE$*tag* command lists a menu of the current instances of the server. You choose the instance for which you want to see status.

The following is an example output showing the status of a running server:

```
   Registered Apache Instances

     1. SWS     APACHE$COMMON:[CONF]HTTPD.CONF
     2. Exit

     Choice: 1

Status of SWS instance of Apache...


OpenVMS V7.3-2 on node APSERV   1-AUG-2005 15:55:34.09  Uptime  67 06:17:52
  Pid    Process Name    State  Pri      I/O       CPU       Page flts  Pages
2020026D APACHE$SWS      LEF      6     2526   0 00:00:11.35       839   1016
2020026F APACHE$SWS0000  LEF      6     2556   0 00:00:12.69       824    979
20200270 APACHE$SWS0001  LEF      6     2530   0 00:00:09.41       834   1010
20200271 APACHE$SWS0002  LEF      6     2493   0 00:00:14.00       811    978
20200272 APACHE$SWS0003  LEF      6     2499   0 00:00:13.66       822    988
20200273 APACHE$SWS0004  LEF      6     2487   0 00:00:12.01       832   1002
20200274 APACHE$SWS0005  LEF      6     2501   0 00:00:15.22       810    994

End status.
```

The following is an example output showing the status of a server that has been shut down:

```
   Registered Apache Instances

     1. SWS     APACHE$COMMON:[CONF]HTTPD.CONF
     2. Exit

     Choice: 1

Status of SWS instance of Apache...

End status.
```

2.3.12 Adding a Node to CSWS in a Cluster Environment

Choosing Option 10 from the Secure Web Server Configuration Menu starts the following command procedure, which adds a node to the Secure Web Server in a cluster environment.

```
    APACHE$COMMON:[000000]APACHE$ADDNODE.COM
```

You must log into the system you want to add as a CSWS cluster member before you choose Option 10. For example, perform the initial installation and configuration of CSWS on NODE1. Then log into NODE2 and enter the following commands:

```
      $ @SYS$STARTUP:APACHE$LOGICALS
      $ @APACHE$COMMON:[000000]APACHE$MENU

      Apache$Menu

                  1. Configure the Secure Web Server
                  2. Create an Apache instance
                  3. Delete an Apache instance
                  4. Manage suEXEC users
                  5. Run OpenSSL Certificate tool
                  6. Convert directory tree to Stream_LF
                  7. Start up an Apache instance
                  8. Shut down an Apache instance
                  9. Show status of an Apache instance
                 10. Add a node to CSWS in a cluster environment
                 11. Exit

      Enter Menu Choice: 10

      APACHE$ADDNODE

Create a set of directories and files on another node in a cluster environment
for the Secure Web Server.  The node name used is that defined by
TCPIP$INET_HOST.  A directory by that name will be created under the
APACHE$SPECIFIC: area.  The top level directories under APACHE$COMMON are
essentially duplicated here.

A new version of HTTPD.CONF is created in APACHE$ROOT:[CONF].  This will be
used by default.  The common configuration in APACHE$COMMON:[CONF] remains
untouched. Remove this new configuration if you wish to use the common one.
```

```
The rights identifiers for the user account APACHE$WWW on this node are set to
the defaults.  If this is a common SYSUAF/RIGHTSLIST, then the account should
be checked as it might be changed.

Continue [YES]?  yes
Granting rights to APACHE$WWW UAF account...
Creating directory tree under device:[000000.APACHE.SPECIFIC.node-name]
Generating Apache configuration file device:[000000.APACHE.SPECIFIC.node-
name.CONF]httpd.conf

Node node added successfully
 Node specific directories created: device:[000000.APACHE.SPECIFIC.node-name]
 Configuration files created in: device:[000000.APACHE.SPECIFIC.node-name.CONF]
 Please review these files for accuracy.

Press return to continue...
```

Exit the configuration menu, then enter the following command to start the Secure Web Server on NODE2:

```
$ @sys$startup:apache$startup
```

2.3.13 Managing Multiple Servers

This section discusses the issues you may encounter when managing multiple servers.

2.3.13.1 HTTPD.CONF

Because there are multiple servers at work, there must be some differences in the HTTPD.CONF file for each server.

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 set the processwide logical name APACHE$SPECIFIC to the configuration-specific directory. You then edit the file APACHE$SPECIFIC:[CONF]HTTPD.CONF.

2.3.13.2 APACHE$SETUP.COM and LOGIN.COM

APACHE$COMMON:[000000]APACHE$SETUP.COM is run for every server (parent and child) and server instance. This command procedure defines the necessary Apache symbols and executes any subsequent product setups if they exist (for example, PHP and Perl). It also defines the CRTL logicals needed to allow the Secure Web Server to run correctly with extended command parsing and file specifications.

The APACHE$ROOT:[000000]LOGIN.COM command procedure is executed after APACHE$SETUP.COM and is determined by the LGICMD stored in SYSUAF for the Apache server user (for example, APACHE$WWW).

The Secure Web Server includes APACHE$SETUP.COM so that each instance of the server can use its own LOGIN.COM procedure, and not have to maintain server critical definitions.

2.3.14 Viewing the OpenSSL 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 HP Secure Web Server SSL User Guide.

After configuring the Secure Web Server, do not start the server. Follow the instructions in the Post Configuration Checklist section.

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. Optionally check the CSWS_PHP configuration now or later.
3. Optionally check the CSWS_PERL 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. If you want to change the default configuration, edit APACHE$COMMON:[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, 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. Because of the Secure Web Server installation, 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 DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
$ MOD APACHE$WWW/PRIV=EXQUOTA
$ EXIT
```

Stop and restart the Secure Web Server so that the APACHE$WWW account picks up the new privilege.

2.4.6 Check for SET TERMINAL/INQUIRE

When the Secure Web Server for OpenVMS is started, the command procedure APACHE$SETUP is executed. 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

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. (In TCP/IP Services Version 5.3 for OpenVMS and higher, user input is not echoed.

Use the following procedure to test the installation.

Enter the following command:

```
$ TELNET 0 80
```

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 ^]
```

Press ENTER and enter the following HTTP command:

```
HEAD / HTTP/1.0
```

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

```
HTTP/1.1 200 OK
Date: Wed, 21 Sep 2005 21:16:37 GMT
Server: Apache/2.0.52 (OpenVMS) mod_ssl/2.0.52 OpenSSL/0.9.7d
Content-Location: index.html.en
Vary: negotiate,accept-language,accept-charset
TCN: choice
Last-Modified: Thu, 08 Sep 2005 20:41:57 GMT
ETag: "2e4550-5b2-b12cef40"
Accept-Ranges: bytes
Content-Length: 1458
Connection: close
Content-Type: text/html; charset=ISO-8859-1
Content-Language: en

%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$ROOT:[000000]APACHE$*tag*
  APACHE$ROOT:[LOGS]ERROR_LOG

- If you have trouble starting the server, enable the logical APACHE$SPL_DISABLED systemwide, then restart the server.
- If you have trouble stopping the server using the APACHE$SHUTDOWN command and 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>
  ```

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 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 the Merge Changes to Files You Have Customized section.
- If you enabled MOD_SSL, follow the instructions for verifying SSL in the *HP 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

If you have installed a previous version or field test 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.

2.8 Installing Optional Modules at a Later Time

If you did not install the optional modules (CSWS_JAVA, CSWS_PERL, or CSWS_PHP) when you installed the server, follow these instructions for installing them at a later time. Before you begin, make sure:

- You have installed the required software.
- You have already installed the Secure Web Server.
- You install CSWS_PHP in the same directory as you installed the server. You do not need to install CSWS_JAVA or CSWS_PERL into the same disk or directory as the Secure Web Server.

Use the appropriate command from the list below.

To install CSWS_JAVA, use the following command:

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

To install CSWS_PHP, use the following command:

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

To install CSWS_PERL, use the following command:

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

The installation is complete when the dollar sign prompt ($) is displayed.

After you install CSWS_JAVA, you must configure it. For more information, see Configure CSWS_JAVA.

CSWS_PHP and CSWS_PERL are preconfigured, but you can change the configurations. For more information, see Check the CSWS_PHP Configuration and Check the CSWS_PERL Configuration.

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 for a configuration file. If you do not specify a value for *configuration-file*, HTTPD.CONF is the default.

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 for a configuration file. If you do not specify a value for *configuration-file*, HTTPD.CONF is the default.

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 are unable to shut down the server, use the following command to check whether APACHE$WWW processes are still running:

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

3.1.2.1 Stopping the Server Using the Server Process Name

If you are unable to shut down the server using the APACHE$SHUTDOWN command, and APACHE$WWW is still running, you can use the server PID to stop it. To determine the server PID, enter the following command (or choose Option 9 from the configuration menu):

```
$ SHOW SYSTEM/PROCESS=APACHE$tag
```

Server processes have a process tag of the form APACHE$ssss, where *ssss* is up to four alphanumeric characters defined in the VmsServerTag directive. The default is APACHE$SWS.

You should then be able to shut down the server by entering the following command:

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

3.2 Server Log File

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

```
APACHE$SPECIFIC:[000000]APACHE$tag
```

Server processes have a process tag of the form APACHE$ssss, where *ssss* is up to four alphanumeric characters defined in the VMSServerTag directive. The default is APACHE$SWS.

Similarly, child processes have a process name of the form APACHE$ssssnnnn, where *APACHE$ssss* is the server name and *nnnn* is the child server process number represented as a hex value. For example:

```
Parent      APACHE$SWS
Child 1     APACHE$SWS0000
Child 2     APACHE$SWS0001
Child 3     APACHE$SWS0002
```

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 create	8	20 You should increase this value for a heavy load.
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

Periodically, check the server's 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

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.

3.5 Modules and Directives

3.5.1 Apache Modules

Following is a list of the modules included in the Secure Web Server for OpenVMS Version 2.1-1 kit. The **Secure Web Server Version 2.1-1 includes all of the Apache 2.0 modules and directives** as well as two OpenVMS-specific modules, MOD_AUTH_OPENVMS and MOD_OSUSCRIPT. The Apache modules and directives function as documented in Apache Version 2.0 Modules and Apache Version 2.0 Directives.

The server documentation from the Apache Software Foundation provides the information needed to use the modules and directives.

Note

In SWS V2.1-1, many loadable modules are no longer loaded by default. You must uncomment the modules in `httpd.conf` to load them. (See the file `httpd-vms.conf` for other modules you may want to load.)

All modules are loadable, with the exception of those that are marked as built into the server.

CORE *(Built into server)*

HTTP_CORE *(Built into server)*

MOD_ACCESS

MOD_ACTIONS

MOD_ALIAS

MOD_AUTH_ANON

MOD_ASIS

MOD_AUTH

MOD_AUTH_DBM

MOD_AUTH_DIGEST

MOD_AUTH_KERB

MOD_AUTH_LDAP

MOD_AUTH_OPENVMS *(OpenVMS specific)*

MOD_AUTOINDEX

MOD_CACHE

MOD_CASE_FILTER

MOD_CASE_FILTER_IN

MOD_CERN_META

MOD_CHARSET_LITE

MOD_CGI

MOD_DAV

MOD_DAV_FS

MOD_DEFLATE

MOD_DIR

MOD_DISK_CACHE

MOD_ECHO

MOD_ENV

MOD_EXAMPLE

MOD_EXPIRES

MOD_FILE_CACHE

MOD_EXT_FILTER

MOD_HEADERS

MOD_IMAP

MOD_INCLUDE

MOD_INFO

MOD_LOG_CONFIG

MOD_LOGIO

MOD_MIME

MOD_MIME_MAGIC

MOD_NEGOTIATION

MOD_OSUSCRIPT *(OpenVMS specific)*

MOD_PROXY

MOD_PROXY_CONNECT

MOD_PROXY_FTP

MOD_PROXY_HTTP

MOD_REWRITE

MOD_SETENVIF

MOD_SO *(Built into server)*

MOD_SPELING

MOD_SSL

MOD_STATUS

MOD_SUEXEC

MOD_UNIQUE_ID

MOD_USERDIR

MOD_USERTRACK

MOD_VHOST_ALIAS

PREFORK *(Built into server)*

3.5.2 Apache 1.3 Modules Not Included

The following modules were part of the Apache 1.3 stream, but are **not** included in Apache 2.0. Therefore, these modules are not included in the Secure Web Server Version 2.1-1 kit.

MOD_AUTH_DB
MOD_DEFINE
MOD_ISAPI
MOD_LOG_AGENT
MOD_LOG_REFERER
MOD_MMAP_STATIC

3.5.3 OpenVMS Directives

The following directives are specific to the Secure Web Server for OpenVMS.

VMSServerTag

This directive is a replacement for the APACHE$SERVER_TAG logical name used in the Secure Web Server Version 1.3. This required directive distinguishes multiple instances of the Apache server, and contains between 1 and 4 alphanumeric characters. This server tag will be used in the server process names for the given instance (for example, APACHE$SWS).

Default: SWS
Example: `VMSServerTag SWS1`

VMSServerStartup

This directive is a replacement for the APACHE$SERVER_STARTUP logical name used in the Secure Web Server Version 1.3. This optional directive specifies a user-developed command procedure that should be executed prior to the activation of each Apache server process.

Default: none
Example:

```
VMSServerStartup APACHE$COMMON:[000000]APACHE$SERVER_STARTUP.COM
```

VMSServerShutdown

This directive is a replacement for the APACHE$SERVER_SHUTDOWN logical name used in the Secure Web Server Version 1.3. This optional directive specifies a user-developed command procedure that should be executed after the activation of each Apache server process.

Default: none
Example:

```
VMSServerShutdown APACHE$COMMON:[000000]APACHE$SERVER_SHUTDOWN.COM
```

3.5.4 Command Line Options

This section describes the command line options supported in the Secure Web Server. You must run APACHE$SETUP.COM before using the command line interface or any other Apache image (such as htpasswd), as follows:

```
$ @APACHE$COMMON:[000000]APACHE$SETUP
```

You can use the command line interface for managing multiple servers or for performing complex operations. HP recommends using the startup and shutdown procedures for simple management of a single server environment. See Starting and Stopping the Server for more information.

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
-D name	Define a name for use in directives
-d directory	Specify an alternate initial ServerRoot
-f file	Specify an alternate ServerConfigFile
-C "directive"	Process directive before reading config files
-c "directive"	Process directive after reading config files
-e level	Show startup errors of level (see LogLevel)
-E file	Log startup errors to file
-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
-t -D DUMP_VHOSTS	Show parsed settings (currently only vhost settings)
-S	A synonym for -t -D DUMP_VHOSTS
-t	Run syntax check for config files

Enter the following command to see a list of available options, as follows:

```
    $ httpd -?

    Usage: dka200:[apache]apache$httpd.exe;1 [-D name] [-d directory] [-f file]
                                             [-C "directive"] [-c "directive"]
                                             [-k start|restart|graceful|stop]
                                             [-v] [-V] [-h] [-l] [-L] [-t] [-S]
Options:
  -D name           : define a name for use in  directives
  -d directory      : specify an alternate initial ServerRoot
  -f file           : specify an alternate ServerConfigFile
  -C "directive"    : process directive before reading config files
  -c "directive"    : process directive after reading config files
  -e level          : show startup errors of level (see LogLevel)
  -E file           : log startup errors to file
  -v                : show version number
  -V                : show compile settings
  -h                : list available command line options (this page)
  -l                : list compiled in modules
  -L                : list available configuration directives
  -t -D DUMP_VHOSTS : show parsed settings (currently only vhost settings)
  -S                : a synonym for -t -D DUMP_VHOSTS
  -t                : run syntax check for config files
```

3.5.5 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 on 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.

3.5.6 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.

3.5.7 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.

3.5.8 Content Negotiation

The MOD_NEGOTIATION module provides content negotiation. This module lets you specify language variants of HTML files.

Note

Beginning with the Secure Web Server Version 2.0, you must specify language variants on OpenVMS systems in the same way as you do on UNIX systems, using multiple dots in the filename. For example, the French variant of a filename is *filename.html.fr*.

In previous versions of the Secure Web Server, you would use an underscore instead of a dot before the language extension (for example, *filename.html_fr*).

For more information, see Content Negotiation Documentation.

3.5.9 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.

3.5.10 WebDAV (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.

Note

WebDAV support requires the VDBM database manager type. VDBM is the default. SDBM and GDBM are not supported in this kit.

To change the database manager type, set the logical name `APACHE$DAV_DBM_TYPE` so that it is visible to Apache, such as in `APACHE$COMMON:[000000]LOGIN.COM`.

For more information, see the MOD_DAV website.

3.5.10.1 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.

3.5.11 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.

For more information about suEXEC, see the Apache.org website.

See suEXEC in the Secure Web Server for configuration information.

3.5.12 MOD_SSL

You can improve SSL cache performance in a Galaxy cluster by placing a memory-based SSL session cache in a Galaxy shared-memory partition. After the partition is created, use the steps in the following section to configure SWS to use the Galaxy partition for the session cache.

3.5.12.1 Setting up a Galaxy Shared-Memory SSL Session Cache

The SSLSessionCache directive on all GALAXY members must be the same:

```
SSLSessionCache   cshm:logs/ssl_scache(512000)
SSLMutex          csem:<anythingyouwant>
```

These directives create the Galaxy global section APACHE$SSL_CSHM_logs/ssl_scache and use the APACHE$SSL_CSHM_<anythingyouwant> mutex to synchronize access to the section. The number in parentheses is the size in bytes of the cache.

These directives are supported only when the system is running as both a Galaxy member and a member of a cluster.

3.5.13 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

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.

3.6 File Formats

The Secure Web Server Version 2.1-1 kit **no longer requires** all served files to be in Stream_LF format. If you still want your files in Stream_LF format, see Converting files to Stream_LF for information about a command procedure that automatically converts your files.

The EnableMMAP directive must be set to OFF to lift the Stream_LF restriction. In V2.1-1, EnableMMAP is set to OFF by default. (In V2.0, the default for EnableMMAP was ON.)

Note

In the Secure Web Server V2.1-1, the Web browser status bar shows page loading progress, which requires an accurate byte count. To obtain an accurate byte count for Variable and VFC format files, the Secure Web Server reads each file twice. You can get better efficiency from a Stream_LF format file because the Secure Web Server reads it only once.

3.7 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]*.*
```

3.8 Logical Names

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

Table 3-3 System Defined Logical Names

Logical Name	Description
APACHE$APU_SHR (New in Version 2.0)	System executive mode logical name that points to the Apache Runtime Utility shared image that provides common utility routines needed by the Secure Web Server. If no logical is specified, the default is APACHE$COMMON:[000000]APACHE$APU_SHR.EXE.
APACHE$APR_SHR (New in Version 2.0)	System executive mode logical name that points to the Apache Runtime Utility shared image that provides common routines needed by the Secure Web Server. If no logical is specified, the default is APACHE$COMMON:[000000]APACHE$APR_SHR.EXE.
APACHE$APR_SHRP (New in Version 2.0)	System executive mode logical name that points to the Apache Runtime protected shared image which provides privileged access routines needed by the Secure Web Server. If no logical is specified, the default is APACHE$COMMON:[000000]APACHE$APR_SHRP.EXE.
APACHE$COMMON	Concealed logical name that defines clusterwide files in APACHE$ROOT (device:[APACHE]).
APACHE$HTTPD_SHR (New in Version 2.0)	System executive mode logical name that points to the Apache HTTPD shared image that provides server routines needed by the Secure Web Server. If this logical is not set, the default is APACHE$COMMON:[000000]APACHE$HTTPD_SHR.EXE.

Table 3-4 Process Logical Names

Logical Name	Description
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 (Obsolete in Version 2.*x*)	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$SERVER_TAG (Obsolete in Version 2.*x*)	Defines the currently running server's tag and is intended *for reference only*. This logical contains the 1-4 character server tag defined via the VmsServerTag directive. If no logical is specified, the default is SWS.
APACHE$SPECIFIC	Concealed logical name that defines system-specific files in APACHE$ROOT (device:[APACHE.SPECIFIC.node-name]).

Note

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.

Table 3-5 User Defined Logical Names

Logical Name	Description
APACHE$BG_PIPE_BUFFER_SIZE (New in Version 2.0)	System logical name that is used to set the socket pipe buffer size for exec functions. If this logical is not set, the default is 32767.
APACHE$CGI_BYPASS_OWNER_CHECK (Obsolete in V2.*x*)	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$CREATE_SYMBOLS_GLOBAL	If defined, this system logical name causes CGI environment symbols to be defined globally. They are defined locally by default.
APACHE$DAV_DBM_TYPE (New in Version 2.0)	Used to define the desired DBM organization to use for MOD_DAV. The valid options for this logical are: GDBM, SDBM, VDBM. If this logical is not set, the default is VDBM.
APACHE$DEBUG_DCL_CGI	If defined, this system logical name enables APACHE$VERIFY_DCL_CGI and APACHE$SHOW_CGI_SYMBOL.
APACHE$DL_CASE (New in Version 2.0)	System logical name that controls how Apache will locate shareable entry points. There are four different options. Note that only one option is available at a time.
	1 Entry points are located using upper case search. 2 Entry points are located using mixed case search. 3 Default. Entry points are located using upper case search, then mixed case search. 4 Entry points are located using mixed case search, then upper case search.
APACHE$DL_FORCE_UPPERCASE (Obsolete in Version 2.*x*)	If defined to be true (1, T, or Y), this system logical name forces case-sensitive dynamic image activation symbol lookups. By default, symbol lookups are first done in a case-sensitive manner and then, if failed, a second attempt is made using case-insensitive symbol lookups. This fallback behavior can be disabled with APACHE$DL_NO_UPPERCASE_FALLBACK.
APACHE$DL_NO_UPPERCASE_FALLBACK (Obsolete in Version 2.*x*)	If defined to be true (1, T, or Y), this system logical name disables case-insensitive symbol name lookups whenever case-sensitive lookups fail. See APACHE$DL_FORCE_UPPERCASE.
APACHE$FIXBG (Obsolete in Version 2.*x*)	System executive mode logical name pointing to installed, shareable images. Not intended to be modified by the user. Replaced by APACHE$SET_CCL.EXE.
APACHE$FLIP_CCL (New in Version 2.0)	Used by APACHE$SET_CCL.EXE, which replaces APACHE$FIXBG.EXE.
APACHE$INPUT	Used by CGI programs for PUT/POST methods of reading the input stream.
APACHE$MB_PIPE_BUFFER_SIZE (New in Version 2.0)	Used to set the mailbox pipe buffer size for exec functions. If this logical is not set, the default is 4096.
APACHE$PLV_ENABLE_<username> (Obsolete in Version 2.*x*)	System executive mode logical name defined during startup and used to control access to the services provided by the APACHE$PRIVILEGED image. Not intended to be modified by the user.
APACHE$PLV_LOGICAL (Obsolete in Version 2.*x*)	System executive mode logical name defined during startup and used to control access to the services provided by the APACHE$PRIVILEGED image. Not intended to be modified by the user.
APACHE$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$PRIVILEGED (Obsolete in Version 2.*x*)	System executive mode logical name pointing to installed, shareable images. Not intended to be modified by the user.
APACHE$READDIR_NO_DOT_FILES (New in Version 2.0)	Used to disable the simulating of dot files when processing directories. There is no default value.
APACHE$READDIR_NO_NULL_TYPE (New in Version 2.0)	Used to disable the elimination of the null type which contains a single dot when processing directories. There is no default value.
APACHE$READDIR_NO_UNIX_OPEN (New in Version 2.0)	Used to disable the processing of unix files when processing directories. There is no default value.
APACHE$SET_CCL (New in Version 2.0)	Used by APACHE$SET_CCL.EXE, which replaces APACHE$FIXBG.EXE.
APACHE$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$DEBUG_DCL_CGI.
APACHE$SSL_DBM_TYPE (New in Version 2.0)	Used to define the desired DBM organization to use for MOD_SSL. The valid options for this logical are: GDBM, SDBM, VDBM. If this logical is not set, the default is VDBM.
APACHE$SPL_DISABLED (New in Version 2.0)	Used to determine whether Shared Process Logging is to be disabled. There is no default value.
APACHE$SPL_MAX_BUFFERS (New in Version 2.0)	Used to determine the maximum buffer quota for each Shared Process Logging mailbox. If this logical is not set, the default is 10.
APACHE$SPL_MAX_MESSAGE (New in Version 2.0)	Used to determine the maximum message size for each Shared Process Logging mailbox. If this logical is not set, the default is 1024.
APACHE$SPL_FLUSH_INTERVAL (New in Version 2.0)	Used to determine the maximum message count per Shared Process Logging file before data is flushed to disk. If this logical is not set, the default is 256.
APACHE$USE_CUSTOM_STAT (New in Version 2.0)	System logical name that is used to indicate that the custom apache stat function should be used rather than the run-time stat function.
APACHE$USER_HOME_PATH_ UPPERCASE (Obsolete in Version 2.*x*)	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.

3.9 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.

Note

The Secure Web Server runs and has been tested in Alpha-only, I64-only, and mixed-architecture (Alpha and VAX) clusters. (The Secure Web Server does not run on VAX systems but runs on Alpha systems in an Alpha and VAX cluster.)

Although HP expects that SWS will also run in a mixed-architecture (Alpha and I64) cluster, the V2.1-1 kit has not been tested in that configuration.

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.

The rights identifiers for APACHE$* must be the same across the cluster as well. These rights identifiers are as follows:

- APACHE$READ
- APACHE$EXECUTE
- APACHE$SUEXEC_SRVR
- APACHE$SUEXEC_USER

These identifiers are used in ACLs and are attached to disk files that might be shared in a cluster.

3.9.1 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.

3.9.2 Mixed-Architecture (Alpha and VAX) Cluster

In a mixed-architecture cluster of Alpha and VAX systems, do not use a cluster alias IP address with the Secure Web Server. Because the VAX systems will not have the Secure Web Server running, they will not be able to service HTTP requests.

3.10 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
```

3.10.2 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)
- Several detached processes

Note

Although the Apache 2.0 stream contains support for running the server in a hybrid multiprocess, multithreaded mode, the SWS Version 2.1-1 kit is built on a **process-based model**. A threads-based version of the Secure Web Server is under investigation and may be included in a subsequent release.

Server processes have a process name of the form APACHE$ssss, where *ssss* is up to four alphanumeric characters defined in the VmsServerTag directive. The default is APACHE$SWS.

Similarly, child processes have a process name of the form APACHE$ssssnnnn, where *APACHE$ssss* is the server name and *nnnn* is the child server process number represented as a hex value. For example:

```
Parent      APACHE$SWS
Child 1     APACHE$SWS0000
Child 2     APACHE$SWS0001
Child 3     APACHE$SWS0002
```

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 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.

4.3 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.

4.4 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 plain text. 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).

4.4.1 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.

4.4.2 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.

4.4.3 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.

4.4.4 MOD_AUTH_OPENVMS Security Considerations

Installing a privileged shareable image library such as APACHE$APR_SHRP adds some risk to system security. In particular, with APACHE$APR_SHRP 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 user accounts that are granted the appropriate APACHE$APR_* rights identifiers are allowed to run these routines. Frequently, this means that only the Secure Web Server server process account (APACHE$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.

4.4.5 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>
```

4.5 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.

4.6 suEXEC in the Secure Web Server

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

4.6.1 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.

4.6.2 Configuring suEXEC

You can configure suEXEC using the configuration utility provided with the installation (APACHE$COMMON:[000000]APACHE$MENU.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.

suEXEC user files can be deployed outside APACHE$COMMON:[HTDOCS.SUEXEC] in another directory by performing the following steps:

1. Execute the command below to assign the APACHE$WWW user with read and execute access privileges for each directory in the entire directory tree where individual user directories reside:

```
$ Set security/acl=(IDENTIFIER=[AP_HTTPD,APACHE$WWW],ACCESS=READ+EXECUTE) <directory name>
```

Note

You must perform this step for all directories above the respective user directories.

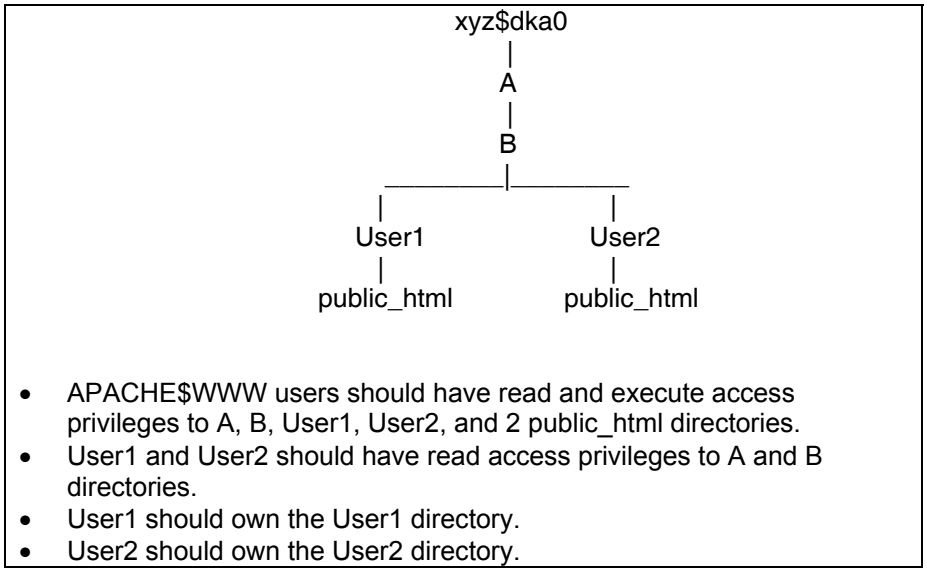
---

2. Ensure that all users have ownership for their respective user directories.
3. Assign the APACHE$WWW user with read and execute access privileges to the user directory.
4. Assign respective suEXEC users with read access privilege to all container directories.
5. Configure either the device name or the logical name.

Note

suEXEC receives the user's home directory details from UAF.

The example below shows a directory structure with user directories as well as the privileges required for each directory.



- APACHE$WWW users should have read and execute access privileges to A, B, User1, User2, and 2 public_html directories.
- User1 and User2 should have read access privileges to A and B directories.
- User1 should own the User1 directory.
- User2 should own the User2 directory.

4.6.2.1 Using Paths with Logicals in UserDir Directive

When the path specified in the UserDir directive in httpd.conf contains a device name, the user's "device" field in UAF must have the same device name.

For example,

In the example above, if XYZ$ROOT points to xyz$dka0:[A.B] and UserDir in "httpd.conf" is defined as "/XYZ$ROOT/*/public_html" then create users User1 and User2 with the "device" field as "XYZ$ROOT" and respective "directory" fields as [User1] and [User2].

4.6.2.2 Using Paths with Device names in UserDir Directive

When the path specified in the UserDir directive in httpd.conf contains a device name, the user's "device" field in UAF must have the same device name.

For example,

In the example above, if UserDir in "httpd.conf" is defined as "/xyz$dka0/a/b/*/public_html" then create users User1 and User2 with the "device" field as "xyz$dka0" and respective "directory" fields as [A.B.User1] and [A.B.User2] respectively.

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.

4.7 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

Note

The instructions in this chapter have **not** been tested by HP on the Secure Web Server Version 2.1-1.

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.

The list of Apache modules that are part of the Apache source code distribution can be found at http://httpd.apache.org/docs-2.0/mod/

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 switch is used to build the server:

/POINTER_SIZE=32

You should use this compiler switch when building your module.

Note

Beginning with the Secure Web Server Version 2.0, the EAPI macro /DEFINE=(EAPI) is no longer required. SSL is built into the Apache base code by default and is no longer patched.

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 Alpha.

```
$! 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"
```

Beginning in the Secure Web Server Version 2.0, a parameter to the SYS$STARTUP:APACHE$STARTUP command procedure is provided to execute the server in a single process.

@SYS$STARTUP:APACHE$STARTUP RUN ! (execute SWS in a single process)

This allows the user to debug the new module as all requests will be directed to the one server process.

This can be shown as an alternative to the step 2 in starting the server.

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

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

Mod_SSL

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

Mod SSL License

OpenSSL

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

OpenSSL License

MOD_PERL

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

Mod_Perl License

Perl

This product includes software developed by the Perl Project.

Perl License

MOD_PHP

This product includes software developed by the PHP Group.

PHP License