vms Software

CSWS_PHP for CSWS for OpenVMS

Installation Guide and Release Notes
December 2015

Note that CSWS_PHP Version 5.2-17A supplied by VMS Software Inc. is equivalent to the Version 5.2-17 kit provided by HP.

What is New

CSWS_PHP Version 5.2-17A is available on OpenVMS Integrity server. For general information about PHP, see http://www.php.net/.

New features included in CSWS_PHP Version 5.2-17A are as follows:

- CSWS_PHP based on PHP 5.2.17; the following changes are included in this release:
  - Fix for critical vulnerability associated to floating point.
  - Updated time zone database to version 2010.5
  - Upgraded bundled PCRE to version 8.02
  - Rewrote `var_export()` to use `smart_str` rather than output buffering, prevents data disclosure if a fatal error occurs
  - Resolved a critical issue, reported as PHP bug #53632
- Security Fixes

  The following security fixes are included in this release:

  - CVE-2011-1464
  - CVE-2011-4885
  - CVE-2011-1148
  - CVE-2011-1938
  - CVE-2011-2202
  - CVE-2011-0421
  - CVE-2011-1092
  - CVE-2011-0708
  - CVE-2010-3807
  - CVE-2006-7243

  A complete list is available at the PHP website http://www.php.net/.

Note: CVE-2011-4566 is applicable to PHP V5.2.17, but only 32-bit PHP binaries are affected by this CVE. The 64-bit PHP binaries are ported to OpenVMS.

- Built PHP MYSQL extension with MYSQL V5.1.23-rc client API.

Software Prerequisites

The CSWS_PHP Version 5.2-17A kit requires that the following software is installed before you install CSWS_PHP:

- OpenVMS for Integrity servers Version 8.4-1H1 or later
- Secure Web Server (CSWS) Version 2.2-1B for OpenVMS

Installing CSWS_PHP for OpenVMS

Important: Earlier versions of CSWS_PHP must be removed manually. Use the command "`PRODUCT REMOVE CSWS_PHP`" to remove earlier version if installed.

Before you install CSWS_PHP (or any optional module), shut down the Secure Web Server. You can restart the server when the installation is complete.

To install the CSWS_PHP, enter the following command:

```
$ PRODUCT INSTALL CSWS_PHP/DESTINATION=DISK1:[WEB_SERVER]
```

Note that you must install the CSWS_PHP kit into the same device and directory where you installed the Secure Web Server for OpenVMS.

For example:

```
$ SHOW LOGICAL APACHE$ROOT
"APACHE$ROOT" = "DISK1:[WEB_SERVER.APACHE.SPECIFIC.hostname.]
= "APACHE$COMMON:"
1 "APACHE$COMMON" = "DISK1:[WEB_SERVER.APACHE.]
```

For a description of `PRODUCT INSTALL` commands, see the PCSI Utility User's Guide.

As the installation procedure for CSWS_PHP Version 5.2-17A progresses, the system displays the following information:

```
The following product has been selected:
    VSI I64VMS CSWS_PHP V5.2-17A           Layered Product

Do you want to continue? [YES]

Configuration phase starting ...

Configuring VSI I64VMS CSWS_PHP V5.2-17A

    VMS Software Inc. & The Apache Software Foundation.

* This product does not have any configuration options.

Execution phase starting ...

The following product will be installed to destination:
    VSI I64VMS CSWS_PHP V5.2-17A           DISK$BIGGLES_SYS:[VMS$COMMON.]

Portion done: 0%...20%...30%...40%...60%...70%...80%...90%...100%

The following product has been installed:
```

```
    VSI I64VMS CSWS_PHP V5.2-17A           Layered Product

VSI I64VMS CSWS_PHP V5.2-17A

    Post-installation tasks are required for PHP for OpenVMS.


    The release notes give detailed directions. This information
    is a brief checklist.

        This installation modifies APACHE$ROOT:[CONF]HTTPD.CONF to
        enable Mod_PHP.  Check HTTPD.CONF for accuracy.  The line
        "Include /apache$root/conf/mod_php.conf" should be the only
        difference.  Also study the Mod_PHP configuration file
        (APACHE$ROOT:[CONF]MOD_PHP.CONF) for options required for your
        site.

        The Apache server must be shutdown and restarted to make these
        changes to the HTTPD.CONF file take place.  Test that Mod_PHP
        is working by accessing the sample script from a browser:

           http://<your web server host>/php/php_rules.php

        Thank you for using PHP for OpenVMS.
```

After the installation is complete, start the Secure Web Server (CSWS) by entering the following command:

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

Removing CSWS_PHP for OpenVMS

You can remove the CSWS_PHP kit by using the `PRODUCT REMOVE` command. The Secure Web Server must be shut down before you remove CSWS_PHP.

To remove CSWS_PHP, enter the following commands:

```
$ @SYS$STARTUP:APACHE$SHUTDOWN
$ PRODUCT REMOVE CSWS_PHP
```

The `PRODUCT REMOVE` command removes all files created by this installation. It also attempts to remove the directories defined by the installation.

Using Extensions

CSWS_PHP Version 5.2-17A includes the extensions listed in the `PHP.INI` file as shown in the following example. There are two ways to load a PHP extension: using the `dl()` function and using the `PHP.INI` file. These methods are described as follows:

- The `dl()` function allows the loading of extensions within a PHP script if the extension resides in the default `extension_dir` directory, which is defined as `PHP_ROOT:[EXTENSIONS]` (which corresponds to `APACHE$COMMON:[PHP.EXTENSIONS]`).
- The `PHP.INI` file provided with the CSWS_PHP kit resides in the `PHP_ROOT:[000000]` directory (`APACHE$COMMON:[PHP]`). `PHP.INI` contains the extension statement to automatically load the extension for every PHP script executed. To enable the loading of the extension for every PHP script, uncomment the relevant "`extension=`" statement as shown in the following example, and restart the Secure Web Server.

```
; PHP.INI
; Uncomment for the automatic loading of extensions
;
;extension=php_bcmath.exe
;extension=php_bzip2.exe
;extension=php_calendar.exe
;extension=php_ctype.exe
;extension=php_dba.exe
;extension=php_exif.exe
;extension=php_ftp.exe
;extension=php_gd.exe
;extension=php_iconv.exe
;extension=php_ldap.exe
;extension=php_mhash.exe
;extension=php_mysql.exe
;extension=php_oci8.exe
;extension=php_odbc.exe
;extension=php_openssl.exe
;extension=php_openvms.exe
;extension=php_pcre.exe
;extension=php_posix.exe
;extension=php_session.exe
;extension=php_sockets.exe
;extension=php_xml.exe
;extension=php_zip.exe
;extension=php_zlib.exe
```

ODBC Extension

The ODBC extension works with any ODBC Version 2.5 capable server. The `ODBC.PHP` script works with the Attunity Connect "On Platform" Package for OpenVMS using the RMS demo.

Add the following lines to the beginning of the `APACHE$COMMON:[000000]LOGIN.COM` file to enable the script to work with Attunity Connect:

```
$ !
$ ! Run the Attunity login if we find it
$ !
$ IF F$SEARCH ("NAVROOT:[BIN]NAV_LOGIN.COM") .NES. ""
$ THEN
$ @NAVROOT:[BIN]NAV_LOGIN.COM
$ DEFINE APACHE$ODBC_SHR ODNAVSHR
$ DEFINE APACHE$ODBC_PFX NV
$ ENDIF
```

The two logical names required to make the ODBC extension functional are `APACHE$ODBC_SHR` and `APACHE$ODBC_PFX`. These logical names are defined as follows:

- `APACHE$ODBC_SHR`

  This logical name defines the ODBC shareable image to be used for the ODBC access.

- `APACHE$ODBC_PFX`

  This logical name defines, if needed, any ODBC API prefix.

OCI Extension

Add the following lines to the beginning of the file `APACHE$COMMON:[000000]LOGIN.COM`:

- For Oracle 8i:

  ```
  $ !
  $ ! Define the OCI extension logicals if we find the OCI client shareable
  $ !
  ```

```
$ IF F$SEARCH ("ORA_ROOT:[UTIL]ORACLIENT_V817.EXE") .NES. ""
$ THEN
$ DEFINE APACHE$OCI_SHR ORA_ROOT:[UTIL]ORACLIENT_V817.EXE
$ ENDIF
```

- For Oracle 9i:

```
$ !
$ ! Define the OCI extension logicals if we find the OCI client shareable
$ !
$ IF F$SEARCH ("ORA_ROOT:[LIB32]LIBCLNTSH.SO") .NES. ""
$ THEN
$ DEFINE APACHE$OCI_SHR ORA_ROOT:[LIB32]LIBCLNTSH.SO
$ ENDIF
```

The two logical names required to make the OCI extension functional are `APACHE$OCI_SHR` and `APACHE$OCI_PFX`. These logical names are defined as follows:

- `APACHE$OCI_SHR`

  This logical defines the OCI8 shareable image to be used for the OCI8 access.

- `APACHE$OCI_PFX`

  This logical defines, if needed, any OCI8 API prefix.

Sample PHP Scripts

The following PHP sample scripts are included in the CSWS_PHP Version 2.2-1 kit (calendar.php, info.php, odbc.php, and php_openvms.php). These scripts demonstrate the use of the provided extensions.

PHP_CALENDAR.PHP (Useful for Hebrew calendar in Unicode)

```
<?php
#
# Load the calendar extension if needed
#
if (! extension_loaded ("calendar"))
   dl ("php_calendar");
#
# Display the header
#
echo " Testing the Calendar extension<br>\n";
#
# Test the calendar functions
#
$m = date("m", time());
$d = date("d", time());
$y = date("Y", time());

$jd = GregorianToJD($m,$d,$y);

echo "Gregorian month (abbr.): " . jdmonthname($jd, 0) . "<br>\n";
echo "Gregorian month: " . jdmonthname($jd, 1) . "<br>\n";
echo "Julian month (abbr.): " . jdmonthname($jd, 2) . "<br>\n";
echo "Julian month: " . jdmonthname($jd, 3) . "<br>\n";
echo "Jewish month: " . jdmonthname($jd, 4) . "<br>\n";

$y = 1800;
$jd = GregorianToJD($m,$d,$y);
echo "French month: " . jdmonthname($jd, 5) . "<br>\n";
?>
```

PHP_INFO.PHP

```
<?php
#
# Display the header
#
echo " Testing the PHPINFO () function<br>\n";
#
# Test the PHPINFO () function
#
phpinfo (INFO_ALL);
?>
```

PHP_ODBC.PHP

```
<?php
#
# Load the ODBC extension if needed
#
if (! extension_loaded ("odbc"))
   dl ("odbc");
#
# Display the header
#
echo " Testing the ODBC extension<br>\n";
#
# Test the ODBC functions
#
$ctx = odbc_connect ("NAVDEMO", "", "");
$cur = odbc_exec ($ctx, "select c_custkey, c_name from customer");
odbc_result_all ($cur, "border=1 align='center'");
$rc = odbc_free_result ($cur);
odbc_close ($ctx);
?>
```

PHP_OPENVMS.PHP

```
<?php
#
# Load the OpenVMS extension if needed
#
if (! extension_loaded ("openvms"))
    dl ("php_openvms");

#
# Display the header
#
echo "<h1><center>Testing the OpenVMS extension</center></h1><br>\n";

#
# Allow only errors to be reported
#
error_reporting (E_ERROR);

#
# Test the OpenVMS convert filename function
#
# openvms_cvt_filename (func_code, file_name)
#
# func_codes:
#       OPENVMS_CVT_VMS_TO_UNIX         Convert vms filespec to unix filespec
#       OPENVMS_CVT_UNIX_TO_VMS         Convert unix filespec to vms filespec
#
$VmsFn = "PHP_ROOT:[SCRIPTS]PHP_OPENVMS.PHP";
$UnixFn = openvms_cvt_filename (OPENVMS_CVT_VMS_TO_UNIX, $VmsFn);
if ($UnixFn === FALSE)
    echo "openvms_cvt_filename (OPENVMS_CVT_VMS_TO_UNIX, \"$VmsFn\") = <b>" . openvms_message
(openvms_status ()) . "</b><br>\n";
else
    echo "openvms_cvt_filename (OPENVMS_CVT_VMS_TO_UNIX, \"$VmsFn\") = $UnixFn<br>\n";
```

```
#
# Test the OpenVMS getdvi function
#
# openvms_getdvi (item_code [,device_name])
#
# item_codes:
#       <item_code>                     Any Item code supported by F$GETDVI
#       "?"                             List of supported item codes
# device_name:                          Defaults to "TT"
#
$item = "DISPLAY_DEVNAM";
$val = openvms_getdvi ($item);
if ($val === FALSE)
    echo "openvms_getdvi (\"$item\") = <b>" . openvms_message (openvms_status ()) .
"</b><br>\n";
else
    echo "openvms_getdvi (\"$item\") = $val<br>\n";

#
# Test the OpenVMS getjpi function
#
# openvms_getjpi (item_code [,proc_name][,pid])
#
# item_codes:
#       <item_code>                     Any Item code supported by F$GETJPI
#       "?"                             List of supported item codes
# proc_name:                            Any process name
# pid:                                  Any process ID or -1 wild card
#
$item = "LAST_LOGIN_I";
$val = openvms_getjpi ($item);
if ($val === FALSE)
    echo "openvms_getjpi (\"$item\") = <b>" . openvms_message (openvms_status ()) .
"</b><br>\n";
else
    echo "openvms_getjpi (\"$item\") = $val<br>\n";

#
# Test the OpenVMS getsyi function
#
# openvms_getsyi (item_code [,node_name][,csid])
#
# item_codes:
#       <item_code>                     Any Item code supported by F$GETSYI
#       "?"                             List of supported item codes
# node_name:                            Any node name
# csid:                                 Any cluster system ID or -1 wild card
#
$item = "BOOTTIME";
$val = openvms_getsyi ($item, "", 0);
if ($val === FALSE)
    echo "openvms_getsyi (\"$item\") = <b>" . openvms_message (openvms_status ()) .
"</b><br>\n";
else
    echo "openvms_getsyi (\"$item\") = $val<br>\n";

#
# Test the OpenVMS time function
#
# openvms_time ([millisecond_time])
#
$val = openvms_time ();
if ($val === FALSE)
    echo "openvms_time () = <b>" . openvms_message (openvms_status ()) . "</b><br>\n";
else
    echo "openvms_time () = $val<br>\n";

#
# Test the OpenVMS uptime function
#
# openvms_uptime ()
#
$uptime = openvms_uptime ();
if ($uptime === FALSE)
    echo "openvms_uptime () = <b>" . openvms_message (openvms_status ()) . "</b><br>\n";
else
    echo "openvms_uptime () = $uptime<br>\n";
```

```
echo "<br>\n";

#
# Show the cluster info
#
ShowCluster ();

#
# Show the system info
#
ShowSystem ();

#
# Show Cluster
#
function ShowCluster ()
{
$SystemId = openvms_getsyi ("SCSSYSTEMID");
$NodeName = openvms_getsyi ("NODENAME");
$Time = strtok (openvms_time (), ".");

echo "<pre>\n";
$hdr = "View of Cluster from system ID $SystemId  node: $NodeName";
$pad = str_repeat (" ", 79 - (strlen ($hdr) + strlen ($Time)));
echo $hdr . $pad . $Time . "\n";
echo "+-----------------------------+\n";
echo "|      SYSTEMS      | MEMBERS |\n";
echo "|-------------------+---------|\n";
echo "|  NODE  | SOFTWARE |  STATUS |\n";
echo "|--------+----------+---------|\n";

$ctx = -1;
while (1)
    {
    $csid = openvms_getsyi ("NODE_CSID", "", &$ctx);
    if ($csid === FALSE)
        {
        $status = openvms_status ();
        if ($status != 2560)
            echo openvms_message (openvms_status ()) . "<br>\n";
        break;
        }
    $NodeName = str_pad (openvms_getsyi ("NODENAME", "", $csid), 6, " ", STR_PAD_RIGHT);
    $swtype = openvms_getsyi ("NODE_SWTYPE", "", $csid);
    $swvers = openvms_getsyi ("NODE_SWVERS", "", $csid);
    $software = str_pad ($swtype . $swvers, 8, " ", STR_PAD_RIGHT);
    if (strcasecmp (openvms_getsyi ("CLUSTER_MEMBER", "", $csid), "TRUE") == 0)
        $status =  "MEMBER";
    else
        $status = "      ";
    echo "| $NodeName | $software | $status  |\n";
    }

if (openvms_getsyi ("CLUSTER_NODES") == 0)
    {
    $NodeName = str_pad (openvms_getsyi ("NODENAME"), 6, " ", STR_PAD_RIGHT);
    $swtype = openvms_getsyi ("NODE_SWTYPE", "", $csid);
    $swvers = openvms_getsyi ("NODE_SWVERS", "", $csid);
    $software = str_pad ($swtype . $swvers, 8, " ", STR_PAD_RIGHT);
    if (strcasecmp (openvms_getsyi ("CLUSTER_MEMBER", "", $csid), "TRUE") == 0)
        $status =  "MEMBER";
    else
        $status = "      ";
    echo "| $NodeName | $software | $status  |\n";
    }

echo "+-----------------------------+\n";
echo "</pre>\n";
}

#
# Show System (Requires World Privilege)
#
function ShowSystem ()
{
$VmsVer = trim (openvms_getsyi ("VERSION"));
```

```
$NodeName = openvms_getsyi ("NODENAME");
$UpTime = trim (openvms_uptime ());
$Time = openvms_time ();

echo "<pre>\n";
echo "OpenVMS $VmsVer  on node $NodeName  $Time  Uptime  $UpTime\n";
echo "  Pid    Process Name    State  Pri      I/O       CPU       Page flts  Pages\n";
$ctx = -1;
while (1)
    {
    $pid = openvms_getjpi ("PID", "", &$ctx);
    if ($pid === FALSE)
        {
        $status = openvms_status ();
        if ($status != 2472)
            echo openvms_message (openvms_status ()) . "<br>\n";
        break;
        }
    $prcpid = str_pad ($pid, 8, " ", STR_PAD_RIGHT);
    $prcnam = str_pad (openvms_getjpi ("PRCNAM", "", $pid), 15, " ", STR_PAD_RIGHT);
    $state = str_pad (openvms_getjpi ("STATE", "", $pid), 5, " ", STR_PAD_RIGHT);
    $pri = str_pad (openvms_getjpi ("PRI", "", $pid), 3, " ", STR_PAD_LEFT);
    $io = openvms_getjpi ("DIRIO", "", $pid) + openvms_getjpi ("BUFIO", "", $pid);
    $io = str_pad ($io, 9, " ", STR_PAD_LEFT);
    $cputim = openvms_time (openvms_getjpi ("CPUTIM", "", $pid));
    $pagflts = str_pad (openvms_getjpi ("PAGEFLTS", "", $pid), 9, " ", STR_PAD_LEFT);
    $pages = openvms_getjpi ("GPGCNT", "", $pid) + openvms_getjpi ("PPGCNT", "", $pid);
    $pages = $pages / (openvms_getsyi ("PAGE_SIZE") / 512);
    $pages = str_pad ($pages, 6, " ", STR_PAD_LEFT);
    $multithread = openvms_getjpi ("MULTITHREAD", "", $pid);
    $owner = openvms_getjpi ("OWNER", "", $pid);
    $mode = openvms_getjpi ("MODE", "", $pid);
    if ($multithread >= 1)
        $sts = "M";
    else
        $sts = " ";
    if ($owner != 0)
        $sts .= "S";
    else
    if (strcasecmp ($mode, "NETWORK") == 0)
        $sts .= "N";
    else
    if (strcasecmp ($mode, "BATCH") == 0)
        $sts .= "B";
    else
        $sts .= " ";
    echo "$prcpid $prcnam $state  $pri$io$cputim $pagflts $pages $sts\n";
    }

echo "</pre>\n";
}
?>
```

Release Notes

Problems Fixed

The following problems are fixed in this release:

- When you attempt to create a PHP session, CSWS_PHP displays the following warning message:

  ```
  Warning: session_start() [function.session-start]:
  open(SYS$SCRATCH:/sess_856075c3a77b849060f8614a36cddf87, O_RDWR) failed: no such
  file or directory (2) in /pubdata/dailymission/index.php
  ```

  This problem is fixed in this release.

- While calling the PHP LDAP routines from a PHP script, PHP displays the following error message:

  ```
  Fatal error: Call to undefined function ldap_search()
  ```

  This problem is fixed in this release.

- The "SYSTEM-S-NORMAL" message is set by the openvms_cvt_filename() function if the directory name passed to it does not exist.

  This problem is fixed in this release.

General Information

This section contains notes on the current release of CSWS_PHP.

- Add logical names to PHP_SETUP.COM for extended file name support

  If you are using CSWS_PHP standalone or with the Secure Web Server, and you have ODS-5 files with extended file names, the files will not be processed correctly.

  To solve this problem, add the following logical definitions to the end of PHP_SETUP.COM located in APACHE$COMMON:[000000]:

```
$ DEFINE /NOLOG DECC$EFS_CASE_PRESERVE ENABLED
$ DEFINE /NOLOG DECC$EFS_CASE_SPECIAL ENABLED
$ DEFINE /NOLOG DECC$EFS_CHARSET ENABLED
$ DEFINE /NOLOG DECC$FILE_SHARING ENABLED
```

  Note that using the following commands do not work in this situation:

  - SET PROCESS/PARSE=EXTENDED
  - SET PROCESS/PARSE=EXTENDED/CASE=SENSITIVE

- Configuring CSWS_PHP not required

  During the installation, the file PHP_SETUP.COM is added to the APACHE$ROOT directory, and an include file for MOD_PHP.CONF is added to the end of HTTPD.CONF. When you start the Secure Web Server, PHP_SETUP.COM is run and PHP is loaded into the server. You do not need to configure CSWS_PHP.

- PHP DNS functions supported only with HP TCP/IP Services for OpenVMS

  This version of CSWS_PHP supports the CHECKDNSRR and GETMXRR functions only on systems using HP TCP/IP Services for OpenVMS. These functions might be supported with other TCP/IP products in a future CSWS_PHP kit.

- PHP LINK functions not supported

  This version of CSWS_PHP does not support the LINK, LINKINFO, SYMLINK, and READLINK functions. These functions might be supported in a future CSWS_PHP kit.