Installing PHP

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Table of Contents

Preface

1. General Installation Considerations

2. Installation on Windows systems

Windows Installer

Manual Installation Steps

ActiveScript

Microsoft IIS / PWS

Apache 1.3.x on Microsoft Windows

Apache 2.0.x on Microsoft Windows

Sun, iPlanet and Netscape servers on Microsoft Windows

OmniHTTPd Server

Sambar Server on Microsoft Windows

Xitami on Microsoft Windows

Installation of extensions on Windows

3. Installation of PECL extensions

Introduction to PECL Installations

Downloading PECL extensions

PECL for Windows users

Compiling shared PECL extensions with the pecl command

Compiling shared PECL extensions with phpize

Compiling PECL extensions statically into PHP

4. Problems?

Read the FAQ

Other problems

Bug reports

5. Runtime Configuration

The configuration file

How to change configuration settings

6. Installation FAQ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Preface

These installation instructions were generated from the HTML version of

the PHP Manual so formatting and linking have been altered. See the

online and updated version at: http://php.net/install.windows

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Chapter 1. General Installation Considerations

Before starting the installation, first you need to know what do you

want to use PHP for. There are three main fields you can use PHP, as

described in the What can PHP do? section:

\* Websites and web applications (server-side scripting)

\* Command line scripting

\* Desktop (GUI) applications

For the first and most common form, you need three things: PHP itself,

a web server and a web browser. You probably already have a web

browser, and depending on your operating system setup, you may also

have a web server (e.g. Apache on Linux and MacOS X; IIS on Windows).

You may also rent webspace at a company. This way, you don't need to

set up anything on your own, only write your PHP scripts, upload it to

the server you rent, and see the results in your browser.

In case of setting up the server and PHP on your own, you have two

choices for the method of connecting PHP to the server. For many

servers PHP has a direct module interface (also called SAPI). These

servers include Apache, Microsoft Internet Information Server, Netscape

and iPlanet servers. Many other servers have support for ISAPI, the

Microsoft module interface (OmniHTTPd for example). If PHP has no

module support for your web server, you can always use it as a CGI or

FastCGI processor. This means you set up your server to use the CGI

executable of PHP to process all PHP file requests on the server.

If you are also interested to use PHP for command line scripting (e.g.

write scripts autogenerating some images for you offline, or processing

text files depending on some arguments you pass to them), you always

need the command line executable. For more information, read the

section about writing command line PHP applications. In this case, you

need no server and no browser.

With PHP you can also write desktop GUI applications using the PHP-GTK

extension. This is a completely different approach than writing web

pages, as you do not output any HTML, but manage Windows and objects

within them. For more information about PHP-GTK, please visit the site

dedicated to this extension. PHP-GTK is not included in the official

PHP distribution.

From now on, this section deals with setting up PHP for web servers on

Unix and Windows with server module interfaces and CGI executables. You

will also find information on the command line executable in the

following sections.

PHP source code and binary distributions for Windows can be found at

http://www.php.net/downloads.php. We recommend you to choose a mirror

nearest to you for downloading the distributions.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Chapter 2. Installation on Windows systems

This section applies to Windows 98/Me and Windows NT/2000/XP/2003. PHP

will not work on 16 bit platforms such as Windows 3.1 and sometimes we

refer to the supported Windows platforms as Win32. Windows 95 is no

longer supported as of PHP 4.3.0.

There are two main ways to install PHP for Windows: either manually or

by using the installer.

If you have Microsoft Visual Studio, you can also build PHP from the

original source code.

Once you have PHP installed on your Windows system, you may also want

to load various extensions for added functionality.

Warning

There are several all-in-one installers over the Internet, but none of

those are endorsed by PHP.net, as we believe that the manual

installation is the best choice to have your system secure and

optimised.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Windows Installer (PHP 5.2 and later)

The Windows PHP installer for later versions of PHP is built using MSI

technology using the Wix Toolkit (http://wix.sourceforge.net/). It will

install and configure PHP and all the built-in and PECL extensions, as

well as configure many of the popular web servers such as IIS, Apache,

and Xitami.

First, install your selected HTTP (web) server on your system, and make

sure that it works. Then proceed with one of the following install

types.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Normal Install

Run the MSI installer and follow the instructions provided by the

installation wizard. You will be prompted to select the Web Server you

wish to configure first, along with any configuration details needed.

You will then be prompted to select which features and extensions you

wish to install and enable. By selecting "Will be installed on local

hard drive" in the drop-down menu for each item you can trigger whether

to install the feature or not. By selecting "Entire feature will be

installed on local hard drive", you will be able to install all

sub-features of the included feature ( for example by selecting this

options for the feature "PDO" you will install all PDO Drivers ).

Warning

It is not recommended to install all extensions by default, since many

other them require dependencies from outside PHP in order to function

properly. Instead, use the Installation Repair Mode that can be

triggered thru the 'Add/Remove Programs' control panel to enable or

disable extensions and features after installation.

The installer then sets up PHP to be used in Windows and the php.ini

file, and configures certain web servers to use PHP. The installer will

currently configure IIS (CGI mode only), Apache, Xitami, and Sambar

Server; if you are using a different web server you'll need to

configure it manually.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Silent Install

The installer also supports a silent mode, which is helpful for Systems

Administrators to deploy PHP easily. To use silent mode:

msiexec.exe /i php-VERSION-win32-install.msi /q

You can control the install directory by passing it as a parameter to

the install. For example, to install to e:\php:

msiexec.exe /i php-VERSION-win32-install.msi /q INSTALLDIR=e:\php

You can also use the same syntax to specify the Apache Configuration

Directory (APACHEDIR), the Sambar Server directory (SAMBARDIR), and the

Xitami Server directory (XITAMIDIR).

You can also specify what features to install. For example, to install

the mysqli extension and the CGI executable:

msiexec.exe /i php-VERSION-win32-install.msi /q ADDLOCAL=cgi,ext\_php\_mysqli

The current list of Features to install is as follows:

MainExecutable - php.exe executable

ScriptExecutable - php-win.exe executable

ext\_php\_\* - the various extensions ( for example: ext\_php\_mysql for MySQL )

apache13 - Apache 1.3 module

apache20 - Apache 2.0 module

apache22 - Apache 2,2 module

apacheCGI - Apache CGI executable

iis4ISAPI - IIS ISAPI module

iis4CGI - IIS CGI executable

NSAPI - Sun/iPlanet/Netscape server module

Xitami - Xitami CGI executable

Sambar - Sambar Server ISAPI module

CGI - php-cgi.exe executable

PEAR - PEAR installer

Manual - PHP Manual in CHM Format

For more information on installing MSI installers from the command

line, visit

http://msdn.microsoft.com/library/en-us/msi/setup/command\_line\_options.

asp

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Windows Installer (PHP 5.1.0 and earlier)

The Windows PHP installer is available from the downloads page at

http://www.php.net/downloads.php. This installs the CGI version of PHP

and for IIS, PWS, and Xitami, it configures the web server as well. The

installer does not include any extra external PHP extensions

(php\_\*.dll) as you'll only find those in the Windows Zip Package and

PECL downloads.

Note: While the Windows installer is an easy way to make PHP work,

it is restricted in many aspects as, for example, the automatic

setup of extensions is not supported. Use of the installer isn't the

preferred method for installing PHP.

First, install your selected HTTP (web) server on your system, and make

sure that it works.

Run the executable installer and follow the instructions provided by

the installation wizard. Two types of installation are supported -

standard, which provides sensible defaults for all the settings it can,

and advanced, which asks questions as it goes along.

The installation wizard gathers enough information to set up the

php.ini file, and configure certain web servers to use PHP. One of the

web servers the PHP installer does not configure for is Apache, so

you'll need to configure it manually.

Once the installation has completed, the installer will inform you if

you need to restart your system, restart the server, or just start

using PHP.

Warning

Be aware, that this setup of PHP is not secure. If you would like to

have a secure PHP setup, you'd better go on the manual way, and set

every option carefully. This automatically working setup gives you an

instantly working PHP installation, but it is not meant to be used on

online servers.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Manual Installation Steps

This install guide will help you manually install and configure PHP

with a web server on Microsoft Windows. To get started you'll need to

download the zip binary distribution from the downloads page at

http://www.php.net/downloads.php.

Although there are many all-in-one installation kits, and we also

distribute a PHP installer for Microsoft Windows, we recommend you take

the time to setup PHP yourself as this will provide you with a better

understanding of the system, and enables you to install PHP extensions

easily when needed.

Upgrading from a previous PHP version: Previous editions of the

manual suggest moving various ini and DLL files into your SYSTEM

(i.e. C:\WINDOWS) folder and while this simplifies the installation

procedure it makes upgrading difficult. We advise you remove all of

these files (like php.ini and PHP related DLLs from the Windows

SYSTEM folder) before moving on with a new PHP installation. Be sure

to backup these files as you might break the entire system. The old

php.ini might be useful in setting up the new PHP as well. And as

you'll soon learn, the preferred method for installing PHP is to

keep all PHP related files in one directory and have this directory

available to your systems PATH.

MDAC requirements: If you use Microsoft Windows 98/NT4 download the

latest version of the Microsoft Data Access Components (MDAC) for

your platform. MDAC is available at http://msdn.microsoft.com/data/.

This requirement exists because ODBC is built into the distributed

Windows binaries.

The following steps should be completed on all installations before any

server specific instructions are performed:

Extract the distribution file into a directory of your choice. If you

are installing PHP 4, extract to C:\, as the zip file expands to a

foldername like php-4.3.7-Win32. If you are installing PHP 5, extract

to C:\php as the zip file doesn't expand as in PHP 4. You may choose a

different location but do not have spaces in the path (like C:\Program

Files\PHP) as some web servers will crash if you do.

The directory structure extracted from the zip is different for PHP

versions 4 and 5 and look like as follows:

Example 2-2. PHP 5 package structure

c:\php

|

+--dev

| |

| |-php5ts.lib

|

+--ext -- extension DLLs for PHP

| |

| |-php\_bz2.dll

| |

| |-php\_cpdf.dll

| |

| |-..

|

+--extras

| |

| +--mibs -- support files for SNMP

| |

| +--openssl -- support files for Openssl

| |

| +--pdf-related -- support files for PDF

| |

| |-mime.magic

|

+--pear -- initial copy of PEAR

|

|

|-go-pear.bat -- PEAR setup script

|

|-fdftk.dll

|

|-..

|

|-php-cgi.exe -- CGI executable

|

|-php-win.exe -- executes scripts without an opened command prompt

|

|-php.exe -- CLI executable - ONLY for command line scripting

|

|-..

|

|-php.ini-development -- development php.ini settings

|

|-php.ini-production -- recommended php.ini settings for production

|

|-php5activescript.dll

|

|-php5apache.dll

|

|-php5apache2.dll

|

|-..

|

|-php5ts.dll -- core PHP DLL

|

|-...

Notice the differences and similarities. Both PHP 4 and PHP 5 have a

CGI executable, a CLI executable, and server modules, but they are

located in different folders and/or have different names. While PHP 4

packages have the server modules in the sapi folder, PHP 5

distributions have no such directory and instead they're in the PHP

folder root. The supporting DLLs for the PHP 5 extensions are also not

in a seperate directory.

Note: In PHP 4, you should move all files located in the dll and

sapi folders to the main folder (e.g. C:\php).

Here is a list of server modules shipped with PHP 5:

\* sapi/php5apache2\_2.dll - Apache 2.2.x module.

\* sapi/php5apache.dll (php5apache.dll) - Apache 1.x module

\* sapi/php5apache2.dll (php5apache2.dll) - - Apache 2.0.x module.

\* sapi/php5isapi.dll - ISAPI Module for ISAPI compliant web servers

like IIS 5.0 or newer. However the FCGI SAPI is recommended with

IIS

\* sapi/php5nsapi.dll (php5nsapi.dll) - Sun/iPlanet/Netscape server

module.

Server modules provide significantly better performance and additional

functionality compared to the CGI binary. The FastCGI is significantly

more stable and can be faster than the ISAPI module with IIS.

The CLI version is designed to let you use PHP for command line

scripting. More information about CLI is available in the chapter

about using PHP from the command line.

Warning

The SAPI modules have been significantly improved as of the 4.1

release, however, in older systems you may encounter server errors or

other server modules failing, such as ASP.

The CGI and CLI binaries, and the web server modules all require the

php5ts.dll file to be available to them. You have to make

sure that this file can be found by your PHP installation. The search

order for this DLL is as follows:

\* The same directory from where php.exe is called, or in case you use

a SAPI module, the web server's directory (e.g. C:\Program

Files\Apache Group\Apache2\bin).

\* Any directory in your Windows PATH environment variable.

To make php5ts.dll available you have three options: copy

the file to the Windows system directory, copy the file to the web

server's directory, or add your PHP directory, C:\php to the PATH. For

better maintenance, we advise you to follow the last option, add C:\php

to the PATH, because it will be simpler to upgrade PHP in the future.

Read more about how to add your PHP directory to PATH in the

corresponding FAQ entry (and then don't forget to restart the computer

- logoff isn't enough).

The next step is to set up a valid configuration file for PHP, php.ini.

There are two ini files distributed in the zip file, php.ini-development

and php.ini-production. We advise you to use php.ini-production,

because we optimized the default settings in this file for performance,

and security. Read this well documented file carefully because it has

changes from php.ini-production that will drastically affect your setup.

Some examples are display\_errors being off and magic\_quotes\_gpc being off.

In addition to reading these, study the ini settings and set every

element manually yourself. If you would like to achieve the best

security, then this is the way for you, although PHP works fine with

these default ini files. Copy your chosen ini-file to a directory that

PHP is able to find and rename it to php.ini. PHP searches for php.ini

in the locations described in the Section called The configuration file

in Chapter 5 section.

If you are running Apache 2, the simpler option is to use the PHPIniDir

directive (read the installation on Apache 2 page), otherwise your best

option is to set the PHPRC environment variable. This process is

explained in the following FAQ entry.

Note: If you're using NTFS on Windows NT, 2000, XP or 2003, make

sure that the user running the web server has read permissions to

your php.ini (e.g. make it readable by Everyone).

The following steps are optional:

\* Edit your new php.ini file. If you plan to use OmniHTTPd, do not

follow the next step. Set the doc\_root to point to your web servers

document\_root. For example:

doc\_root = c:\inetpub\wwwroot // for IIS/PWS

doc\_root = c:\apache\htdocs // for Apache

\* Choose the extensions you would like to load when PHP starts. See

the section about Windows extensions, about how to set up one, and

what is already built in. Note that on a new installation it is

advisable to first get PHP working and tested without any

extensions before enabling them in php.ini.

\* On PWS and IIS, you can set the browscap configuration setting to

point to: c:\windows\system\inetsrv\browscap.ini on Windows 9x/Me,

c:\winnt\system32\inetsrv\browscap.ini on NT/2000, and

c:\windows\system32\inetsrv\browscap.ini on XP. For an up-to-date

browscap.ini, read the following FAQ.

PHP is now setup on your system. The next step is to choose a web

server, and enable it to run PHP. Choose a web server from the table of

contents.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ActiveScript

This section contains notes specific to the ActiveScript installation.

ActiveScript is a Windows only SAPI that enables you to use PHP script

in any ActiveScript compliant host, like Windows Script Host,

ASP/ASP.NET, Windows Script Components or Microsoft Scriptlet control.

As of PHP 5.0.1, ActiveScript has been moved to the PECL repository.

The DLL for this PECL extension may be downloaded from either the PHP

Downloads page or from http://pecl4win.php.net/

Note: You should read the manual installation steps first!

After installing PHP, you should download the ActiveScript DLL

(php5activescript.dll) and place it in the main PHP folder (e.g.

C:\php).

After having all the files needed, you must register the DLL on your

system. To achieve this, open a Command Prompt window (located in the

Start Menu). Then go to your PHP directory by typing something like cd

C:\php. To register the DLL just type regsvr32 php5activescript.dll.

To test if ActiveScript is working, create a new file, named test.wsf

(the extension is very important) and type:

<job id="test">

<script language="PHPScript">

$WScript->Echo("Hello World!");

</script>

</job>

Save and double-click on the file. If you receive a little window

saying "Hello World!" you're done.

Note: In PHP 4, the engine was named 'ActivePHP', so if you are

using PHP 4, you should replace 'PHPScript' with 'ActivePHP' in the

above example.

Note: ActiveScript doesn't use the default php.ini file. Instead, it

will look only in the same directory as the .exe that caused it to

load. You should create php-activescript.ini and place it in that

folder, if you wish to load extensions, etc.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Microsoft IIS / PWS

This section contains notes and hints specific to IIS (Microsoft

Internet Information Server).

Warning

By using the CGI setup, your server is open to several possible

attacks. Please read our CGI security section to learn how to defend

yourself from those attacks.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

General considerations for all installations of PHP with IIS or PWS

\* First, read the Manual Installation Instructions. Do not skip this

step as it provides crucial information for installing PHP on

Windows.

\* CGI users must set the cgi.force\_redirect PHP directive to 0 inside

php.ini. Read the faq on cgi.force\_redirect for important details.

Also, CGI users may want to set the cgi.redirect\_status\_env

directive. When using directives, be sure these directives aren't

commented out inside php.ini.

\* The PHP 4 CGI is named php.exe while in PHP 5 it's php-cgi.exe. In

PHP 5, php.exe is the CLI, and not the CGI.

\* Modify the Windows PATH environment variable to include the PHP

directory. This way the PHP DLL files and PHP executables can all

remain in the PHP directory without cluttering up the Windows

system directory. For more details, see the FAQ on Setting the

PATH.

\* The IIS user (usually IUSR\_MACHINENAME) needs permission to read

various files and directories, such as php.ini, docroot, and the

session tmp directory.

\* Be sure the extension\_dir and doc\_root PHP directives are

appropriately set in php.ini. These directives depend on the system

that PHP is being installed on. In PHP 4, the extension\_dir is

extensions while with PHP 5 it's ext. So, an example PHP 5

extensions\_dir value is "c:\php\ext" and an example IIS doc\_root

value is "c:\Inetpub\wwwroot".

\* PHP extension DLL files, such as php\_mysql.dll and php\_curl.dll,

are found in the zip package of the PHP download (not the PHP

installer). In PHP 5, many extensions are part of PECL and can be

downloaded in the "Collection of PECL modules" package. Files such

as php\_zip.dll and php\_ssh2.dll. Download PHP files here.

\* When defining the executable, the 'check that file exists' box may

also be checked. For a small performance penalty, the IIS (or PWS)

will check that the script file exists and sort out authentication

before firing up PHP. This means that the web server will provide

sensible 404 style error messages instead of CGI errors complaining

that PHP did not output any data.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Windows NT/200x/XP and IIS 4 or newer

PHP may be installed as a CGI binary, or with the ISAPI module. In

either case, you need to start the Microsoft Management Console (may

appear as 'Internet Services Manager', either in your Windows NT 4.0

Option Pack branch or the Control Panel=>Administrative Tools under

Windows 2000/XP). Then right click on your Web server node (this will

most probably appear as 'Default Web Server'), and select 'Properties'.

If you want to use the CGI binary, do the following:

\* Under 'Home Directory', 'Virtual Directory', or 'Directory', do the

following:

\* Change the Execute Permissions to 'Scripts only'

\* Click on the 'Configuration' button, and choose the Application

Mappings tab. Click Add and set the Executable path to the

appropriate CGI file. An example PHP 5 value is: C:\php\php-cgi.exe

Supply .php as the extension. Leave 'Method exclusions' blank, and

check the 'Script engine' checkbox. Now, click OK a few times.

\* Set up the appropriate security. (This is done in Internet Service

Manager), and if your NT Server uses NTFS file system, add execute

rights for I\_USR\_ to the directory that contains php.exe /

php-cgi.exe.

To use the ISAPI module, do the following:

\* If you don't want to perform HTTP Authentication using PHP, you can

(and should) skip this step. Under ISAPI Filters, add a new ISAPI

filter. Use PHP as the filter name, and supply a path to the

php5isapi.dll.

\* Under 'Home Directory', 'Virtual Directory', or 'Directory', do the

following:

\* Change the Execute Permissions to 'Scripts only'

\* Click on the 'Configuration' button, and choose the Application

Mappings tab. Click Add and set the Executable path to the

appropriate ISAPI DLL. An example PHP 5 value is:

C:\php\php5isapi.dll Supply .php as the extension. Leave 'Method

exclusions' blank, and check the 'Script engine' checkbox. Now,

click OK a few times.

\* Stop IIS completely (NET STOP iisadmin)

\* Start IIS again (NET START w3svc)

With IIS 6 (2003 Server), open up the IIS Manager, go to Web Service

Extensions, choose "Add a new Web service extension", enter in a name

such as PHP, choose the Add button and for the value browse to either

the ISAPI file (php5isapi.dll) or CGI (php.exe or

php-cgi.exe) then check "Set extension status to Allowed" and click OK.

In order to use index.php as a default content page, do the following:

From within the Documents tab, choose Add. Type in index.php and click

OK. Adjust the order by choosing Move Up or Move Down. This is similar

to setting DirectoryIndex with Apache.

The steps above must be repeated for each extension that is to be

associated with PHP scripts. .php is the most common although .php3 may

be required for legacy applications.

If you experience 100% CPU usage after some time, turn off the IIS

setting Cache ISAPI Application.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Windows and IIS

See http://www.php.net/install.windows

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Apache 1.3.x on Microsoft Windows

This section contains notes and hints specific to Apache 1.3.x installs

of PHP on Microsoft Windows systems. There are also instructions and

notes for Apache 2 on a separate page.

Note: Please read the manual installation steps first!

There are two ways to set up PHP to work with Apache 1.3.x on Windows.

One is to use the CGI binary (php.exe for PHP 4 and php-cgi.exe for PHP

5), the other is to use the Apache Module DLL. In either case you need

to edit your httpd.conf to configure Apache to work with PHP, and then

restart the server.

It is worth noting here that now the SAPI module has been made more

stable under Windows, we recommend it's use above the CGI binary, since

it is more transparent and secure.

Although there can be a few variations of configuring PHP under Apache,

these are simple enough to be used by the newcomer. Please consult the

Apache Documentation for further configuration directives.

After changing the configuration file, remember to restart the server,

for example, NET STOP APACHE followed by NET START APACHE, if you run

Apache as a Windows Service, or use your regular shortcuts.

Note: Remember that when adding path values in the Apache

configuration files on Windows, all backslashes such as

c:\directory\file.ext must be converted to forward slashes, as

c:/directory/file.ext. A trailing slash may also be necessary for

directories.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Installing as an Apache module

You should add the following lines to your Apache httpd.conf file:

Example 2-3. PHP as an Apache 1.3.x module

This assumes PHP is installed to c:\php. Adjust the path if this is not

the case.

For PHP 5:

# Add to the end of the LoadModule section

LoadModule php5\_module "C:/php/php5apache.dll"

# Add to the end of the AddModule section

AddModule mod\_php5.c

For both:

# Add this line inside the <IfModule mod\_mime.c> conditional brace

AddType application/x-httpd-php .php

# For syntax highlighted .phps files, also add

AddType application/x-httpd-php-source .phps

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Installing as a CGI binary

If you unzipped the PHP package to C:\php\ as described in the Manual

Installation Steps section, you need to insert these lines to your

Apache configuration file to set up the CGI binary:

Example 2-4. PHP and Apache 1.3.x as CGI

ScriptAlias /php/ "c:/php/"

AddType application/x-httpd-php .php

# For PHP 4

Action application/x-httpd-php "/php/php.exe"

# For PHP 5

Action application/x-httpd-php "/php/php-cgi.exe"

# specify the directory where php.ini is

SetEnv PHPRC C:/php

Note that the second line in the list above can be found in the actual

versions of httpd.conf, but it is commented out. Remember also to

substitute the c:/php/ for your actual path to PHP.

Warning

By using the CGI setup, your server is open to several possible

attacks. Please read our CGI security section to learn how to defend

yourself from those attacks.

If you would like to present PHP source files syntax highlighted, there

is no such convenient option as with the module version of PHP. If you

chose to configure Apache to use PHP as a CGI binary, you will need to

use the highlight\_file() function. To do this simply create a PHP

script file and add this code: <?php

highlight\_file('some\_php\_script.php'); ?>.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Apache 2.0.x on Microsoft Windows

This section contains notes and hints specific to Apache 2.0.x installs

of PHP on Microsoft Windows systems. We also have instructions and

notes for Apache 1.3.x users on a separate page.

Note: You should read the manual installation steps first!

Apache 2.2.x Support: Users of Apache 2.2.x may use the

documentation below except the appropriate DLL file is named

php5apache2\_2.dll and it only exists as of PHP 5.2.0. See also

http://snaps.php.net/

Warning

We do not recommend using a threaded MPM in production with Apache2.

Use the prefork MPM instead, or use Apache1. For information on why,

read the related FAQ entry on using Apache2 with a threaded MPM

You are highly encouraged to take a look at the Apache Documentation to

get a basic understanding of the Apache 2.0.x Server. Also consider to

read the Windows specific notes for Apache 2.0.x before reading on

here.

PHP and Apache 2.0.x compatibility notes: The following versions of

PHP are known to work with the most recent version of Apache 2.0.x:

\* PHP 4.3.0 or later available at http://www.php.net/downloads.php.

\* the latest stable development version. Get the source code

http://snaps.php.net/php5-latest.tar.gz or download binaries for

Windows http://snaps.php.net/win32/php5-win32-latest.zip.

\* a prerelease version downloadable from http://qa.php.net/.

\* you have always the option to obtain PHP through SVN.

These versions of PHP are compatible to Apache 2.0.40 and later.

Apache 2.0 SAPI-support started with PHP 4.2.0. PHP 4.2.3 works with

Apache 2.0.39, don't use any other version of Apache with PHP 4.2.3.

However, the recommended setup is to use PHP 4.3.0 or later with the

most recent version of Apache2.

All mentioned versions of PHP will work still with Apache 1.3.x.

Warning

Apache 2.0.x is designed to run on Windows NT 4.0, Windows 2000 or

Windows XP. At this time, support for Windows 9x is incomplete. Apache

2.0.x is not expected to work on those platforms at this time.

Download the most recent version of Apache 2.0.x and a fitting PHP

version. Follow the Manual Installation Steps and come back to go on

with the integration of PHP and Apache.

There are two ways to set up PHP to work with Apache 2.0.x on Windows.

One is to use the CGI binary the other is to use the Apache module DLL.

In either case you need to edit your httpd.conf to configure Apache to

work with PHP and then restart the server.

Note: Remember that when adding path values in the Apache

configuration files on Windows, all backslashes such as

c:\directory\file.ext must be converted to forward slashes, as

c:/directory/file.ext. A trailing slash may also be necessary for

directories.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Installing as a CGI binary

You need to insert these three lines to your Apache httpd.conf

configuration file to set up the CGI binary:

Example 2-5. PHP and Apache 2.0 as CGI

ScriptAlias /php/ "c:/php/"

AddType application/x-httpd-php .php

# For PHP 4

Action application/x-httpd-php "/php/php.exe"

# For PHP 5

Action application/x-httpd-php "/php/php-cgi.exe"

Warning

By using the CGI setup, your server is open to several possible

attacks. Please read our CGI security section to learn how to defend

yourself from those attacks.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Installing as an Apache module

You need to insert these two lines to your Apache httpd.conf

configuration file to set up the PHP module for Apache 2.0:

Example 2-6. PHP and Apache 2.0 as Module

# For PHP 5 do something like this:

LoadModule php5\_module "c:/php/php5apache2.dll"

AddType application/x-httpd-php .php

# configure the path to php.ini

PHPIniDir "C:/php"

Note: Remember to substitute your actual path to PHP for the c:/php/

in the above examples. Take care to use either

php5apache2.dll in your LoadModule directive and not php5apache.dll

as the latter ones are designed to run with Apache 1.3.x.

Note: If you want to use content negotiation, read related FAQ.

Warning

Don't mix up your installation with DLL files from different PHP

versions. You have the only choice to use the DLL's and extensions that

ship with your downloaded PHP version.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Sun, iPlanet and Netscape servers on Microsoft Windows

This section contains notes and hints specific to Sun Java System Web

Server, Sun ONE Web Server, iPlanet and Netscape server installs of PHP

on Windows.

From PHP 4.3.3 on you can use PHP scripts with the NSAPI module to

generate custom directory listings and error pages. Additional

functions for Apache compatibility are also available. For support in

current web servers read the note about subrequests.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CGI setup on Sun, iPlanet and Netscape servers

To install PHP as a CGI handler, do the following:

\* Copy php5ts.dll to your systemroot (the directory where you

installed Windows)

\* Make a file association from the command line. Type the following

two lines:

assoc .php=PHPScript

ftype PHPScript=c:\php\php.exe %1 %\*

\* In the Netscape Enterprise Administration Server create a dummy

shellcgi directory and remove it just after (this step creates 5

important lines in obj.conf and allow the web server to handle

shellcgi scripts).

\* In the Netscape Enterprise Administration Server create a new mime

type (Category: type, Content-Type: magnus-internal/shellcgi, File

Suffix:php).

\* Do it for each web server instance you want PHP to run

More details about setting up PHP as a CGI executable can be found

here: http://benoit.noss.free.fr/php/install-php.html

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

NSAPI setup on Sun, iPlanet and Netscape servers

To install PHP with NSAPI, do the following:

\* Copy php5ts.dll to your systemroot (the directory where you

installed Windows)

\* Make a file association from the command line. Type the following

two lines:

assoc .php=PHPScript

ftype PHPScript=c:\php\php.exe %1 %\*

\* In the Netscape Enterprise Administration Server create a new mime

type (Category: type, Content-Type: magnus-internal/x-httpd-php,

File Suffix: php).

\* Edit magnus.conf (for servers >= 6) or obj.conf (for servers < 6)

and add the following: You should place the lines after mime types

init.

Init fn="load-modules" funcs="php5\_init,php5\_execute,php5\_auth\_trans" shlib="c:/

php/sapi/php5nsapi.dll"

Init fn="php5\_init" LateInit="yes" errorString="Failed to initialise PHP!" [php\_

ini="c:/path/to/php.ini"]

The php\_ini parameter is optional but with it you

can place your php.ini in your web server configuration directory.

\* Configure the default object in obj.conf (for virtual server

classes [Sun Web Server 6.0+] in their vserver.obj.conf): In the

<Object name="default"> section, place this line necessarily after

all 'ObjectType' and before all 'AddLog' lines:

Service fn="php5\_execute" type="magnus-internal/x-httpd-php" [inikey=value inike

y=value ...]

As additional parameters you can add some special

php.ini-values, for example you can set a

docroot="/path/to/docroot" specific to the context php5\_execute is

called. For boolean ini-keys please use 0/1 as value, not

"On","Off",... (this will not work correctly), e.g.

zlib.output\_compression=1 instead of zlib.output\_compression="On"

\* This is only needed if you want to configure a directory that only

consists of PHP scripts (same like a cgi-bin directory):

<Object name="x-httpd-php">

ObjectType fn="force-type" type="magnus-internal/x-httpd-php"

Service fn=php5\_execute [inikey=value inikey=value ...]

</Object>

After that you can configure a directory in the Administration

server and assign it the style x-httpd-php. All files in it will

get executed as PHP. This is nice to hide PHP usage by renaming

files to .html.

\* Restart your web service and apply changes

\* Do it for each web server instance you want PHP to run

Note: More details about setting up PHP as an NSAPI filter can be

found here: http://benoit.noss.free.fr/php/install-php4.html

Note: The stacksize that PHP uses depends on the configuration of

the web server. If you get crashes with very large PHP scripts, it

is recommended to raise it with the Admin Server (in the section

"MAGNUS EDITOR").

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CGI environment and recommended modifications in php.ini

Important when writing PHP scripts is the fact that Sun JSWS/Sun ONE

WS/iPlanet/Netscape is a multithreaded web server. Because of that all

requests are running in the same process space (the space of the web

server itself) and this space has only one environment. If you want to

get CGI variables like PATH\_INFO, HTTP\_HOST etc. it is not the correct

way to try this in the old PHP 3.x way with getenv() or a similar way

(register globals to environment, $\_ENV). You would only get the

environment of the running web server without any valid CGI variables!

Note: Why are there (invalid) CGI variables in the environment?

Answer: This is because you started the web server process from the

admin server which runs the startup script of the web server, you

wanted to start, as a CGI script (a CGI script inside of the admin

server!). This is why the environment of the started web server has

some CGI environment variables in it. You can test this by starting

the web server not from the administration server. Use the command

line as root user and start it manually - you will see there are no

CGI-like environment variables.

Simply change your scripts to get CGI variables in the correct way for

PHP 5.x by using the superglobal $\_SERVER. If you have older scripts

which use $HTTP\_HOST, etc., you should turn on register\_globals in

php.ini and change the variable order too (important: remove "E" from

it, because you do not need the environment here):

variables\_order = "GPCS"

register\_globals = On

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Special use for error pages or self-made directory listings (PHP >= 4.3.3)

You can use PHP to generate the error pages for "404 Not Found" or

similar. Add the following line to the object in obj.conf for every

error page you want to overwrite:

Error fn="php5\_execute" code=XXX script="/path/to/script.php" [inikey=value inik

ey=value...]

where XXX is the HTTP error code. Please delete any other Error

directives which could interfere with yours. If you want to place a

page for all errors that could exist, leave the code parameter out.

Your script can get the HTTP status code with $\_SERVER['ERROR\_TYPE'].

Another possibility is to generate self-made directory listings. Just

create a PHP script which displays a directory listing and replace the

corresponding default Service line for type="magnus-internal/directory"

in obj.conf with the following:

Service fn="php5\_execute" type="magnus-internal/directory" script="/path/to/scri

pt.php" [inikey=value inikey=value...]

For both error and directory listing pages the original URI and

translated URI are in the variables $\_SERVER['PATH\_INFO'] and

$\_SERVER['PATH\_TRANSLATED'].

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Note about nsapi\_virtual() and subrequests (PHP >= 4.3.3)

The NSAPI module now supports the nsapi\_virtual() function (alias:

virtual()) to make subrequests on the web server and insert the result

in the web page. The problem is, that this function uses some

undocumented features from the NSAPI library.

Under Unix this is not a problem, because the module automatically

looks for the needed functions and uses them if available. If not,

nsapi\_virtual() is disabled.

Under Windows limitations in the DLL handling need the use of a

automatic detection of the most recent ns-httpdXX.dll file. This is

tested for servers till version 6.1. If a newer version of the Sun

server is used, the detection fails and nsapi\_virtual() is disabled.

If this is the case, try the following: Add the following parameter to

php5\_init in magnus.conf/obj.conf:

Init fn=php5\_init ... server\_lib="ns-httpdXX.dll"

where XX is the correct DLL version number. To get it, look in the

server-root for the correct DLL name. The DLL with the biggest filesize

is the right one.

You can check the status by using the phpinfo() function.

Note: But be warned: Support for nsapi\_virtual() is EXPERIMENTAL!!!

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

OmniHTTPd Server

This section contains notes and hints specific to OmniHTTPd on Windows.

Note: You should read the manual installation steps first!

Warning

By using the CGI setup, your server is open to several possible

attacks. Please read our CGI security section to learn how to defend

yourself from those attacks.

You need to complete the following steps to make PHP work with

OmniHTTPd. This is a CGI executable setup. SAPI is supported by

OmniHTTPd, but some tests have shown that it is not so stable to use

PHP as an ISAPI module.

Important for CGI users: Read the faq on cgi.force\_redirect for

important details. This directive needs to be set to 0.

1. Install OmniHTTPd server.

2. Right click on the blue OmniHTTPd icon in the system tray and

select Properties

3. Click on Web Server Global Settings

4. On the 'External' tab, enter: virtual = .php | actual =

c:\php\php.exe (use php-cgi.exe if installing PHP 5), and use the

Add button.

5. On the Mime tab, enter: virtual = wwwserver/stdcgi | actual = .php,

and use the Add button.

6. Click OK

Repeat steps 2 - 6 for each extension you want to associate with PHP.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Xitami on Microsoft Windows

This section contains notes and hints specific to Xitami on Windows.

Note: You should read the manual installation steps first!

This list describes how to set up the PHP CGI binary to work with

Xitami on Windows.

Important for CGI users: Read the faq on cgi.force\_redirect for

important details. This directive needs to be set to 0. If you want

to use $\_SERVER['PHP\_SELF'] you have to enable the cgi.fix\_pathinfo

directive.

Warning

By using the CGI setup, your server is open to several possible

attacks. Please read our CGI security section to learn how to defend

yourself from those attacks.

\* Make sure the web server is running, and point your browser to

xitamis admin console (usually http://127.0.0.1/admin), and click

on Configuration.

\* Navigate to the Filters, and put the extension which PHP should

parse (i.e. .php) into the field File extensions (.xxx).

\* In Filter command or script put the path and name of your PHP CGI

executable i.e. C:\php\php-cgi.exe.

\* Press the 'Save' icon.

\* Restart the server to reflect changes.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Installation of extensions on Windows

After installing PHP and a web server on Windows, you will probably

want to install some extensions for added functionality. You can choose

which extensions you would like to load when PHP starts by modifying

your php.ini. You can also load a module dynamically in your script

using dl().

The DLLs for PHP extensions are prefixed with php\_.

Many extensions are built into the Windows version of PHP. This means

additional DLL files, and the extension directive, are not used to load

these extensions. The Windows PHP Extensions table lists extensions

that require, or used to require, additional PHP DLL files. Here's a

list of built in extensions:

In PHP 5 (updated PHP 5.0.4), the following changes exist. Built in:

DOM, LibXML, Iconv, SimpleXML, SPL and SQLite. And the following are no

longer built in: MySQL and Overload.

The default location PHP searches for extensions is C:\php5 in PHP 5.

To change this setting to reflect your setup of PHP edit your php.ini

file:

\* You will need to change the extension\_dir setting to point to the

directory where your extensions lives, or where you have placed

your php\_\*.dll files. For example:

extension\_dir = C:\php\extensions

\* Enable the extension(s) in php.ini you want to use by uncommenting

the extension=php\_\*.dll lines in php.ini. This is done by deleting

the leading ; from the extension you want to load.

Example 2-8. Enable Bzip2 extension for PHP-Windows

// change the following line from ...

;extension=php\_bz2.dll

// ... to

extension=php\_bz2.dll

\* Some of the extensions need extra DLLs to work. Couple of them can

be found in the distribution package, in in the main folder in PHP 5,

but some, for example Oracle (php\_oci8.dll) require DLLs which are

not bundled with the distribution package.

\* Some of these DLLs are not bundled with the PHP distribution. See

each extensions documentation page for details. Also, read the

manual section titled Installation of PECL extensions for details

on PECL. An increasingly large number of PHP extensions are found

in PECL, and these extensions require a separate download.

Note: If you are running a server module version of PHP remember to

restart your web server to reflect your changes to php.ini.

The following table describes some of the extensions available and

required additional dlls.

Table 2-1. PHP Extensions

Extension Description Notes

php\_bz2.dll bzip2 compression functions None

php\_calendar.dll Calendar conversion functions

php\_cpdf.dll ClibPDF functions None

php\_crack.dll Crack functions None

php\_ctype.dll ctype family functions

php\_curl.dll CURL, Client URL library functions Requires: libeay32.dll,

ssleay32.dll (bundled)

php\_db.dll DBM functions Deprecated. Use DBA instead (php\_dba.dll)

php\_dba.dll DBA: DataBase (dbm-style) Abstraction layer functions None

php\_dbase.dll dBase functions None

php\_dbx.dll dbx functions

php\_exif.dll EXIF functions php\_mbstring.dll. And, php\_exif.dll must be

loaded after php\_mbstring.dll in php.ini.

php\_fdf.dll FDF: Forms Data Format functions. Requires: fdftk.dll

(bundled)

php\_filepro.dll filePro functions Read-only access

php\_ftp.dll FTP functions

php\_gd2.dll GD library image functions GD2

php\_gettext.dll Gettext functions, requires libintl-1.dll,

iconv.dll (bundled).

php\_iconv.dll ICONV characterset conversion Requires: iconv.dll

php\_imap.dll IMAP POP3 and NNTP functions None

php\_interbase.dll InterBase functions Requires: gds32.dll (bundled)

php\_ldap.dll LDAP functions requires libeay32.dll, ssleay32.dll (bundled)

php\_mbstring.dll Multi-Byte String functions None

php\_mcrypt.dll Mcrypt Encryption functions Requires: libmcrypt.dll

php\_mime\_magic.dll Mimetype functions Requires: magic.mime (bundled)

php\_ming.dll Ming functions for Flash None

php\_msql.dll mSQL functions Requires: msql.dll (bundled)

php\_mssql.dll MSSQL functions Requires: ntwdblib.dll (bundled)

php\_mysql.dll MySQL functions PHP >= 5.0.0, requires libmysql.dll

(bundled)

php\_mysqli.dll MySQLi functions PHP >= 5.0.0, requires libmysql.dll

(libmysqli.dll in PHP <= 5.0.2) (bundled)

php\_oci8.dll Oracle 8 functions Requires: Oracle 8.1+ client libraries

php\_openssl.dll OpenSSL functions Requires: libeay32.dll (bundled)

php\_oracle.dll Oracle functions Requires: Oracle 7 client libraries

php\_pgsql.dll PostgreSQL functions None

php\_printer.dll Printer functions None

php\_shmop.dll Shared Memory functions None

php\_snmp.dll SNMP get and walk functions NT only!

php\_soap.dll SOAP functions PHP >= 5.0.0

php\_sockets.dll Socket functions None

php\_sybase\_ct.dll Sybase functions Requires: Sybase client libraries

php\_tidy.dll Tidy functions PHP >= 5.0.0

php\_tokenizer.dll Tokenizer functions Built in since PHP 4.3.0

php\_xmlrpc.dll XML-RPC functions PHP >= 4.2.1 requires: iconv.dll

(bundled)

php\_xslt.dll XSLT requires libxslt.dll, iconv.dll (bundled).

php\_zip.dll Zip File functions

php\_zlib.dll ZLib compression functions

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Chapter 3. Installation of PECL extensions

Introduction to PECL Installations

PECL is a repository of PHP extensions that are made available to you

via the PEAR packaging system. This section of the manual is intended

to demonstrate how to obtain and install PECL extensions.

These instructions assume /your/phpsrcdir/ is the path to the PHP

source distribution, and that extname is the name of the PECL

extension. Adjust accordingly. These instructions also assume a

familiarity with the pear command. The information in the PEAR manual

for the pear command also applies to the pecl command.

To be useful, a shared extension must be built, installed, and loaded.

The methods described below provide you with various instructions on

how to build and install the extensions, but they do not automatically

load them. Extensions can be loaded by adding an extension directive.

To this php.ini file, or through the use of the dl() function.

When building PHP modules, it's important to have known-good versions

of the required tools (autoconf, automake, libtool, etc.) See the

SVN Instructions for details on the required tools, and required

versions.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Downloading PECL extensions

There are several options for downloading PECL extensions, such as:

\* http://pecl.php.net

The PECL web site contains information about the different

extensions that are offered by the PHP Development Team. The

information available here includes: ChangeLog, release notes,

requirements and other similar details.

\* pecl download extname

PECL extensions that have releases listed on the PECL web site are

available for download and installation using the pecl command.

Specific revisions may also be specified.

\* SVN

Most PECL extensions also reside in SVN. A web-based view may be

seen at http://svn.php.net/pecl/. To download straight from SVN,

the following sequence of commands may be used.

$ svn co http://svn.php.net/repository/pecl/<extname>/trunk

\* Windows downloads

Windows users may find compiled PECL binaries by downloading the

Collection of PECL modules from the PHP Downloads page, or by

retrieving a PECL Snapshot or an extension DLL on PECL4WIN. To

compile PHP under Windows, read the appropriate chapter.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

PECL for Windows users

As with any other PHP extension DLL, installation is as simple as

copying the PECL extension DLLs into the extension\_dir folder and

loading them from php.ini. For example, add the following line to your

php.ini:

extension=php\_extname.dll

After doing this, restart the web server.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Compiling shared PECL extensions with the pecl command

PECL makes it easy to create shared PHP extensions. Using the pecl

command, do the following:

$ pecl install extname

This will download the source for extname, compile, and install

extname.so into your extension\_dir. extname.so may then be loaded via

php.ini

By default, the pecl command will not install packages that are marked

with the alpha or beta state. If no stable packages are available, you

may install a beta package using the following command:

$ pecl install extname-beta

You may also install a specific version using this variant:

$ pecl install extname-0.1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Compiling shared PECL extensions with phpize

Sometimes, using the pecl installer is not an option. This could be

because you're behind a firewall, or it could be because the extension

you want to install is not available as a PECL compatible package, such

as unreleased extensions from SVN. If you need to build such an

extension, you can use the lower-level build tools to perform the build

manually.

The phpize command is used to prepare the build environment for a PHP

extension. In the following sample, the sources for an extension are in

a directory named extname:

$ cd extname

$ phpize

$ ./configure

$ make

# make install

A successful install will have created extname.so and put it into the

PHP extensions directory. You'll need to and adjust php.ini and add an

extension=extname.so line before you can use the extension.

If the system is missing the phpize command, and precompiled packages

(like RPM's) are used, be sure to also install the appropriate devel

version of the PHP package as they often include the phpize command

along with the appropriate header files to build PHP and its

extensions.

Execute phpize --help to display additional usage information.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Compiling PECL extensions statically into PHP

You might find that you need to build a PECL extension statically into

your PHP binary. To do this, you'll need to place the extension source

under the php-src/ext/ directory and tell the PHP build system to

regenerate its configure script.

$ cd /your/phpsrcdir/ext

$ pecl download extname

$ gzip -d < extname.tgz | tar -xvf -

$ mv extname-x.x.x extname

This will result in the following directory:

/your/phpsrcdir/ext/extname

From here, force PHP to rebuild the configure script, and then build

PHP as normal:

$ cd /your/phpsrcdir

$ rm configure

$ ./buildconf --force

$ ./configure --help

$ ./configure --with-extname --enable-someotherext --with-foobar

$ make

$ make install

Note: To run the 'buildconf' script you need autoconf 2.13 and

automake 1.4+ (newer versions of autoconf may work, but are not

supported).

Whether --enable-extname or --with-extname is used depends on the

extension. Typically an extension that does not require external

libraries uses --enable. To be sure, run the following after buildconf:

$ ./configure --help | grep extname

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Chapter 4. Problems?

Read the FAQ

Some problems are more common than others. The most common ones are

listed in the PHP FAQ, part of this manual.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Other problems

If you are still stuck, someone on the PHP installation mailing list

may be able to help you. You should check out the archive first, in

case someone already answered someone else who had the same problem as

you. The archives are available from the support page on

http://www.php.net/support.php. To subscribe to the PHP installation

mailing list, send an empty mail to

php-install-subscribe@lists.php.net. The mailing list address is

php-install@lists.php.net.

If you want to get help on the mailing list, please try to be precise

and give the necessary details about your environment (which operating

system, what PHP version, what web server, if you are running PHP as

CGI or a server module, safe mode, etc...), and preferably enough code

to make others able to reproduce and test your problem.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Bug reports

If you think you have found a bug in PHP, please report it. The PHP

developers probably don't know about it, and unless you report it,

chances are it won't be fixed. You can report bugs using the

bug-tracking system at http://bugs.php.net/. Please do not send bug

reports in mailing list or personal letters. The bug system is also

suitable to submit feature requests.

Read the How to report a bug document before submitting any bug

reports!

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Chapter 5. Runtime Configuration

The configuration file

The configuration file (called php3.ini in PHP 3, and simply php.ini as

of PHP 4) is read when PHP starts up. For the server module versions of

PHP, this happens only once when the web server is started. For the CGI

and CLI version, it happens on every invocation.

php.ini is searched in these locations (in order):

\* SAPI module specific location (PHPIniDir directive in Apache 2, -c

command line option in CGI and CLI, php\_ini parameter in NSAPI,

PHP\_INI\_PATH environment variable in THTTPD)

\* The PHPRC environment variable. Before PHP 5.2.0 this was checked

after the registry key mentioned below.

\* As of PHP 5.2.0, the following registry locations are searched in

order: HKEY\_LOCAL\_MACHINE\SOFTWARE\PHP\x.y.z\IniFilePath,

HKEY\_LOCAL\_MACHINE\SOFTWARE\PHP\x.y\IniFilePath and

HKEY\_LOCAL\_MACHINE\SOFTWARE\PHP\x\IniFilePath, where x, y and z

mean the PHP major, minor and release versions.

\* HKEY\_LOCAL\_MACHINE\SOFTWARE\PHP\IniFilePath (Windows Registry

location)

\* Current working directory (except CLI)

\* The web server's directory (for SAPI modules), or directory of PHP

(otherwise in Windows)

\* Windows directory (C:\windows or C:\winnt) (for Windows), or

--with-config-file-path compile time option

If php-SAPI.ini exists (where SAPI is used SAPI, so the filename is

e.g. php-cli.ini or php-apache.ini), it's used instead of php.ini. SAPI

name can be determined by php\_sapi\_name().

Note: The Apache web server changes the directory to root at startup

causing PHP to attempt to read php.ini from the root filesystem if

it exists.

The php.ini directives handled by extensions are documented

respectively on the pages of the extensions themselves. The list of the

core directives is available in the appendix. Probably not all PHP

directives are documented in the manual though. For a complete list of

directives available in your PHP version, please read your well

commented php.ini file. Alternatively, you may find the the latest

php.ini from SVN helpful too.

Example 5-1. php.ini example

; any text on a line after an unquoted semicolon (;) is ignored

[php] ; section markers (text within square brackets) are also ignored

; Boolean values can be set to either:

; true, on, yes

; or false, off, no, none

register\_globals = off

track\_errors = yes

; you can enclose strings in double-quotes

include\_path = ".:/usr/local/lib/php"

; backslashes are treated the same as any other character

include\_path = ".;c:\php\lib"

Since PHP 5.1.0, it is possible to refer to existing .ini variables

from within .ini files. Example: open\_basedir = ${open\_basedir}

":/new/dir".

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

How to change configuration settings

Running PHP as an Apache module

When using PHP as an Apache module, you can also change the

configuration settings using directives in Apache configuration files

(e.g. httpd.conf) and .htaccess files. You will need "AllowOverride

Options" or "AllowOverride All" privileges to do so.

With PHP 4 and PHP 5, there are several Apache directives that allow

you to change the PHP configuration from within the Apache

configuration files. For a listing of which directives are PHP\_INI\_ALL,

PHP\_INI\_PERDIR, or PHP\_INI\_SYSTEM, have a look at the List of php.ini

directives appendix.

Note: With PHP 3, there are Apache directives that correspond to

each configuration setting in the php3.ini name, except the name is

prefixed by "php3\_".

php\_value name value

Sets the value of the specified directive. Can be used only with

PHP\_INI\_ALL and PHP\_INI\_PERDIR type directives. To clear a

previously set value use none as the value.

Note: Don't use php\_value to set boolean values. php\_flag (see

below) should be used instead.

php\_flag name on|off

Used to set a boolean configuration directive. Can be used only

with PHP\_INI\_ALL and PHP\_INI\_PERDIR type directives.

php\_admin\_value name value

Sets the value of the specified directive. This can not be used

in .htaccess files. Any directive type set with php\_admin\_value

can not be overridden by .htaccess or virtualhost directives. To

clear a previously set value use none as the value.

php\_admin\_flag name on|off

Used to set a boolean configuration directive. This can not be

used in .htaccess files. Any directive type set with

php\_admin\_flag can not be overridden by .htaccess or virtualhost

directives.

Example 5-2. Apache configuration example

<IfModule mod\_php5.c>

php\_value include\_path ".:/usr/local/lib/php"

php\_admin\_flag safe\_mode on

</IfModule>

Caution

PHP constants do not exist outside of PHP. For example, in httpd.conf

you can not use PHP constants such as E\_ALL or E\_NOTICE to set the

error\_reporting directive as they will have no meaning and will

evaluate to 0. Use the associated bitmask values instead. These

constants can be used in php.ini

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Changing PHP configuration via the Windows registry

When running PHP on Windows, the configuration values can be modified

on a per-directory basis using the Windows registry. The configuration

values are stored in the registry key HKLM\SOFTWARE\PHP\Per Directory

Values, in the sub-keys corresponding to the path names. For example,

configuration values for the directory c:\inetpub\wwwroot would be

stored in the key HKLM\SOFTWARE\PHP\Per Directory

Values\c\inetpub\wwwroot. The settings for the directory would be

active for any script running from this directory or any subdirectory

of it. The values under the key should have the name of the PHP

configuration directive and the string value. PHP constants in the

values are not parsed. However, only configuration values changeable in

PHP\_INI\_USER can be set this way, PHP\_INI\_PERDIR values can not.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Other interfaces to PHP

Regardless of how you run PHP, you can change certain values at runtime

of your scripts through ini\_set(). See the documentation on the

ini\_set() page for more information.

If you are interested in a complete list of configuration settings on

your system with their current values, you can execute the phpinfo()

function, and review the resulting page. You can also access the values

of individual configuration directives at runtime using ini\_get() or

get\_cfg\_var().

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Chapter 6. Installation FAQ

This section holds common questions about the way to install PHP. PHP

is available for almost any OS (except maybe for MacOS before OSX), and

almost any web server.

To install PHP, follow the instructions in Installing PHP.

1. Why shouldn't I use Apache2 with a threaded MPM in a production

environment?

2. Unix/Windows: Where should my php.ini file be located?

3. Unix: I installed PHP, but every time I load a document, I get the

message 'Document Contains No Data'! What's going on here?

4. Unix: I installed PHP using RPMS, but Apache isn't processing the

PHP pages! What's going on here?

5. Unix: I installed PHP 3 using RPMS, but it doesn't compile with the

database support I need! What's going on here?

6. Unix: I patched Apache with the FrontPage extensions patch, and

suddenly PHP stopped working. Is PHP incompatible with the

Apache FrontPage extensions?

7. Unix/Windows: I have installed PHP, but when I try to access a PHP

script file via my browser, I get a blank screen.

8. Unix/Windows: I have installed PHP, but when try to access a PHP

script file via my browser, I get a server 500 error.

9. Some operating systems: I have installed PHP without errors, but

when I try to start apache I get undefined symbol errors:

[mybox:user /src/php5] root# apachectl configtest

apachectl: /usr/local/apache/bin/httpd Undefined symbols:

\_compress

\_uncompress

10. Windows: I have installed PHP, but when I to access a PHP script

file via my browser, I get the error:

cgi error:

The specified CGI application misbehaved by not

returning a complete set of HTTP headers.

The headers it did return are:

11. Windows: I've followed all the instructions, but still can't get

PHP and IIS to work together!

12. When running PHP as CGI with IIS, PWS, OmniHTTPD or Xitami, I get

the following error: Security Alert! PHP CGI cannot be accessed

directly..

13. How do I know if my php.ini is being found and read? It seems like

it isn't as my changes aren't being implemented.

14. How do I add my PHP directory to the PATH on Windows?

15. How do I make the php.ini file available to PHP on windows?

16. Is it possible to use Apache content negotiation (MultiViews

option) with PHP?

17. Is PHP limited to process GET and POST request methods only?

1. Why shouldn't I use Apache2 with a threaded MPM in a production

environment?

PHP is glue. It is the glue used to build cool web applications by

sticking dozens of 3rd-party libraries together and making it all

appear as one coherent entity through an intuitive and easy to learn

language interface. The flexibility and power of PHP relies on the

stability and robustness of the underlying platform. It needs a working

OS, a working web server and working 3rd-party libraries to glue

together. When any of these stop working PHP needs ways to identify the

problems and fix them quickly. When you make the underlying framework

more complex by not having completely separate execution threads,

completely separate memory segments and a strong sandbox for each

request to play in, feet of clay are introduced into PHP's system.

If you feel you have to use a threaded MPM, look at a FastCGI

configuration where PHP is running in its own memory space.

And finally, this warning against using a threaded MPM is not as strong

for Windows systems because most libraries on that platform tend to be

threadsafe.

2. Unix/Windows: Where should my php.ini file be located?

By default on Unix it should be in /usr/local/lib which is

<install-path>/lib. Most people will want to change this at

compile-time with the --with-config-file-path flag. You would, for

example, set it with something like:

--with-config-file-path=/etc

And then you would copy php.ini-production from the distribution to

/etc/php.ini and edit it to make any local changes you want.

--with-config-file-scan-dir=PATH

On Windows the default path for the php.ini file is the Windows

directory. If you're using the Apache webserver, php.ini is first

searched in the Apaches install directory, e.g. c:\program files\apache

group\apache. This way you can have different php.ini files for

different versions of Apache on the same machine.

See also the chapter about the configuration file.

3. Unix: I installed PHP, but every time I load a document, I get the

message 'Document Contains No Data'! What's going on here?

This probably means that PHP is having some sort of problem and is

core-dumping. Look in your server error log to see if this is the case,

and then try to reproduce the problem with a small test case. If you

know how to use 'gdb', it is very helpful when you can provide a

backtrace with your bug report to help the developers pinpoint the

problem. If you are using PHP as an Apache module try something like:

\* Stop your httpd processes

\* gdb httpd

\* Stop your httpd processes

\* > run -X -f /path/to/httpd.conf

\* Then fetch the URL causing the problem with your browser

\* > run -X -f /path/to/httpd.conf

\* If you are getting a core dump, gdb should inform you of this now

\* type: bt

\* You should include your backtrace in your bug report. This should

be submitted to http://bugs.php.net/

If your script uses the regular expression functions (ereg() and

friends), you should make sure that you compiled PHP and Apache with

the same regular expression package. This should happen automatically

with PHP and Apache 1.3.x

4. Unix: I installed PHP using RPMS, but Apache isn't processing the

PHP pages! What's going on here?

Assuming you installed both Apache and PHP from RPM packages, you need

to uncomment or add some or all of the following lines in your

httpd.conf file:

# Extra Modules

AddModule mod\_php.c

AddModule mod\_php3.c

AddModule mod\_perl.c

# Extra Modules

LoadModule php\_module modules/mod\_php.so

LoadModule perl\_module modules/libperl.so

And add:

AddType application/x-httpd-php3 .php3 # for PHP 3

AddType application/x-httpd-php .php # for PHP 4

... to the global properties, or to the properties of the VirtualDomain

you want to have PHP support added to.

5. Unix: I installed PHP 3 using RPMS, but it doesn't compile with the

database support I need! What's going on here?

Due to the way PHP 3 built, it is not easy to build a complete flexible

PHP RPM. This issue is addressed in PHP 4. For PHP 3, we currently

suggest you use the mechanism described in the INSTALL.REDHAT file in

the PHP distribution. If you insist on using an RPM version of PHP 3,

read on...

The RPM packagers are setting up the RPMS to install without database

support to simplify installations and because RPMS use /usr/ instead of

the standard /usr/local/ directory for files. You need to tell the RPM

spec file which databases to support and the location of the top-level

of your database server.

This example will explain the process of adding support for the popular

MySQL database server, using the mod installation for Apache.

Of course all of this information can be adjusted for any database

server that PHP supports. We will assume you installed MySQL and Apache

completely with RPMS for this example as well.

\* First remove mod\_php3 :

rpm -e mod\_php3

\* Then get the source rpm and INSTALL it, NOT --rebuild

rpm -Uvh mod\_php3-3.0.5-2.src.rpm

\* Then edit the /usr/src/redhat/SPECS/mod\_php3.spec file

In the %build section add the database support you want, and the

path.

For MySQL you would add --with-mysql=/usr The %build section will

look something like this:

./configure --prefix=/usr \

--with-apxs=/usr/sbin/apxs \

--with-config-file-path=/usr/lib \

--enable-debug=no \

--enable-safe-mode \

--with-exec-dir=/usr/bin \

--with-mysql=/usr \

--with-system-regex

\* Once this modification is made then build the binary rpm as

follows:

rpm -bb /usr/src/redhat/SPECS/mod\_php3.spec

\* Then install the rpm

rpm -ivh /usr/src/redhat/RPMS/i386/mod\_php3-3.0.5-2.i386.rpm

Make sure you restart Apache, and you now have PHP 3 with MySQL support

using RPM's. Note that it is probably much easier to just build from

the distribution tarball of PHP 3 and follow the instructions in

INSTALL.REDHAT found in that distribution.

6. Unix: I patched Apache with the FrontPage extensions patch, and

suddenly PHP stopped working. Is PHP incompatible with the Apache

FrontPage extensions?

No, PHP works fine with the FrontPage extensions. The problem is that

the FrontPage patch modifies several Apache structures, that PHP relies

on. Recompiling PHP (using 'make clean ; make') after the FP patch is

applied would solve the problem.

7. Unix/Windows: I have installed PHP, but when I try to access a PHP

script file via my browser, I get a blank screen.

Do a 'view source' in the web browser and you will probably find that

you can see the source code of your PHP script. This means that the web

server did not send the script to PHP for interpretation. Something is

wrong with the server configuration - double check the server

configuration against the PHP installation instructions.

8. Unix/Windows: I have installed PHP, but when try to access a PHP

script file via my browser, I get a server 500 error.

Something went wrong when the server tried to run PHP. To get to see a

sensible error message, from the command line, change to the directory

containing the PHP executable (php.exe on Windows) and run php -i. If

PHP has any problems running, then a suitable error message will be

displayed which will give you a clue as to what needs to be done next.

If you get a screen full of HTML codes (the output of the phpinfo()

function) then PHP is working, and your problem may be related to your

server configuration which you should double check.

9. Some operating systems: I have installed PHP without errors, but

when I try to start apache I get undefined symbol errors:

[mybox:user /src/php5] root# apachectl configtest

apachectl: /usr/local/apache/bin/httpd Undefined symbols:

\_compress

\_uncompress

This has actually nothing to do with PHP, but with the MySQL client

libraries. Some need --with-zlib, others do not. This is also covered

in the MySQL FAQ.

10. Windows: I have installed PHP, but when I to access a PHP script

file via my browser, I get the error:

cgi error:

The specified CGI application misbehaved by not

returning a complete set of HTTP headers.

The headers it did return are:

This error message means that PHP failed to output anything at all. To

get to see a sensible error message, from the command line, change to

the directory containing the PHP executable (php.exe on Windows) and

run php -i. If PHP has any problems running, then a suitable error

message will be displayed which will give you a clue as to what needs

to be done next. If you get a screen full of HTML codes (the output of

the phpinfo() function) then PHP is working.

Once PHP is working at the command line, try accessing the script via

the browser again. If it still fails then it could be one of the

following:

\* File permissions on your PHP script, php.exe, php5ts.dll, php.ini

or any PHP extensions you are trying to load are such that the

anonymous internet user ISUR\_<machinename> cannot access them.

\* The script file does not exist (or possibly isn't where you think

it is relative to your web root directory). Note that for IIS you

can trap this error by ticking the 'check file exists' box when

setting up the script mappings in the Internet Services Manager. If

a script file does not exist then the server will return a 404

error instead. There is also the additional benefit that IIS will

do any authentication required for you based on the NTLanMan

permissions on your script file.

11. Windows: I've followed all the instructions, but still can't get

PHP and IIS to work together!

Make sure any user who needs to run a PHP script has the rights to run

php.exe! IIS uses an anonymous user which is added at the time IIS is

installed. This user needs rights to php.exe. Also, any authenticated

user will also need rights to execute php.exe. And for IIS4 you need to

tell it that PHP is a script engine. Also, you will want to read this

faq.

12. When running PHP as CGI with IIS, PWS, OmniHTTPD or Xitami, I get

the following error: Security Alert! PHP CGI cannot be accessed

directly..

You must set the cgi.force\_redirect directive to 0. It defaults to 1 so

be sure the directive isn't commented out (with a ;). Like all

directives, this is set in php.ini

Because the default is 1, it's critical that you're 100% sure that the

correct php.ini file is being read. Read this faq for details.

13. How do I know if my php.ini is being found and read? It seems like

it isn't as my changes aren't being implemented.

To be sure your php.ini is being read by PHP, make a call to phpinfo()

and near the top will be a listing called Configuration File (php.ini).

This will tell you where PHP is looking for php.ini and whether or not

it's being read. If just a directory PATH exists than it's not being

read and you should put your php.ini in that directory. If php.ini is

included within the PATH than it is being read.

If php.ini is being read and you're running PHP as a module, then be

sure to restart your web server after making changes to php.ini

14. How do I add my PHP directory to the PATH on Windows?

On Windows NT, 2000, XP and 2003:

\* Go to Control Panel and open the System icon (Start -> Settings ->

Control Panel -> System, or just Start -> Control Panel -> System

for Windows XP/2003)

\* Go to the Advanced tab

\* Click on the 'Environment Variables' button

\* Look into the 'System Variables' pane

\* Find the Path entry (you may need to scroll to find it)

\* Double click on the Path entry

\* Enter your PHP directory at the end, including ';' before (e.g.

;C:\php)

\* Press OK and restart your computer

On Windows 98/Me you need to edit the autoexec.bat file:

\* Open the Notepad (Start -> Run and enter notepad)

\* Open the C:\autoexec.bat file

\* Locate the line with PATH=C:\WINDOWS;C:\WINDOWS\COMMAND;..... and

add: ;C:\php to the end of the line

\* Save the file and restart your computer

Note: Be sure to reboot after following the steps above to ensure

that the PATH changes are applied.

The PHP manual used to promote the copying of files into the Windows

system directory, this is because this directory (C:\Windows, C:\WINNT,

etc.) is by default in the systems PATH. Copying files into the Windows

system directory has long since been deprecated and may cause problems.

15. How do I make the php.ini file available to PHP on windows?

There are several ways of doing this. If you are using Apache, read

their installation specific instructions (Apache 1, Apache 2),

otherwise you must set the PHPRC environment variable:

On Windows NT, 2000, XP and 2003:

\* Go to Control Panel and open the System icon (Start -> Settings ->

Control Panel -> System, or just Start -> Control Panel -> System

for Windows XP/2003)

\* Go to the Advanced tab

\* Click on the 'Environment Variables' button

\* Look into the 'System variables' pane

\* Click on 'New' and enter 'PHPRC' as the variable name and the

directory where php.ini is located as the variable value (e.g.

C:\php)

\* Press OK and restart your computer

On Windows 98/Me you need to edit the autoexec.bat file:

\* Open the Notepad (Start -> Run and enter notepad)

\* Open the C:\autoexec.bat file

\* Add a new line to the end of the file: set PHPRC=C:\php (replace

C:\php with the directory where php.ini is located). Please note

that the path cannot contain spaces. For instance, if you have

installed PHP in C:\Program Files\PHP, you would enter

C:\PROGRA~1\PHP instead.

\* Save the file and restart your computer

16. Is it possible to use Apache content negotiation (MultiViews

option) with PHP?

If links to PHP files include extension, everything works perfect. This

FAQ is only for the case when links to PHP files don't include

extension and you want to use content negotiation to choose PHP files

from URL with no extension. In this case, replace the line AddType

application/x-httpd-php .php with:

# PHP 4

AddHandler php-script php

AddType text/html php

# PHP 5

AddHandler php5-script php

AddType text/html php

This solution doesn't work for Apache 1 as PHP module doesn't catch

php-script.

17. Is PHP limited to process GET and POST request methods only?

No, it is possible to handle any request method, e.g. CONNECT. Proper

response status can be sent with header(). If only GET and POST methods

should be handled, it can be achieved with this Apache configuration:

<LimitExcept GET POST>

Deny from all

</LimitExcept>