PHP Manual

Stig Sæther Bakken
Alexander Aulbach
Egon Schmid
Jim Winstead
Lars Torben Wilson
Rasmus Lerdorf
Andrei Zmievski
Jouni Ahto

Edited by

Stig Sæther Bakken
Egon Schmid

27-01-2003

Table of Contents

Preface

> PHP, which stands for "PHP: Hypertext Preprocessor" is a widely-used Open Source general-purpose scripting language that is especially suited for Web development and can be embedded into HTML. Its syntax draws upon C, Java, and Perl, and is easy to learn. The main goal of the language is to allow web developers to write dynamically generated webpages quickly, but you can do much more with PHP.

This manual consists primarily of a function reference, but also contains a language reference, explanations of some of PHP's major features, and other supplemental information.

You can download this manual in several formats at http://www.php.net/docs.php. The downloads are updated as the content changes. More information about how this manual is developed can be found in the 'About the manual' appendix.

See also PHP History

I. Getting Started

Table of Contents

Chapter 1. Introduction

What is PHP?

PHP (recursive acronym for "PHP: Hypertext Preprocessor") is a widely-used Open Source general-purpose scripting language that is especially suited for Web development and can be embedded into HTML.

Simple answer, but what does that mean? An example:

Example 1-1. An introductory example

```
<html>
```

```
    <head>
        <title>Example</title>
    </head>
    <body>

        <?php
        echo "Hi, I'm a PHP script!";
        ?>

    </body>
</html>
```

Notice how this is different from a script written in other languages like Perl or C -- instead of writing a program with lots of commands to output HTML, you write an HTML script with some embedded code to do something (in this case, output some text). The PHP code is enclosed in special start and end tags that allow you to jump into and out of "PHP mode".

What distinguishes PHP from something like client-side JavaScript is that the code is executed on the server. If you were to have a script similar to the above on your server, the client would receive the results of running that script, with no way of determining what the underlying code may be. You can even configure your web server to process all your HTML files with PHP, and then there's really no way that users can tell what you have up your sleeve.

The best things in using PHP are that it is extremely simple for a newcomer, but offers many advanced features for a professional programmer. Don't be afraid reading the long list of PHP's features. You can jump in, in a short time, and start writing simple scripts in a few hours.

Although PHP's development is focused on server-side scripting, you can do much more with it. Read on, and see more in the What can PHP do? section.

What can PHP do?

Anything. PHP is mainly focused on server-side scripting, so you can do anything any other CGI program can do, such as collect form data, generate dynamic page content, or send and receive cookies. But PHP can do much more.

There are three main fields where PHP scripts are used.

- Server-side scripting. This is the most traditional and main target field for PHP. You need three things to make this work. The PHP parser (CGI or server module), a webserver and a web browser. You need to run the webserver, with a connected PHP installation. You can access the PHP program output with a web browser, viewing the PHP page through the server. See the installation instructions section for more information.
- Command line scripting. You can make a PHP script to run it without any server or browser. You only need the PHP parser to use it this way. This type of usage is ideal for scripts regularly executed using cron (on *nix or Linux) or Task Scheduler (on Windows). These scripts can also be used for simple text processing tasks. See the section about Command line usage of PHP for more information.
- Writing client-side GUI applications. PHP is probably not the very best language to write windowing applications, but if you know PHP very well, and would like to use some advanced PHP features in your client-side applications you can also use PHP-GTK to write such programs. You also have the ability to write cross-platform applications this way. PHP-GTK is an extension to PHP, not available in the main distribution. If you are interested in PHP-GTK, visit it's own website.

PHP can be used on all major operating systems, including Linux, many Unix variants (including HP-UX, Solaris and OpenBSD), Microsoft Windows, Mac OS X, RISC OS, and probably others. PHP has also support for most of the web servers today. This includes Apache, Microsoft Internet Information Server, Personal Web Server, Netscape and iPlanet servers, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd, and many others. For the majority of the servers PHP has a module, for the others supporting the CGI standard, PHP can work as a CGI processor.

So with PHP, you have the freedom of choosing an operating system and a web server. Furthermore, you also have the choice of using procedural programming or object oriented programming, or a mixture of them. Although not every standard OOP feature is realized in the current version of PHP, many code libraries and large applications (including the PEAR library) are written only using OOP code.

With PHP you are not limited to output HTML. PHP's abilities includes outputting images, PDF files and even Flash movies (using libswf and Ming) generated on the fly. You can also output easily any text, such as XHTML and any other XML file. PHP can autogenerate these files, and save them in the file system, instead of printing it out, forming a server-side cache for your dynamic content.

One of the strongest and most significant feature in PHP is its support for a wide range of databases. Writing a database-enabled web page is incredibly simple. The following databases are currently supported:

Adabas D	Ingres	Oracle (OCI7 and OCI8)
dBase	InterBase	Ovrimos

Empress	FrontBase	PostgreSQL
FilePro (read-only)	mSQL	Solid
Hyperwave	Direct MS-SQL	Sybase
IBM DB2	MySQL	Velocis
Informix	ODBC	Unix dbm

We also have a DBX database abstraction extension allowing you to transparently use any database supported by that extension. Additionally PHP supports ODBC, the Open Database Connection standard, so you can connect to any other database supporting this world standard.

PHP also has support for talking to other services using protocols such as LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (on Windows) and countless others. You can also open raw network sockets and interact using any other protocol. PHP has support for the WDDX complex data exchange between virtually all Web programming languages. Talking about interconnection, PHP has support for instantiation of Java objects and using them transparently as PHP objects. You can also use our CORBA extension to access remote objects.

PHP has extremely useful text processing features, from the POSIX Extended or Perl regular expressions to parsing XML documents. For parsing and accessing XML documents, we support the SAX and DOM standards. You can use our XSLT extension to transform XML documents.

While using PHP in the ecommerce field, you'll find the Cybercash payment, CyberMUT, VeriSign Payflow Pro and CCVS functions useful for your online payment programs.

At last but not least, we have many other interesting extensions, the mnoGoSearch search engine functions, the IRC Gateway functions, many compression utilities (gzip, bz2), calendar conversion, translation...

As you can see this page is not enough to list all the features and benefits PHP can offer. Read on in the sections about installing PHP, and see the function reference part for explanation of the extensions mentioned here.

Chapter 2. A simple tutorial

Here we would like to show the very basics of PHP in a short simple tutorial. This text only deals with dynamic webpage creation with PHP, though PHP is not only capable of creating webpages. See the section titled What can PHP do for more information.

PHP-enabled web pages are treated just like regular HTML pages and you can create and edit them the same way you normally create regular HTML pages.

What do I need?

In this tutorial we assume that your server has support for PHP activated and that all files ending in `.php` are handled by PHP. On most servers this is the default extension for PHP files, but ask your server administrator to be sure. If your server supports PHP then you don't need to do anything. Just create your `.php` files and put them in your web directory and the server will magically parse them for you. There is no need to compile anything nor do you need to install any extra tools. Think of these PHP-enabled files as simple HTML files with a whole new family of magical tags that let you do all sorts of things.

Let's say you want to save precious bandwidth and develop locally. In this case, you'll want to install a web server, such as Apache, and of course PHP. You'll most likely want to install a database as well, such as MySQL. You can install these individually or a simpler way is to locate a pre-configured package that automatically installs all of these with just a few mouse clicks. It's easy to setup a web server with PHP support on any operating system, including Linux and Windows. In linux, you may find rpmfind helpful for locating RPMs.

Your first PHP-enabled page

Create a file named `hello.php` and put it in your web servers root directory (`DOCUMENT_ROOT`) with the following content:

Example 2-1. Our first PHP script: `hello.php`

```
<html>
 <head>
  <title>PHP Test</title>
 </head>
```

```
 <body>
 <?php echo "<p>Hello World</p>"; ?>
 </body>
</html>
```

Use your browser to access the file with your web access URL, ending with the "/hello.php" file reference. When developing locally this url will be something like `http://localhost/hello.php` or `http://127.0.0.1/hello.php` but this depends on the web servers configuration. Although this is outside the scope of this tutorial, see also the `DocumentRoot` and `ServerName` directives in your web servers configuration file. (on Apache this is `httpd.conf`). If everything is setup correctly, this file will be parsed by PHP and the following output will make it to your browser:

```
<html>
 <head>
  <title>PHP Test</title>
 </head>
 <body>
 <p>Hello World</p>
 </body>
</html>
```

Note that this is not like a CGI script. The file does not need to be executable or special in any way. Think of it as a normal HTML file which happens to have a set of special tags available to you that do a lot of interesting things.

This program is extremely simple and you really didn't need to use PHP to create a page like this. All it does is display: `Hello World` using the PHP **echo()** statement.

If you tried this example and it didn't output anything, or it prompted for download, or you see the whole file as text, chances are that the server you are on does not have PHP enabled. Ask your administrator to enable it for you using the Installation chapter of the manual. If you're developing locally, also read the installation chapter to make sure everything is configured properly. If problems continue to persist, don't hesitate to use one of the many PHP support options.

The point of the example is to show the special PHP tag format. In this example we used `<?php` to indicate the start of a PHP tag. Then we put the PHP statement and left PHP mode by adding the closing tag, `?>`. You may jump in and out of PHP mode in an HTML file like this all you want. For more details, read the manual section on basic PHP syntax.

> **A Note on Text Editors:** There are many text editors and Integrated Development Environments (IDEs) that you can use to create, edit and manage PHP files. A partial list of these tools is maintained at PHP Editor's List. If you wish to recommend an editor, please visit the above page and ask the page maintainer to add the editor to the list. Having an editor with syntax highlighting can be helpful.
>
> **A Note on Word Processors:** Word processors such as StarOffice Writer, Microsoft Word and Abiword are not good choices for editing PHP files. If you wish to use one for this test script, you must ensure that you save the file as PLAIN TEXT or PHP will not be able to read and execute the script.
>
> **A Note on Windows Notepad:** If you are writing your PHP scripts using Windows Notepad, you will need to ensure that your files are saved with the .php extension. (Notepad adds a .txt extension to files automatically unless you take one of the following steps to prevent it.) When you save the file and are prompted to provide a name for the file, place the filename in quotes (i.e. `"hello.php"`). Alternately, you can click on the 'Text Documents' drop-down menu in the save dialog box and change the setting to "All Files". You can then enter your filename without quotes.

Now that you've successfully created a simple PHP script that works, it's time to create the most famous PHP script! Make a call to the **phpinfo()** function and you'll see a lot of useful information about your system and setup such as available Predefined Variables, loaded PHP modules, and configuration settings. Take some time and review this important information.

Something Useful

Let's do something a bit more useful now. We are going to check what sort of browser the person viewing the page is using. In order to do that we check the user agent string that the browser sends as part of its HTTP request. This information is stored in a variable. Variables always start with a dollar-sign in PHP. The variable we are interested in right now is `$_SERVER["HTTP_USER_AGENT"]`.

> **PHP Autoglobals Note:** $_SERVER is a special reserved PHP variable that contains all web server information. It's known as an Autoglobal (or Superglobal). See the related manual page on Autoglobals for more information. These special variables were introduced in PHP 4.1.0. Before this time, we used the older `$HTTP_*_VARS` arrays instead, such as `$HTTP_SERVER_VARS`. Although deprecated, these older variables still exist. (See also the note on old code.)

To display this variable, we can simply do:

Example 2-2. Printing a variable (Array element)

```
<?php echo $_SERVER["HTTP_USER_AGENT"]; ?>
```

A sample output of this script may be:

```
Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
```

There are many types of variables available in PHP. In the above example we printed an Array element. Arrays can be very useful.

`$_SERVER` is just one variable that's automatically made available to you by PHP. A list can be seen in the Reserved Variables section of the manual or you can get a complete list of them by creating a file that looks like this:

Example 2-3. Show all predefined variables with phpinfo()

```
<?php phpinfo(); ?>
```

If you load up this file in your browser you will see a page full of information about PHP along with a list of all the variables available to you.

You can put multiple PHP statements inside a PHP tag and create little blocks of code that do more than just a single echo. For example, if we wanted to check for Internet Explorer we could do something like this:

Example 2-4. Example using control structures and functions

```
<?php
if (strstr($_SERVER["HTTP_USER_AGENT"], "MSIE")) {
        echo "You are using Internet Explorer<br />";
}
?>
```

A sample output of this script may be:

```
You are using Internet Explorer<br />
```

Here we introduce a couple of new concepts. We have an if statement. If you are familiar with the basic syntax used by the C language this should look logical to you. If you don't know enough C or some other language where the syntax used above is used, you should probably pick up any introductory PHP book and read the first couple of chapters, or read the Language Reference part of the manual. You can find a list of PHP books at http://www.php.net/books.php.

The second concept we introduced was the **strstr()** function call. **strstr()** is a function built into PHP which searches a string for another string. In this case we are looking for `"MSIE"` inside `$_SERVER["HTTP_USER_AGENT"]`. If the string is found, the function returns **TRUE** and if it isn't, it returns **FALSE**. If it returns **TRUE**, the if statement evaluates to **TRUE** and the code within its {braces} is executed. Otherwise, it's not. Feel free to create similar examples, with if, else, and other functions such as **strtoupper()** and **strlen()**. Each related manual page contains examples too.

We can take this a step further and show how you can jump in and out of PHP mode even in the middle of a PHP block:

Example 2-5. Mixing both HTML and PHP modes

```
<?php
if (strstr($_SERVER["HTTP_USER_AGENT"], "MSIE")) {
?>
<h3>strstr must have returned true</h3>
<center><b>You are using Internet Explorer</b></center>
<?php
} else {
?>
<h3>strstr must have returned false</h3>
<center><b>You are not using Internet Explorer</b></center>
<?php
}
?>
```

A sample output of this script may be:

```
<h3>strstr must have returned true</h3>
<center><b>You are using Internet Explorer</b></center>
```

Instead of using a PHP echo statement to output something, we jumped out of PHP mode and just sent straight HTML. The important and powerful point to note here is that the logical flow of the script remains intact. Only one of the HTML blocks will end up getting sent to the viewer depending on if **strstr()** returned **TRUE** or **FALSE** In other words, if the string `MSIE` was found or not.

Dealing with Forms

One of the most powerful features of PHP is the way it handles HTML forms. The basic concept that is important to understand is that any form element in a form will automatically be available to your PHP scripts. Please read the manual section on Variables from outside of PHP for more information and examples on using forms with PHP. Here's an example HTML form:

Example 2-6. A simple HTML form

```
<form action="action.php" method="POST">
 Your name: <input type="text" name="name" />
 Your age: <input type="text" name="age" />
 <input type="submit">
</form>
```

There is nothing special about this form. It is a straight HTML form with no special tags of any kind. When the user fills in this form and hits the submit button, the `action.php` page is called. In this file you would have something like this:

Example 2-7. Printing data from our form

```
Hi <?php echo $_POST["name"]; ?>.
You are <?php echo $_POST["age"]; ?> years old.
```

A sample output of this script may be:

```
Hi Joe.
You are 22 years old.
```

It should be obvious what this does. There is nothing more to it. The `$_POST["name"]` and `$_POST["age"]` variables are automatically set for you by PHP. Earlier we used the `$_SERVER` autoglobal, now above we just introduced the $_POST autoglobal which contains all POST data. Notice how the *method* of our form is POST. If we used the method *GET* then our form information would live in the $_GET autoglobal instead. You may also use the $_REQUEST autoglobal if you don't care the source of your request data. It contains a mix of GET, POST, COOKIE and FILE data. See also the **import_request_variables()** function.

Using old code with new versions of PHP

Now that PHP has grown to be a popular scripting language, there are more resources out there that have listings of code you can reuse in your own scripts. For the most part the developers of the PHP language have tried to be backwards compatible, so a script written for an older version should run (ideally) without changes in a newer version of PHP, in practice some changes will usually be needed.

Two of the most important recent changes that affect old code are:

- The deprecation of the old `$HTTP_*_VARS` arrays (which need to be indicated as global when used inside a function or method). The following autoglobal arrays were introduced in PHP 4.1.0. They are: `$_GET`, `$_POST`, `$_COOKIE`, `$_SERVER`, `$_ENV`, `$_REQUEST`, and `$_SESSION`. The older `$HTTP_*_VARS` arrays, such as $HTTP_POST_VARS, still exist and have since PHP 3.
- External variables are no longer registered in the global scope by default. In other words, as of PHP 4.2.0 the PHP directive register_globals is *off* by default in `php.ini`. The preferred method of accessing these values is via the autoglobal arrays mentioned above. Older scripts, books, and tutorials may rely on this directive being on. If on, for example, one could use `$id` from the URL `http://www.example.com/foo.php?id=42`. Whether on or off, `$_GET['id']` is available.

For more details on these changes, see the section on predefined variables and links therein.

What's next?

With what you know now you should be able to understand most of the manual and also the various example scripts available in the example archives. You can also find other examples on the php.net websites in the links section: http://www.php.net/links.php.

Chapter 3. Installation

General Installation Considerations

Before installing 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:

- Server-side scripting
- Command line scripting
- Client-side 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 (eg. Apache on Linux or 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.

While 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 processor. This means you set up your server to use the command line executable of PHP (`php.exe` on Windows) to process all PHP file requests on the server.

If you are also interested to use PHP for command line scripting (eg. 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 client side 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.

Downloading PHP, the source code, and binary distributions for Windows can be found at http://www.php.net/. We recommend you to choose a mirror nearest to you for downloading the distributions.

Unix/HP-UX installs

This section contains notes and hints specific to installing PHP on HP-UX systems.

Example 3-1. Installation Instructions for HP-UX 10

```
From: paul_mckay@clearwater-it.co.uk
04-Jan-2001 09:49
(These tips are for PHP 4.0.4 and Apache v1.3.9)

So you want to install PHP and Apache on a HP-UX 10.20 box?

1. You need gzip, download a binary distribution from
http://hpux.connect.org.uk/ftp/hpux/Gnu/gzip-1.2.4a/gzip-1.2.4a-sd-10.20.depot.Z
uncompress the file and install using swinstall

2. You need gcc, download a binary distribution from
http://gatekeep.cs.utah.edu/ftp/hpux/Gnu/gcc-2.95.2/gcc-2.95.2-sd-10.20.depot.gz
gunzip this file and install gcc using swinstall.

3. You need the GNU binutils, you can download a binary distribution from
http://hpux.connect.org.uk/ftp/hpux/Gnu/binutils-2.9.1/binutils-2.9.1-sd-10.20.depot.gz
gunzip and install using swinstall.

4. You now need bison, you can download a binary distribution from
http://hpux.connect.org.uk/ftp/hpux/Gnu/bison-1.28/bison-1.28-sd-10.20.depot.gz
install as above.

5. You now need flex, you need to download the source from one of the
http://www.gnu.org mirrors. It is in the <filename>non-gnu</filename> directory of the ftp site.
Download the file, gunzip, then tar -xvf it. Go into the newly created flex
directory and do a ./configure, then a make, and then a make install
```

```
If you have errors here, it's probably because gcc etc. are not in your
PATH so add them to your PATH.

Right, now into the hard stuff.

6. Download the PHP and apache sources.

7. gunzip and tar -xvf them.

We need to hack a couple of files so that they can compile ok.

8. Firstly the configure file needs to be hacked because it seems to lose
track of the fact that you are a hpux machine, there will be a
better way of doing this but a cheap and cheerful hack is to put
    lt_target=hpux10.20
on line 47286 of the configure script.

9. Next, the Apache GuessOS file needs to be hacked. Under
apache_1.3.9/src/helpers change line 89 from
    "echo "hp${HPUXMACH}-hpux${HPUXVER}"; exit 0"
to:
    "echo "hp${HPUXMACH}-hp-hpux${HPUXVER}"; exit 0"

10. You cannot install PHP as a shared object under HP-UX so you must compile
it as a static, just follow the instructions at the Apache page.

11. PHP and apache should have compiled OK, but Apache won't start. you need
to create a new user for Apache, eg www, or apache. You then change lines 252
and 253 of the conf/httpd.conf in Apache so that instead of
    User nobody
    Group nogroup
you have something like
    User www
    Group sys

This is because you can't run Apache as nobody under hp-ux.
Apache and PHP should then work.

Hope this helps somebody,
Paul Mckay.
```

Unix/Linux installs

This section contains notes and hints specific to installing PHP on Linux distributions.

Using Packages

Many Linux distributions have some sort of package installation system, such as RPM. This can assist in setting up a standard configuration, but if you need to have a different set of features (such as a secure server, or a different database driver), you may need to build PHP and/or your webserver. If you are unfamiliar with building and compiling your own software, it is worth checking to see whether somebody has already built a packaged version of PHP with the features you need.

Unix/Mac OS X installs

This section contains notes and hints specific to installing PHP on Mac OS X Server.

Using Packages

There are a few pre-packaged and pre-compiled versions of PHP for Mac OS X. This can help in setting up a standard configuration, but if you need to have a different set of features (such as a secure server, or a different database driver), you may need to build PHP and/or your web server yourself. If you are unfamiliar with building and compiling your own software, it's worth checking whether somebody has already built a packaged version of PHP with the features you need.

Compiling for OS X server

There are two slightly different versions of Mac OS X, client and server. The following is for OS X Server.

Example 3-2. Mac OS X server install

```
1. Get the latest distributions of Apache and PHP
2. Untar them, and run the configure program on Apache like so.
    ./configure --exec-prefix=/usr \
    --localstatedir=/var \
    --mandir=/usr/share/man \
    --libexecdir=/System/Library/Apache/Modules \
    --iconsdir=/System/Library/Apache/Icons \
    --includedir=/System/Library/Frameworks/Apache.framework/Versions/1.3/Headers \
    --enable-shared=max \
    --enable-module=most \
    --target=apache

4. You may also want to add this line:
    setenv OPTIM=-O2
    If you want the compiler to do some optimization.

5. Next, go to the PHP 4 source directory and configure it.
    ./configure --prefix=/usr \
    --sysconfdir=/etc \
    --localstatedir=/var \
    --mandir=/usr/share/man \
    --with-xml \
    --with-apache=/src/apache_1.3.12

    If you have any other additions (MySQL, GD, etc.), be sure to add
    them here. For the --with-apache string, put in the path to your
    apache source directory, for example "/src/apache_1.3.12".
6. make
7. make install
    This will add a directory to your Apache source directory under
    src/modules/php4.

8. Now, reconfigure Apache to build in PHP 4.
    ./configure --exec-prefix=/usr \
    --localstatedir=/var \
    --mandir=/usr/share/man \
    --libexecdir=/System/Library/Apache/Modules \
    --iconsdir=/System/Library/Apache/Icons \
    --includedir=/System/Library/Frameworks/Apache.framework/Versions/1.3/Headers \
    --enable-shared=max \
    --enable-module=most \
    --target=apache \
    --activate-module=src/modules/php4/libphp4.a

    You may get a message telling you that libmodphp4.a is out of date.
    If so, go to the src/modules/php4 directory inside your apache
    source directory and run this command:

    ranlib libmodphp4.a

    Then go back to the root of the apache source directory and run the
    above configure command again. That'll bring the link table up to
    date.

9. make

10. make install

11. copy and rename the php.ini-dist file to your "bin" directory from your
    PHP 4 source directory:
    cp php.ini-dist /usr/local/bin/php.ini

    or (if your don't have a local directory)

    cp php.ini-dist /usr/bin/php.ini
```

Compiling for MacOS X client

Those tips are graciously provided by Marc Liyanage.

The PHP module for the Apache web server included in Mac OS X. This version includes support for the MySQL and PostgreSQL databases.

NOTE: Be careful when you do this, you could screw up your Apache web server!

Do this to install:

- 1. Open a terminal window
- 2. Type "wget http://www.diax.ch/users/liyanage/software/macosx/libphp4.so.gz", wait for download to finish
- 3. Type "gunzip libphp4.so.gz"
- 4. Type "sudo apxs -i -a -n php4 libphp4.so"

Now type "`sudo open -a TextEdit /etc/httpd/httpd.conf`" TextEdit will open with the web server configuration file. Locate these two lines towards the end of the file: (Use the Find command)

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

Remove the two hash marks (#), then save the file and quit TextEdit.

Finally, type "`sudo apachectl graceful`" to restart the web server.

PHP should now be up and running. You can test it by dropping a file into your "Sites" folder which is called "test.php". Into that file, write this line: "`<?php phpinfo() ?>`".

Now open up `127.0.0.1/~your_username/test.php` in your web browser. You should see a status table with information about the PHP module.

Unix/OpenBSD installs

This section contains notes and hints specific to installing PHP on OpenBSD 3.2.

Using Binary Packages

Using binary packages to install PHP on OpenBSD is the recommended and simplest method. The core package has been separated from the various modules, and each can be installed and removed independently from the others. The files you need can be found on your OpenBSD CD or on the FTP site.

The main package you need to install is `php4-core-4.2.3.tgz`, which contains the basic engine (plus gettext and iconv). Next, take a look at the module packages, such as `php4-mysql-4.2.3.tgz` or `php4-imap-4.2.3.tgz`. You need to use the **phpxs** command to activate and deactivate these modules in your `php.ini` file.

Example 3-3. OpenBSD Package Install Example

```
# pkg_add php4-core-4.2.3.tgz
# /usr/local/sbin/phpxs -s
# cp /usr/local/share/doc/php4/php.ini-recommended /var/www/conf/php.ini
  (add in mysql)
# pkg_add php4-mysql-4.2.3.tgz
# /usr/local/sbin/phpxs -a mysql
  (add in imap)
# pkg_add php4-imap-4.2.3.tgz
# /usr/local/sbin/phpxs -a imap
  (remove mysql as a test)
# pkg_delete php4-mysql-4.2.3
# /usr/local/sbin/phpxs -r mysql
  (install the PEAR libraries)
# pkg_add php4-pear-4.2.3.tgz
```

Read the packages(7) manual page for more information about binary packages on OpenBSD.

Using Ports

You can also compile up PHP from source using the ports tree. However, this is only recommended for users familiar with OpenBSD. The PHP4 port is split into three sub-directories: core, extensions and pear. The extensions directory generates sub-packages for all of the supported PHP modules. If you find you do not want to create some of these modules, use the **no_*** FLAVOR. For example, to skip building the imap module, set the FLAVOR to **no_imap**.

Older Releases

Older releases of OpenBSD used the FLAVORS system to compile up a statically linked PHP. Since it is hard to generate binary packages using this method, it is now deprecated. You can still use the old stable ports trees if you wish, but they are unsupported by the OpenBSD team. If you have any comments about this, the current maintainer for the port is Anil Madhavapeddy.

Unix/Solaris installs

This section contains notes and hints specific to installing PHP on Solaris systems.

Required software

Solaris installs often lack C compilers and their related tools. The required software is as follows:

- gcc (recommended, other C compilers may work)
- make
- flex
- bison
- m4
- autoconf
- automake
- perl
- gzip
- tar
- GNU sed

In addition, you will need to install (and possibly compile) any additional software specific to your configuration, such as Oracle or MySQL.

Using Packages

You can simplify the Solaris install process by using pkgadd to install most of your needed components.

Installation on UNIX systems

This section will guide you through the general configuration and installation of PHP on Unix systems. Be sure to investigate any sections specific to your platform or web server before you begin the process.

Prerequisite knowledge and software:

- Basic UNIX skills (being able to operate "make" and a C compiler, if compiling)
- An ANSI C compiler (if compiling)
- flex (for compiling)
- bison (for compiling)
- A web server
- Any module specific components (such as gd, pdf libs, etc.)

There are several ways to install PHP for the Unix platform, either with a compile and configure process, or through various pre-packaged methods. This documentation is mainly focused around the process of compiling and configuring PHP.

The initial PHP setup and configuration process is controlled by the use of the commandline options of the `configure` script. This page outlines the usage of the most common options, but there are many others to play with. Check out the Complete list of configure options for an exhaustive rundown. There are several ways to install PHP:

- As an Apache module
- As an fhttpd module
- For use with AOLServer, NSAPI, phttpd, Pi3Web, Roxen, thttpd, or Zeus.
- As a CGI executable

Apache Module Quick Reference

PHP can be compiled in a number of different ways, but one of the most popular is as an Apache module. The following is a quick installation overview.

Example 3-4. Quick Installation Instructions for PHP 4 (Apache Module Version)

```
1.  gunzip apache_1.3.x.tar.gz
2.  tar xvf apache_1.3.x.tar
3.  gunzip php-x.x.x.tar.gz
4.  tar xvf php-x.x.x.tar
5.  cd apache_1.3.x
6.  ./configure --prefix=/www
7.  cd ../php-x.x.x
8.  ./configure --with-mysql --with-apache=../apache_1.3.x --enable-track-vars
9.  make
10. make install
11. cd ../apache_1.3.x
12. ./configure --activate-module=src/modules/php4/libphp4.a
13. make
14. make install
15. cd ../php-x.x.x
16. cp php.ini-dist /usr/local/lib/php.ini
17. Edit your httpd.conf or srm.conf file and add:
      AddType application/x-httpd-php .php

18. Use your normal procedure for restarting the Apache server. (You must
    stop and restart the server, not just cause the server to reload by
    use a HUP or USR1 signal.)
```

Building

When PHP is configured, you are ready to build the CGI executable. The command **make** should take care of this. If it fails and you can't figure out why, see the Problems section.

Installation on Windows systems

This section applies to Windows 95/98/Me and Windows NT/2000/XP. Do not expect PHP to work on 16 bit platforms such as Windows 3.1. Sometimes we refer to the supported Windows platforms as Win32.

There are two main ways to install PHP for Windows: either manually or by using the InstallShield 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.

Windows InstallShield

The Windows PHP installer 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, configures the web server as well.

> **Note:** Also note, that while the InstallShield installer is an easy way to make PHP work, it is restricted in many aspects, as automatic setup of extensions for example is not supported. The whole set of supported extensions is only available by downloading the zip binary distribution.

Install your selected HTTP 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 the web server to use PHP. For IIS and also PWS on NT Workstation, a list of all the nodes on the server with script map settings is displayed, and you can choose those nodes to which you wish to add the PHP script mappings.

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 on your Windows webserver. You need to download the zip binary distribution from the downloads page at http://www.php.net/downloads.php. The original version of this guide was compiled by Bob Silva, and can be found at http://www.umesd.k12.or.us/php/win32install.html.

This guide provides manual installation support for:

- Personal Web Server 3 and 4 or newer
- Internet Information Server 3 and 4 or newer
- Apache 1.3.x
- OmniHTTPd 2.0b1 and up
- Oreilly Website Pro
- Xitami
- Netscape Enterprise Server, iPlanet

PHP 4 for Windows comes in two flavours - a CGI executable (php.exe), and several SAPI modules (for example: php4isapi.dll). The latter form is new to PHP 4, and provides significantly improved performance and some new functionality. There is also a CLI version which is further described in the commandline chapter.

Warning
The SAPI modules have been significantly improved in the 4.1 release, however, you may find that you encounter possible server errors or other server modules such as ASP failing, in older systems.

If you choose one of the SAPI modules and use Windows 95, be sure to download the DCOM update from the Microsoft DCOM pages. For the ISAPI module, an ISAPI 4.0 compliant Web server is required (tested on IIS 4.0, PWS 4.0 and IIS 5.0). IIS 3.0 is *NOT* supported. You should download and install the Windows NT 4.0 Option Pack with IIS 4.0 if you want native PHP support.

The following steps should be performed on all installations before the server specific instructions.

- Extract the distribution file to a directory of your choice. `c:\php\` is a good start. You probably do not want to use a path in which spaces are included (for example: c:\program files\php is not a good idea). Some web servers will crash if you do.
- You need to ensure that the DLLs which PHP uses can be found. The precise DLLs involved depend on which web server you use and whether you want to run PHP as a CGI or as a server module. `php4ts.dll` is always used. If you are using a server module (e.g. ISAPI or Apache) then you will need the relevant DLL from the `sapi` folder. If you are using any PHP extension DLLs then you will need those as well. To make sure that the DLLs can be found, you can either copy them to the system directory (e.g. `winnt/system32` or `windows/system`) or you can make sure that they live in the same directory as the main PHP executable or DLL your web server will use (e.g. `php.exe`, `php4apache.dll`).

  The PHP binary, the SAPI modules, and some extensions rely on external DLLs for execution. Make sure that these DLLs in the distribution exist in a directory that is in the Windows PATH. For example, if you enable `php_oci8.dll` in `php.ini` then you'll want to make sure the Oracle home directory can be seen in PATH so PHP can find `oci.dll`.

  The best bet to do it is to copy the files below into your system directory, which is typically:

  `c:\windows\system` for Windows 9x/ME
  `c:\winnt\system32` for Windows NT/2000
  `c:\windows\system32` for Windows XP

  The files to copy are:

  `php4ts.dll`, if it already exists there, overwrite it
  The files in your distribution's 'dlls' directory. If you have them already installed on your system, overwrite them only if something doesn't work correctly (Before overwriting them, it is a good idea to make a backup of them, or move them to another folder - just in case something goes wrong).

  Download the latest version of the Microsoft Data Access Components (MDAC) for your platform, especially if you use

Microsoft Windows 9x/NT4. MDAC is available at http://www.microsoft.com/data/.

- Copy your chosen ini file (see below) to your '%WINDOWS%' directory on Windows 9x/Me or to your '%SYSTEMROOT%' directory under Windows NT/2000/XP and rename it to `php.ini`. Your '%WINDOWS%' or '%SYSTEMROOT%' directory is typically:

  `c:\windows` for Windows 9x/ME/XP
  `c:\winnt` or `c:\winnt40` for NT/2000 servers

  There are two ini files distributed in the zip file, `php.ini-dist` and `php.ini-optimized`. We advise you to use `php.ini-optimized`, because we optimized the default settings in this file for performance, and security. The best is to study all 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.

- Edit your new `php.ini` file:
  - You will need to change the 'extension_dir' setting to point to your php-install-dir, or where you have placed your `php_*.dll` files. Please do not forget the last backslash. ex: `c:\php\extensions\`
  - If you are using OmniHTTPd, do not follow the next step. Set the 'doc_root' to point to your webservers document_root. For example: `c:\apache\htdocs` or `c:\webroot`
  - Choose which 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.ini` 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.
  - Note that the `mibs` directory supplied with the Windows distribution contains support files for SNMP. This directory should be moved to `DRIVE:\usr\mibs` (`DRIVE` being the drive where PHP is installed.)
  - If you're using NTFS on Windows NT, 2000 or XP, make sure that the user running the webserver has read permissions to your `php.ini` (e.g. make it readable by Everyone).
- For PWS give execution permission to the webroot:
  - Start PWS Web Manager
  - Edit Properties of the "Home"-Directory
  - Select the "execute"-Checkbox

Building from source

Before getting started, it is worthwhile answering the question: "Why is building on Windows so hard?" Two reasons come to mind:

1. Windows does not (yet) enjoy a large community of developers who are willing to freely share their source. As a direct result, the necessary investment in infrastructure required to support such development hasn't been made. By and large, what is available has been made possible by the porting of necessary utilities from Unix. Don't be surprised if some of this heritage shows through from time to time.
2. Pretty much all of the instructions that follow are of the "set and forget" variety. So sit back and try follow the instructions below as faithfully as you can.

Requirements

To compile and build PHP you need a Microsoft Development Environment. Microsoft Visuaul C++ 6.0 is recommended. To extract the downloaded files you need a extraction utilitiy (e.g.: Winzip). If you don't already have an unzip utility, you can get a free version from InfoZip.

Before you get started, you have to download...

- ..the win32 buildtools from the PHP site at http://www.php.net/extra/win32build.zip.
- ..the source code for the DNS name resolver used by PHP from http://www.php.net/extra/bindlib_w32.zip. This is a replacement for the `resolv.lib` library included in `win32build.zip`.

- If you plan to compile PHP as a Apache module you will also need the Apache sources.

Finally, you are going to need the source to PHP 4 itself. You can get the latest development version using anonymous CVS, a snapshot or the most recent released source tarball.

Putting it all together

After downloading the required packages you have to extract them in a proper place.

- Create a working directory where all files end up after extracting, e.g: `c:\work`.
- Create the directory `win32build` under your working directory (`c:\work`) and unzip `win32build.zip` into it.
- Create the directory `bindlib_w32` under your working directory (`c:\work`) and unzip `bindlib_w32.zip` into it.
- Extract the downloaded PHP source code into your working directory (`c:\work`).

Following this steps your directory structure looks like this:

```
+--c:\work
|  |
|  +--bindlib_w32
|  |  |
|  |  +--arpa
|  |  |
|  |  +--conf
|  |  |
|  |  +--...
|  |
|  +--php-4.x.x
|  |  |
|  |  +--build
|  |  |
|  |  +--...
|  |  |
|  |  +--win32
|  |  |
|  |  +--...
|  |
|  +--win32build
|  |  |
|  |  +--bin
|  |  |
|  |  +--include
|  |  |
|  |  +--lib
```

Create the directories `c:\usr\local\lib`. Copy `bison.simple` from `c:\work\win32build\bin` to `c:\usr\local\lib`.

> **Note:** Cygwin users may omit the last step. A properly installed Cygwin environment provides the mandatory files `bison.simple` and `bison.exe`.

Configure MVC ++

The next step is to configure MVC ++ to prepare for compiling. Launch Microsoft Visual C++, and from the menu select Tools => Options. In the dialog, select the directories tab. Sequentially change the dropdown to Executables, Includes, and Library files. Your entries should look like this:

- Executable files: `c:\work\win32build\bin`, Cygwin users: `cygwin\bin`
- Include files: `c:\work\win32build\include`
- Library files: `c:\work\win32build\lib`

Build resolv.lib

You must build the `resolv.lib` library. Decide whether you want to have debug symbols available (bindlib - Win32 Debug) or not (bindlib - Win32 Release). Build the appropriate configuration:

- For GUI users, launch VC++, and then select File => Open Workspace, navigate to `c:\work\bindlib_w32`and select `bindlib.dsw`. Then select Build=>Set Active Configuration and select the desired configuration. Finally select Build=>Rebuild All.

- For command line users, make sure that you either have the C++ environment variables registered, or have run **vcvars.bat**, and then execute one of the following commands:
  - **msdev bindlib.dsp /MAKE "bindlib - Win32 Debug"**
  - **msdev bindlib.dsp /MAKE "bindlib - Win32 Release"**

At this point, you should have a usable `resolv.lib` in either your `c:\work\bindlib_w32\Debug` or `Release` subdirectories. Copy this file into your `c:\work\win32build\lib` directory over the file by the same name found in there.

Compiling

The best way to get started is to build the CGI version.

- For GUI users, launch VC++, and then select File => Open Workspace and select `c:\work\php-4.x.x\win32\php4ts.dsw` . Then select Build=>Set Active Configuration and select the desired configuration, either `php4ts - Win32 Debug_TS` or `php4ts - Win32 Release_TS`. Finally select Build=>Rebuild All.
- For command line users, make sure that you either have the C++ environment variables registered, or have run **vcvars.bat**, and then execute one of the following commands from the `c:\work\php-4.x.x\win32` directory:
  - **msdev php4ts.dsp /MAKE "php4ts - Win32 Debug_TS"**
  - **msdev php4ts.dsp /MAKE "php4ts - Win32 Release_TS"**
  - At this point, you should have a usable `php.exe` in either your `c:\work\php-4.x.x.\Debug_TS` or `Release_TS` subdirectories.

It is possible to do minor customization to the build process by editing the `main/config.win32.h.in` file. For example you can change the builtin extensions, the location of `php.ini` and

Next you may want to build the CLI version which is designed to use PHP from the command line. The steps are the same as for building the CGI version, except you have to select the `php4ts_cli - Win32 Debug_TS` or `php4ts_cli - Win32 Release_TS` project file. After a succcessfull compiling run you will find the `php.exe` in either the directory `Release_TS\cli\` or `Debug_TS\cli\`.

> **Note:** If you want to use PEAR and the comfortable command line installer, the CLI-SAPI is mandatory. For more information about PEAR and the installer read the documantation at the PEAR website.

In order to build the SAPI module (`php4isapi.dll` for integrating PHP with Microsoft IIS, set your active configuration to `php4isapi-whatever-config` and build the desired dll.

Installation of Windows extensions

After installing PHP and a webserver on Windows, you will probably want to install some extensions for added functionality. The following table describes some of the extensions available. You can choose which extensions you would like to load when PHP starts by uncommenting the: 'extension=php_*.dll' lines in `php.ini`. You can also load a module dynamically in your script using **dl()**.

The DLLs for PHP extensions are prefixed with 'php_' in PHP 4 (and 'php3_' in PHP 3). This prevents confusion between PHP extensions and their supporting libraries.

> **Note:** In PHP 4.0.6 BCMath, Calendar, COM, FTP, MySQL, ODBC, PCRE, Session, WDDX and XML support is *built in*. You don't need to load any additional extensions in order to use these functions. See your distributions `README.txt` or `install.txt` for a list of built in modules.
>
> **Note:** Some of these extensions need extra DLLs to work. Couple of them can be found in the distribution package, in the 'dlls' folder but some, for example Oracle (php_oci8.dll) require DLLs which are not bundled with the distribution package.
>
> Copy the bundled DLLs from 'DLLs' folder to your Windows PATH, safe places are:
>
> c:\windows\system for Windows 9x/Me
> c:\winnt\system32 for Windows NT/2000
> c:\windows\system32 for Windows XP
>
> If you have them already installed on your system, overwrite them only if something doesn't work correctly (Before overwriting them, it is a good idea to make a backup of them, or move them to another folder - just in case something goes wrong).

Table 3-1. PHP Extensions

Extension	Description	Notes
php_bz2.dll	bzip2 compression functions	None
php_calendar.dll	Calendar conversion functions	Built in since PHP 4.0.3
php_cpdf.dll	ClibPDF functions	None
php_crack.dll	Crack functions	None
php3_crypt.dll	Crypt functions	unknown
php_ctype.dll	ctype family functions	None
php_curl.dll	CURL, Client URL library functions	Requires: libeay32.dll, ssleay32.dll (bundled)
php_cybercash.dll	Cybercash payment functions	None
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
php3_dbm.dll	Berkeley DB2 library	unknown
php_domxml.dll	DOM XML functions	Requires: libxml2.dll (bundled)
php_dotnet.dll	.NET functions	None
php_exif.dll	Read EXIF headers from JPEG	None
php_fbsql.dll	FrontBase functions	None
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	Built-in since PHP 4.0.3
php_gd.dll	GD library image functions	None
php_gd2.dll	GD2 library image functions	None
php_gettext.dll	Gettext functions	Requires: gnu_gettext.dll (bundled)
php_hyperwave.dll	HyperWave functions	None
php_iconv.dll	ICONV characterset conversion	Requires: iconv-1.3.dll (bundled)
php_ifx.dll	Informix functions	Requires: Informix libraries
php_iisfunc.dll	IIS management functions	None
php_imap.dll	IMAP POP3 and NNTP functions	PHP 3: php3_imap4r1.dll
php_ingres.dll	Ingres II functions	Requires: Ingres II libraries
php_interbase.dll	InterBase functions	Requires: gds32.dll (bundled)
php_java.dll	Java extension	Requires: jvm.dll (bundled)
php_ldap.dll	LDAP functions	Requires: libsasl.dll (bundled)
php_mhash.dll	Mhash Functions	None
php_ming.dll	Ming functions for Flash	None
php_msql.dll	mSQL functions	Requires: msql.dll (bundled)
php3_msql1.dll	mSQL 1 client	unknown
php3_msql2.dll	mSQL 2 client	unknown
php_mssql.dll	MSSQL functions	Requires: ntwdblib.dll (bundled)
php3_mysql.dll	MySQL functions	Built-in in PHP 4
php3_nsmail.dll	Netscape mail functions	unknown
php3_oci73.dll	Oracle functions	unknown
php_oci8.dll	Oracle 8 functions	Requires: Oracle 8 client libraries
php_openssl.dll	OpenSSL functions	Requires: libeay32.dll (bundled)
php_oracle.dll	Oracle functions	Requires: Oracle 7 client libraries
php_pdf.dll	PDF functions	None
php_pgsql.dll	PostgreSQL functions	None
php_printer.dll	Printer functions	None
php_xslt.dll	XSLT functions	Requires: sablot.dll (bundled)
php_snmp.dll	SNMP get and walk functions	NT only!
php_sybase_ct.dll	Sybase functions	Requires: Sybase client libraries

Extension	Description	Notes
php_yaz.dll	YAZ functions	None
php_zlib.dll	ZLib compression functions	None

Servers-CGI/Commandline

The default is to build PHP as a CGI program. This creates a commandline interpreter, which can be used for CGI processing, or for non-web-related PHP scripting. If you are running a web server PHP has module support for, you should generally go for that solution for performance reasons. However, the CGI version enables Apache users to run different PHP-enabled pages under different user-ids. Please make sure you read through the Security chapter if you are going to run PHP as a CGI.

As of PHP 4.3.0, some important additions have happened to PHP. A new SAPI named CLI also exists and it has the same name as the CGI binary. What is installed at `{PREFIX}/bin/php` depends on your configure line and this is described in detail in the manual section named Using PHP from the command line. For further details please read that section of the manual.

Testing

If you have built PHP as a CGI program, you may test your build by typing **make test**. It is always a good idea to test your build. This way you may catch a problem with PHP on your platform early instead of having to struggle with it later.

Benchmarking

If you have built PHP 3 as a CGI program, you may benchmark your build by typing **make bench**. Note that if Safe Mode is on by default, the benchmark may not be able to finish if it takes longer then the 30 seconds allowed. This is because the **set_time_limit()** can not be used in safe mode. Use the max_execution_time configuration setting to control this time for your own scripts. **make bench** ignores the configuration file.

> **Note: make bench** is only available for PHP 3.

Using Variables

Some server supplied enviroment variables are not defined in the current CGI/1.1 specification. Only the following variables are defined there; everything else should be treated as 'vendor extensions': AUTH_TYPE, CONTENT_LENGTH, CONTENT_TYPE, GATEWAY_INTERFACE, PATH_INFO, PATH_TRANSLATED, QUERY_STRING, REMOTE_ADDR, REMOTE_HOST, REMOTE_IDENT, REMOTE_USER, REQUEST_METHOD, SCRIPT_NAME, SERVER_NAME, SERVER_PORT, SERVER_PROTOCOL and SERVER_SOFTWARE

Servers-Apache

This section contains notes and hints specific to Apache installs of PHP, both for Unix and Windows versions. We also have instructions and notes for Apache 2 on a separate page.

Details of installing PHP with Apache on Unix

You can select arguments to add to the **configure** on line 10 below from the Complete list of configure options. The version numbers have been omitted here, to ensure the instructions are not incorrect. You will need to replace the 'xxx' here with the correct values from your files.

Example 3-5. Installation Instructions (Apache Shared Module Version) for PHP 4

```
1.  gunzip apache_xxx.tar.gz
2.  tar -xvf apache_xxx.tar
3.  gunzip php-xxx.tar.gz
4.  tar -xvf php-xxx.tar
5.  cd apache_xxx
6.  ./configure --prefix=/www --enable-module=so
7.  make
8.  make install
```

```
9.  cd ../php-xxx
10. ./configure --with-mysql --with-apxs=/www/bin/apxs
11. make
12. make install

  If you decide to change your configure options after installation
  you only need to repeat the last three steps. You only need to
  restart apache for the new module to take effect. A recompile of
  Apache is not needed.

13. cp php.ini-dist /usr/local/lib/php.ini

  You can edit your .ini file to set PHP options.  If
  you prefer this file in another location, use
  --with-config-file-path=/path in step 10.

14. Edit your httpd.conf or srm.conf file and check that these lines are
    present and not commented out:

   AddType application/x-httpd-php .php

   LoadModule php4_module        libexec/libphp4.so

  You can choose any extension you wish here.  .php is simply the one
  we suggest. You can even include .html, and .php3 can be added for
  backwards compatibility.

  The path on the right hand side of the LoadModule statement must point
  to the path of the PHP module on your system. The above statement is
  correct for the steps shown above.


15. Use your normal procedure for starting the Apache server. (You must
    stop and restart the server, not just cause the server to reload by
    use a HUP or USR1 signal.)
```

Depending on your Apache install and Unix variant, there are many possible ways to stop and restart the server. Below are some typical lines used in restarting the server, for different apache/unix installations. You should replace `/path/to/` with the path to these applications on your systems.

```
1. Several Linux and SysV variants:
/etc/rc.d/init.d/httpd restart

2. Using apachectl scripts:
/path/to/apachectl stop
/path/to/apachectl start

3. httpdctl and httpsdctl (Using OpenSSL), similar to apachectl:
/path/to/httpsdctl stop
/path/to/httpsdctl start

4. Using mod_ssl, or another SSL server, you may want to manually
stop and start:
/path/to/apachectl stop
/path/to/apachectl startssl
```

The locations of the apachectl and http(s)dctl binaries often vary. If your system has `locate` or `whereis` or `which` commands, these can assist you in finding your server control programs.

Different examples of compiling PHP for apache are as follows:

```
./configure --with-apxs --with-pgsql
```

This will create a `libphp4.so` shared library that is loaded into Apache using a LoadModule line in Apache's `httpd.conf` file. The PostgreSQL support is embedded into this `libphp4.so` library.

```
./configure --with-apxs --with-pgsql=shared
```

This will create a `libphp4.so` shared library for Apache, but it will also create a `pgsql.so` shared library that is loaded into PHP either by using the extension directive in `php.ini` file or by loading it explicitly in a script using the **dl()** function.

```
./configure --with-apache=/path/to/apache_source --with-pgsql
```

This will create a `libmodphp4.a` library, a `mod_php4.c` and some accompanying files and copy this into the `src/modules/php4` directory in the Apache source tree. Then you compile Apache using `--activate-module=src/modules/php4/libphp4.a` and the Apache build system will create `libphp4.a` and link it statically into the `httpd` binary. The PostgreSQL support is included directly into this `httpd` binary, so the final result here is a single `httpd` binary that includes all of Apache and all of PHP.

```
./configure --with-apache=/path/to/apache_source --with-pgsql=shared
```

Same as before, except instead of including PostgreSQL support directly into the final `httpd` you will get a `pgsql.so` shared library that you can load into PHP from either the `php.ini` file or directly using **dl()**.

When choosing to build PHP in different ways, you should consider the advantages and drawbacks of each method. Building as a shared object will mean that you can compile apache separately, and don't have to recompile everything as you add to, or change, PHP. Building PHP into apache (static method) means that PHP will load and run faster. For more information, see the Apache webpage on DSO support.

> **Note:** Apache's default http.conf currently ships with a section that looks like this:
>
> ```
> User nobody
> Group "#-1"
> ```
>
> Unless you change that to "Group nogroup" or something like that ("Group daemon" is also very common) PHP will not be able to open files.
>
> **Note:** Make sure you specify the installed version of apxs when using --with-apxs=/path/to/apxs. You must NOT use the apxs version that is in the apache sources but the one that is actually installed on your system.

Installing PHP on Windows with Apache 1.3.x

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), the other is to use the Apache module DLL. In either case you need to stop the Apache server, and edit your `srm.conf` or `httpd.conf` to configure Apache to work with PHP.

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 Docs for further configuration directives.

If you unziped 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:

- `ScriptAlias /php/ "c:/php/"`
- `AddType application/x-httpd-php .php .phtml`
- `Action application/x-httpd-php "/php/php.exe"`

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 attacks.

If you would like to use PHP as a module in Apache, be sure to move `php4ts.dll` to the windows/system (for Windows 9x/Me) or winnt/system32 (for Windows NT/2000/XP) directory, overwriting any older file. Then you should add the following two lines to you Apache conf file:

- `LoadModule php4_module c:/php/sapi/php4apache.dll`
- `AddType application/x-httpd-php .php .phtml`

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:** You may find after using the windows installer for Apache that you need to define the `AddModule` directive for `mod_php4.c` in the configuration file (`httpd.conf`). This is done by adding `AddModule mod_php4.c` to the `AddModule` list, near the beginning of the configuration file. This is especially important if the `ClearModuleList` directive is defined. Failure to do this may mean PHP will not be registered as an Apache module.

There are 2 ways you can use the source code highlighting feature, however their ability to work depends on your installation. If you have configured Apache to use PHP as an ISAPI module, then by adding the following line to your configuration file you can use this feature: `AddType application/x-httpd-php-source .phps`

If you chose to configure Apache to use PHP as a CGI binary, you will need to use the **show_source()** function. To do this simply create a PHP script file and add this code: `<?php show_source ("original_php_script.php"); ?>`. Substitute `original_php_script.php` with the name of the file you wish to show the source of.

> **Note:** On Win-Apache all backslashes in a path statement such as "c:\directory\file.ext", must be converted to forward slashes, as "c:/directory/file.ext".

Servers-Apache 2.0

This section contains notes and hints specific to Apache 2.0 installs of PHP, both for Unix and Windows versions.

Warning
Do not use Apache 2.0 and PHP in a production environment neither on Unix nor on Windows.

You are highly encouraged to take a look at the Apache Documentation to get a basic understanding of the Apache 2.0 Server.

PHP and Apache 2.0 compatibility notes

The following versions of PHP are known to work with the most recent version of Apache 2.0:

- 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/php4-latest.tar.gz or download binaries for windows http://snaps.php.net/win32/php4-win32-latest.zip.
- a prerelease version downloadable from http://qa.php.net/.
- you have always the option to obtain PHP through anonymous CVS.

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

> **Note:** Apache 2.0 SAPI-support started with PHP 4.2.0. PHP 4.2.3 its known to work in conjunction with Apache 2.0.39. Don't try to use this version of PHP with any other version of Apache. We do not recommend to use PHP 4.2.3 along with Apache 2.0.39.
>
> All mentioned versions of PHP will work still with Apache 1.3.x.

PHP and Apache 2 on Linux

Download the most recent version of Apache 2.0 and a fitting PHP version from the above mentioned places. This quick guide covers only the basics to get started with Apache 2.0 and PHP. For more information read the Apache Documentation. The version numbers have been omitted here, to ensure the instructions are not incorrect. You will need to replace the 'NN' here with the correct values from your files.

Example 3-6. Installation Instructions (Apache 2 Shared Module Version)

```
 1.  gzip -d httpd-2_0_NN.tar.gz
 2.  tar xvf httpd-2_0_NN.tar
 3.  gunzip php-NN.tar.gz
 4.  tar -xvf php-NN.tar
 5.  cd httpd-2_0_NN
 6.  ./configure --enable-so
 7.  make
 8.  make install

     Now you have Apache 2.0.NN available under /usr/local/apache2,
     configured with loadable module support and the standard MPM prefork.
     To test the installation use your normal procedure for starting
     the Apache server, e.g.:
     /usr/local/apache2/bin/apachectl start
     and stop the server to go on with the configuration for PHP:
     /usr/local/apache2/bin/apachectl stop.

 9.  cd ../php4-NN
 10. ./configure --with-apxs2=/usr/local/apache2/bin/apxs
 11. make
 12. make install
 13. cp php.ini-dist /usr/local/lib/php.ini

     Edit your php.ini file to set PHP options. If
     you prefer this file in another location, use
     --with-config-file-path=/path in step 10.

 14. Edit your httpd.conf file and check that these lines are
     present:

    LoadModule php4_module modules/libphp4.so
    AddType application/x-httpd-php .php

   You can choose any extension you wish here. .php is simply the one
   we suggest.
```

```
   The path on the right hand side of the LoadModule statement must point
   to the path of the PHP module on your system. The above statement is
   correct for the steps shown above.

15. Use your normal procedure for starting the Apache server, e.g.:
    /usr/local/apache2/bin/apachectl start
```

Following the steps above you will have a running Apache 2.0 with support for PHP as SAPI module. Of course there are many more configuration options available for both, Apache and PHP. For more information use **./configure --help** in the corresponding source tree. In case you wish to build a multithreaded version of Apache 2.0 you must overwrite the standard MPM-Module `prefork` either with `worker` or `perchild`. To do so append to your configure line in step 6 above either the option `--with-mpm=worker` or `--with-mpm=perchild`. Take care about the consequences and understand what you are doing. For more information read the Apache documentation about the MPM-Modules.

> **Note:** To build a multithreaded version of Apache your system must support threads. This also implies to build PHP with experimental Zend Thread Safety (ZTS). Therefore not all extensions might be available. The recommended setup is to build Apache with the standard `prefork` MPM-Module.

PHP and Apache 2.0 on Windows

Consider to read the Windows specific notes for Apache 2.0.

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

Download the most recent version of Apache 2.0 and a fitting PHP version from the above mentioned places. 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 on Windows. One is to use the CGI binary the other is to use the Apache module DLL. In either case you need to stop the Apache server, and edit your `httpd.conf` to configure Apache to work with PHP.

You need to insert these three lines to your Apache `httpd.conf` configuration file to set up the *CGI binary*:

Example 3-7. PHP and Apache 2.0 as CGI

```
ScriptAlias /php/ "c:/php/"
AddType application/x-httpd-php .php
Action application/x-httpd-php "/php/php.exe"
```

If you would like to use PHP as a module in Apache 2.0, be sure to move `php4ts.dll` to `winnt/system32` (for Windows NT/2000) or `windows/system32` (for Windows XP), overwriting any older file. 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 3-8. PHP and Apache 2.0 as Module

```
LoadModule php4_module c:/php/sapi/php4apache2.dll
AddType application/x-httpd-php .php
```

> **Note:** Remember to substitute the `c:/php/` for your actual path to PHP in the above examples.

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.

Servers-Caudium

PHP 4 can be built as a Pike module for the Caudium webserver. Note that this is not supported with PHP 3. Follow the simple instructions below to install PHP 4 for Caudium.

Example 3-9. Caudium Installation Instructions

```
1.  Make sure you have Caudium installed prior to attempting to
    install PHP 4. For PHP 4 to work correctly, you will need Pike
    7.0.268 or newer. For the sake of this example we assume that
    Caudium is installed in /opt/caudium/server/.
2.  Change directory to php-x.y.z (where x.y.z is the version number).
```

```
3.  ./configure --with-caudium=/opt/caudium/server
4.  make
5.  make install
6.  Restart Caudium if it's currently running.
7.  Log into the graphical configuration interface and go to the
    virtual server where you want to add PHP 4 support.
8.  Click Add Module and locate and then add the PHP 4 Script Support module.
9.  If the documentation says that the 'PHP 4 interpreter isn't
    available', make sure that you restarted the server. If you did
    check /opt/caudium/logs/debug/default.1 for any errors related to
    <filename>PHP4.so</filename>. Also make sure that
    <filename>caudium/server/lib/[pike-version]/PHP4.so</filename>
    is present.
10. Configure the PHP Script Support module if needed.
```

You can of course compile your Caudium module with support for the various extensions available in PHP 4. See the complete list of configure options for an exhaustive rundown.

> **Note:** When compiling PHP 4 with MySQL support you must make sure that the normal MySQL client code is used. Otherwise there might be conflicts if your Pike already has MySQL support. You do this by specifying a MySQL install directory the --with-mysql option.

Servers-fhttpd

To build PHP as an fhttpd module, answer "yes" to "Build as an fhttpd module?" (the --with-fhttpd=*DIR* option to configure) and specify the fhttpd source base directory. The default directory is `/usr/local/src/fhttpd`. If you are running fhttpd, building PHP as a module will give better performance, more control and remote execution capability.

Servers-IIS/PWS

This section contains notes and hints specific to IIS (Microsoft Internet Information Server). Installing PHP for PWS/IIS 3, PWS 4 or newer and IIS 4 or newer versions.

> **Important for CGI users:** Read the faq on cgi.force_redirect for important details. This directive needs to be set to `0`.

Windows and PWS/IIS 3

The recommended method for configuring these servers is to use the REG file included with the distribution (pws-php4cgi.reg). You may want to edit this file and make sure the extensions and PHP install directories match your configuration. Or you can follow the steps below to do it manually.

Warning
These steps involve working directly with the Windows registry. One error here can leave your system in an unstable state. We highly recommend that you back up your registry first. The PHP Development team will not be held responsible if you damage your registry.

- Run Regedit.
- Navigate to: `HKEY_LOCAL_MACHINE /System /CurrentControlSet /Services /W3Svc /Parameters /ScriptMap`.
- On the edit menu select: `New->String Value`.
- Type in the extension you wish to use for your php scripts. For example `.php`
- Double click on the new string value and enter the path to `php.exe` in the value data field. ex: `c:\php\php.exe`.
- Repeat these steps for each extension you wish to associate with PHP scripts.

The following steps do not affect the web server installation and only apply if you want your php scripts to be executed when they are run from the command line (ex. run `c:\myscripts\test.php`) or by double clicking on them in a directory viewer window. You may wish to skip these steps as you might prefer the PHP files to load into a text editor when you double click on them.

- Navigate to: `HKEY_CLASSES_ROOT`
- On the edit menu select: `New->Key`.

- Name the key to the extension you setup in the previous section. ex: `.php`
- Highlight the new key and in the right side pane, double click the "default value" and enter `phpfile`.
- Repeat the last step for each extension you set up in the previous section.
- Now create another `New->Key` under `HKEY_CLASSES_ROOT` and name it `phpfile`.
- Highlight the new key `phpfile` and in the right side pane, double click the "default value" and enter `PHP Script`.
- Right click on the `phpfile` key and select `New->Key`, name it `Shell`.
- Right click on the `Shell` key and select `New->Key`, name it `open`.
- Right click on the `open` key and select `New->Key`, name it `command`.
- Highlight the new key `command` and in the right side pane, double click the "default value" and enter the path to `php.exe`. ex: `c:\php\php.exe -q %1`. (don't forget the `%1`).
- Exit Regedit.
- If using PWS on Windows, reboot to reload the registry.

PWS and IIS 3 users now have a fully operational system. IIS 3 users can use a nifty tool from Steven Genusa to configure their script maps.

Windows and PWS 4 or newer

When installing PHP on Windows with PWS 4 or newer version, you have two options. One to set up the PHP CGI binary, the other is to use the ISAPI module DLL.

If you choose the CGI binary, do the following:

- Edit the enclosed `pws-php4cgi.reg` file (look into the SAPI dir) to reflect the location of your `php.exe`. Backslashes should be escaped, for example: `[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\w3svc\parameters\Script Map] ".php"="c:\\php\\php.exe"` Now merge this registery file into your system; you may do this by double-clicking it.
- In the PWS Manager, right click on a given directory you want to add PHP support to, and select Properties. Check the 'Execute' checkbox, and confirm.

If you choose the ISAPI module, do the following:

- Edit the enclosed `pws-php4isapi.reg` file (look into the SAPI dir) to reflect the location of your `php4isapi.dll`. Backslashes should be escaped, for example: `[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\w3svc\parameters\Script Map] ".php"="c:\\php\\sapi\\php4isapi.dll"` Now merge this registery file into your system; you may do this by double-clicking it.
- In the PWS Manager, right click on a given directory you want to add PHP support to, and select Properties. Check the 'Execute' checkbox, and confirm.

Windows NT/2000/XP and IIS 4 or newer

To install PHP on an NT/2000/XP Server running IIS 4 or newer, follow these instructions. You have two options to set up PHP, using the CGI binary (php.exe) 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', click on the 'Configuration' button, and then enter the App Mappings tab.
- Click Add, and in the Executable box, type: `c:\php\php.exe` (assuming that you have unziped PHP in c:\php\).
- In the Extension box, type the file name extension you want associated with PHP scripts. Leave 'Method exclusions' blank, and check the Script engine checkbox. You may also like to check the 'check that file exists' box - for a small performance penalty, IIS (or PWS) will check that the script file exists and sort out authentication before firing up php. This means that

you will get sensible 404 style error messages instead of cgi errors complaining that php did not output any data.

You must start over from the previous step for each extension you want associated with PHP scripts. `.php` and `.phtml` are common, although `.php3` may be required for legacy applications.

- 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`.

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 php4isapi.dll.
- Under 'Home Directory', click on the 'Configuration' button. Add a new entry to the Application Mappings. Use the path to the php4isapi.dll as the Executable, supply `.php` as the extension, leave Method exclusions blank, and check the Script engine checkbox.
- Stop IIS completely (NET STOP iisadmin)
- Start IIS again (NET START w3svc)

Servers-Netscape and iPlanet

This section contains notes and hints specific to Netscape and iPlanet installs of PHP, both for Sun Solaris and Windows versions.

You can find more information about setting up PHP for the Netscape Enterprise Server here:
http://benoit.noss.free.fr/php/install-php4.html

Installing PHP with Netscape on Sun Solaris

To build PHP with NES or iPlanet web servers, enter the proper install directory for the --with-nsapi = *DIR* option. The default directory is usually `/opt/netscape/suitespot/`. Please also read `/php-xxx-version/sapi/nsapi/nsapi-readme.txt`.

Example 3-10. Installation Example for Netscape Enterprise on Solaris

```
Instructions for Sun Solaris 2.6 with Netscape Enterprise Server 3.6
From: bhager@invacare.com

1. Install the following packages from www.sunfreeware.com or another
download site:

    flex-2_5_4a-sol26-sparc-local
    gcc-2_95_2-sol26-sparc-local
    gzip-1.2.4-sol26-sparc-local
    perl-5_005_03-sol26-sparc-local
    bison-1_25-sol26-sparc-local
    make-3_76_1-sol26-sparc-local
    m4-1_4-sol26-sparc-local
    autoconf-2.13
    automake-1.4
    mysql-3.23.24-beta (if you want mysql support)
    tar-1.13 (GNU tar)

2. Make sure your path includes the proper directories
    PATH=.:/usr/local/bin:/usr/sbin:/usr/bin:/usr/ccs/bin
    export PATH

3. gunzip php-x.x.x.tar.gz (if you have a .gz dist, otherwise go to 4)
4. tar xvf php-x.x.x.tar
5. cd ../php-x.x.x

6. For the following step, make sure /opt/netscape/suitespot/ is where
   your netscape server is installed. Otherwise, change to correct path:
    /configure --with-mysql=/usr/local/mysql --with-nsapi=/opt/netscape/suitespot/ --enable-track-vars --enable-libgcc
7. make
8. make install
```

After performing the base install and reading the appropriate readme file, you may need to performs some additional configuration steps.

Firstly you may need to add some paths to the LD_LIBRARY_PATH environment for Netscape to find all the shared libs. This can best done in the start script for your Netscape server. Windows users can probably skip this step. The start script is often located in: `/path/to/server/https-servername/start`

You may also need to edit the configuration files that are located in:`/path/to/server/https-servername/config/`.

Example 3-11. Configuration Example for Netscape Enterprise

```
Configuration Instructions for Netscape Enterprise Server
From: bhager@invacare.com

1. Add the following line to mime.types:
    type=magnus-internal/x-httpd-php exts=php

2. Add the following to obj.conf, shlib will vary depending on
    your OS, for unix it will be something like
    /opt/netscape/suitespot/bin/libphp4.so.

    You should place the following lines after mime types init.
    Init fn="load-modules" funcs="php4_init,php4_close,php4_execute,php4_auth_trans" shlib="/php4/nsapiPHP4.dll"
    Init fn=php4_init errorString="Failed to initialize PHP!"

    <object name="default">
    .
    .
    .
    .#NOTE this next line should happen after all 'ObjectType' and before all 'AddLog' lines
    Service fn="php4_execute" type="magnus-internal/x-httpd-php"
    .
    .
    </Object>


    <Object name="x-httpd-php">
    ObjectType fn="force-type" type="magnus-internal/x-httpd-php"
    Service fn=php4_execute
    </Object>


    Authentication configuration

    PHP authentication cannot be used with any other authentication. ALL AUTHENTICATION IS
    PASSED TO YOUR PHP SCRIPT. To configure PHP Authentication for the entire server, add
    the following line:

    <Object name="default">
    AuthTrans fn=php4_auth_trans
    .
    .
    .
    .
    </Object>

    To use PHP Authentication on a single directory, add the following:

    <Object ppath="d:\path\to\authenticated\dir\*">
    AuthTrans fn=php4_auth_trans
    </Object>
```

If you are running Netscape Enterprise 4.x, then you should use the following:

Example 3-12. Configuration Example for Netscape Enterprise 4.x

```
Place these lines after the mime types init, and everything else is similar
to the example configuration above.
From: Graeme Hoose (GraemeHoose@BrightStation.com)

Init fn="load-modules" shlib="/path/to/server4/bin/libphp4.so" funcs="php4_init,php4_close,php4_execute,php4_auth_trans"
Init fn="php4_init" LateInit="yes"
```

Installing PHP with Netscape on Windows

To Install PHP as CGI (for Netscape Enterprise Server, iPlanet, perhaps Fastrack), do the following:

- Copy `php4ts.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

To Install PHP as NSAPI (for Netscape Enterprise Server, iPlanet, perhaps Fastrack, do the following:

- Copy `php4ts.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).
- Stop your web service and edit `obj.conf`. At the end of the Init section, place these two lines (necessarily after mime type init!):
```
Init fn="load-modules" funcs="php4_init,php4_close,php4_execute,php4_auth_trans" shlib="c:/php/sapi/php4nsapi.dll"
Init fn="php4_init" errorString="Failed to initialise PHP!"
```
- In The `< Object name="default" >` section, place this line necessarily after all 'ObjectType' and before all 'AddLog' lines:
```
Service fn="php4_execute" type="magnus-internal/x-httpd-php"
```
- At the end of the file, create a new object called `x-httpd-php`, by inserting these lines:
```
<Object name="x-httpd-php">
ObjectType fn="force-type" type="magnus-internal/x-httpd-php"
Service fn=php4_execute
</Object>
```
- Restart your web service and apply changes
- Do it for each web server instance you want PHP to run

More details about setting up PHP as an NSAPI filter can be found here: http://benoit.noss.free.fr/php/install-php4.html

---

Servers-OmniHTTPd Server

This section contains notes and hints specific to OmniHTTPd.

OmniHTTPd 2.0b1 and up for Windows

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`.

- Step 1: Install OmniHTTPd server.
- Step 2: Right click on the blue OmniHTTPd icon in the system tray and select `Properties`
- Step 3: Click on `Web Server Global Settings`
- Step 4: On the 'External' tab, enter: `virtual = .php | actual = c:\path-to-php-dir\php.exe`, and use the Add button.
- Step 5: On the `Mime` tab, enter: `virtual = wwwserver/stdcgi | actual = .php`, and use the Add button.
- Step 6: Click `OK`

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

> **Note:** Some OmniHTTPd packages come with built in PHP support. You can choose at setup time to do a custom setup, and uncheck the PHP component. We recommend you to use the latest PHP binaries. Some OmniHTTPd servers come with PHP 4 beta distributions, so you should choose not to set up the built in support, but install your own. If the server is already on your machine, use the Replace button in Step 4 and 5 to set the new, correct information.

Servers-Oreilly Website Pro

This section contains notes and hints specific to Oreilly Website Pro.

Oreilly Website Pro 2.5 and up for Windows

This list describes how to set up the PHP CGI binary or the ISAPI module to work with Oreilly Website Pro on Windows.

- Edit the Server Properties and select the tab "Mapping".
- From the List select "Associations" and enter the desired extension (`.php`) and the path to the CGI exe (ex. `c:\php\php.exe`) or the ISAPI DLL file (ex. `c:\php\sapi\php4isapi.dll`).
- Select "Content Types" add the same extension (`.php`) and enter the content type. If you choose the CGI executable file, enter 'wwwserver/shellcgi', if you choose the ISAPI module, enter 'wwwserver/isapi' (both without quotes).

Servers-Sambar

This section contains notes and hints specific to the Sambar server for Windows.

Sambar Windows

This list describes how to set up the ISAPI module to work with the Sambar server on Windows.

- Find the file called mappings.ini (in the config directory) in the Sambar isntall directory.
- Open `mappings.ini` and add the following line under `[ISAPI]`:

  ```
  *.php = c:\php\php4isapi.dll
  ```

  (This line assumes that PHP was installed in `c:\php`.)
- Now restart the Sambar server for the changes to take effect.

Servers-Xitami

This section contains notes and hints specific to Xitami.

Xitami for Windows

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`.

- Make sure the webserver 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 executable i.e. `c:\php\php.exe`.
- Press the 'Save' icon.
- Restart the server to reflect changes.

Servers-Other web servers

PHP can be built to support a large number of web servers. Please see Server-related options for a full list of server-related configure options. The PHP CGI binaries are compatible with almost all webservers supporting the CGI standard.

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/. 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, 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!

Complete list of configure options

> **Note:** These are only used at compile time. If you want to alter PHP's runtime configuration, please see the chapter on Configuration.

The following is a complete list of options supported by PHP 4 `configure` scripts (as of 4.1.0), used when compiling in Unix-like environments. Some are available in PHP 3, some in PHP 4, and some in both. This is not noted yet.

There are general configuration options for the **configure** script, consult the appropriate manual pages for GNU **autoconf** or use the command **configure --help** for a full, up-to-date list.

- Database
- Graphics
- Miscellaneous
- PHP Behaviour
- Server

Configure Options in PHP 4

> **Note:** These options are only used in PHP 4 as of PHP 4.1.0. Some are available in older versions of PHP 4, some even in PHP 3, some only in PHP 4.1.0. If you want to compile an older version, some options will probably not be available.

Database options

`--with-dbplus`

Include dbplus support.

`--with-adabas[=DIR]`

Include Adabas D support. DIR is the Adabas base install directory, defaults to /usr/local.

--with-sapdb[=DIR]

Include SAP DB support. DIR is SAP DB base install directory, defaults to /usr/local.

--with-solid[=DIR]

Include Solid support. DIR is the Solid base install directory, defaults to /usr/local/solid.

--with-ibm-db2[=DIR]

Include IBM DB2 support. DIR is the DB2 base install directory, defaults to /home/db2inst1/sqllib.

--with-empress[=DIR]

Include Empress support. DIR is the Empress base install directory, defaults to $EMPRESSPATH. From PHP4, this option only supports Empress Version 8.60 and above.

--with-empress-bcs[=DIR]

Include Empress Local Access support. DIR is the Empress base install directory, defaults to $EMPRESSPATH. From PHP4, this option only supports Empress Version 8.60 and above.

--with-birdstep[=DIR]

Include Birdstep support. DIR is the Birdstep base install directory, defaults to /usr/local/birdstep.

--with-custom-odbc[=DIR]

Include a user defined ODBC support. The DIR is ODBC install base directory, which defaults to /usr/local. Make sure to define CUSTOM_ODBC_LIBS and have some odbc.h in your include dirs. E.g., you should define following for Sybase SQL Anywhere 5.5.00 on QNX, prior to run configure script: CPPFLAGS="-DODBC_QNX -DSQLANY_BUG" LDFLAGS=-lunix CUSTOM_ODBC_LIBS="-ldblib -lodbc".

--with-iodbc[=DIR]

Include iODBC support. DIR is the iODBC base install directory, defaults to /usr/local.

--with-esoob[=DIR]

Include Easysoft OOB support. DIR is the OOB base install directory, defaults to /usr/local/easysoft/oob/client.

--with-unixODBC[=DIR]

Include unixODBC support. DIR is the unixODBC base install directory, defaults to /usr/local.

--with-openlink[=DIR]

Include OpenLink ODBC support. DIR is the OpenLink base install directory, defaults to /usr/local. This is the same as iODBC.

--with-dbmaker[=DIR]

Include DBMaker support. DIR is the DBMaker base install directory, defaults to where the latest version of DBMaker is installed (such as /home/dbmaker/3.6).

--disable-unified-odbc

Disable unified ODBC support. Only applicable if iODBC, Adabas, Solid, Velocis or a custom ODBC interface is enabled. PHP 3 only!

Graphics options

--without-gd

Disable GD support. PHP 3 only!

--with-imagick

The imagick extension has been moved to PECL in PEAR and can be found here. Install instructions for PHP 4 can be found on the PEAR site.

Simply doing `--with-imagick` is only supported in PHP 3 unless you follow the instructions found on the PEAR site.

--with-ming[=DIR]

Include ming support.

Misc options

--enable-force-cgi-redirect

Enable the security check for internal server redirects. You should use this if you are running the CGI version with Apache.

--enable-discard-path

If this is enabled, the PHP CGI binary can safely be placed outside of the web tree and people will not be able to circumvent .htaccess security.

--with-fastcgi=SRCDIR

Build PHP as FastCGI application.

--enable-debug

Compile with debugging symbols.

--with-layout=TYPE

Sets how installed files will be laid out. Type is one of PHP (default) or GNU.

--with-pear=DIR

Install PEAR in DIR (default PREFIX/lib/php).

--without-pear

Do not install PEAR.

--enable-sigchild

Enable PHP's own SIGCHLD handler.

--disable-rpath

Disable passing additional runtime library search paths.

--enable-libgcc

Enable explicitly linking against libgcc.

--enable-php-streams

Include experimental php streams. Do not use unless you are testing the code!

--with-zlib-dir=<DIR>

Define the location of zlib install directory.

--with-aspell[=DIR]

Include ASPELL support.

--with-ccvs[=DIR]

Include CCVS support.

--with-cybercash[=DIR]

Include CyberCash support. DIR is the CyberCash MCK install directory.

--with-icap[=DIR]

Include ICAP support.

--with-ircg-config

Path to the ircg-config script.

--with-ircg

Include ircg support.

--enable-mailparse

Enable mailparse support.

--with-muscat[=DIR]

Include muscat support.

--with-satellite[=DIR]

Enable CORBA support via Satellite (EXPERIMENTAL) DIR is the base directory for ORBit.

--enable-trans-sid

Enable transparent session id propagation.

--with-system-regex

Use system regex library (deprecated).

--with-vpopmail[=DIR]

Include vpopmail support.

--with-tsrm-pthreads

Use POSIX threads (default).

--enable-shared[=PKGS]

Build shared libraries [default=yes].

--enable-static[=PKGS]

Build static libraries [default=yes].

--enable-fast-install[=PKGS]

Optimize for fast installation [default=yes].

--with-gnu-ld

Assume the C compiler uses GNU ld [default=no].

--disable-libtool-lock

Avoid locking (might break parallel builds).

--with-pic

Try to use only PIC/non-PIC objects [default=use both].

--enable-memory-limit

Compile with memory limit support.

--disable-url-fopen-wrapper

Disable the URL-aware fopen wrapper that allows accessing files via HTTP or FTP.

--enable-versioning

Export only required symbols. See INSTALL for more information.

--with-imsp[=DIR]

Include IMSp support (DIR is IMSP's include dir and libimsp.a dir). PHP 3 only!

--with-mck[=DIR]

Include Cybercash MCK support. DIR is the cybercash mck build directory, defaults to /usr/src/mck-3.2.0.3-linux for help look in extra/cyberlib. PHP 3 only!

--with-mod-dav=DIR

Include DAV support through Apache's mod_dav, DIR is mod_dav's installation directory (Apache module version only!) PHP 3 only!

--enable-debugger

Compile with remote debugging functions. PHP 3 only!

--enable-versioning

Take advantage of versioning and scoping provided by Solaris 2.x and Linux. PHP 3 only!

PHP options

--enable-maintainer-mode

Enable make rules and dependencies not useful (and sometimes confusing) to the casual installer.

--with-config-file-path=PATH

Sets the path in which to look for `php.ini`, defaults to PREFIX/lib.

--enable-safe-mode

Enable safe mode by default.

--with-exec-dir[=DIR]

Only allow executables in DIR when in safe mode defaults to /usr/local/php/bin.

--enable-magic-quotes

Enable magic quotes by default.

--disable-short-tags

Disable the short-form <? start tag by default.

Server options

--with-aolserver=DIR

Specify path to the installed AOLserver.

--with-apxs[=FILE]

Build shared Apache module. FILE is the optional pathname to the Apache apxs tool; defaults to apxs. Make sure you specify the version of apxs that is actually installed on your system and NOT the one that is in the apache source tarball.

--with-apache[=DIR]

Build Apache module. DIR is the top-level Apache build directory, defaults to /usr/local/apache.

--with-mod_charset

Enable transfer tables for mod_charset (Rus Apache).

--with-apxs2[=FILE]

Build shared Apache 2.0 module. FILE is the optional pathname to the Apache apxs tool; defaults to apxs.

--with-fhttpd[=DIR]

Build fhttpd module. DIR is the fhttpd sources directory, defaults to /usr/local/src/fhttpd.

--with-isapi=DIR

Build PHP as an ISAPI module for use with Zeus.

--with-nsapi=DIR

Specify path to the installed Netscape Server.

--with-phttpd=DIR

No information yet.

--with-pi3web=DIR

Build PHP as a module for use with Pi3Web.

--with-roxen=DIR

Build PHP as a Pike module. DIR is the base Roxen directory, normally /usr/local/roxen/server.

--enable-roxen-zts

Build the Roxen module using Zend Thread Safety.

--with-servlet[=DIR]

Include servlet support. DIR is the base install directory for the JSDK. This SAPI prereqs the java extension must be built as a shared dl.

--with-thttpd=SRCDIR

Build PHP as thttpd module.

--with-tux=MODULEDIR

Build PHP as a TUX module (Linux only).

Chapter 4. Configuration

The configuration file

The configuration file (called `php3.ini` in PHP 3.0, and simply `php.ini` as of PHP 4.0) 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.

The default location of `php.ini` is a compile time option (see the FAQ entry), but can be changed for the CGI and CLI version with the `-c` command line switch, see the chapter about using PHP from the command line. You can also use the environment variable PHPRC for an additional path to search for a `php.ini` file.

Not every PHP directive is documented below. For a list of all directives, please read your well commented `php.ini` file. You may want to view the latest php.ini here from CVS.

Note: The default value for the PHP directive register_globals changed from *on* to *off* in PHP 4.2.0.

Example 4-1. `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
magic_quotes_gpc = 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"
```

How to change configuration settings

Running PHP as 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)

With PHP 4.0, 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 table found within the **ini_set()** documentation.

> **Note:** With PHP 3.0, 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.
>
> ```
> php_value auto_prepend_file none
> ```

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.

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 4-2. Apache configuration example

```
<IfModule mod_php4.c>
  php_value include_path ".:/usr/local/lib/php"
  php_admin_flag safe_mode on
</IfModule>
<IfModule mod_php3.c>
  php3_include_path ".:/usr/local/lib/php"
  php3_safe_mode on
</IfModule>
```

> **Note:** 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`

Other interfaces to PHP

Regardless of the interface to PHP you can change certain values at runtime of your scripts through **ini_set()**. The following table provides an overview at which level a directive can be set/changed.

Table 4-1. Definition of PHP_INI_* constants

Constant	Value	Meaning
PHP_INI_USER	1	Entry can be set in user scripts
PHP_INI_PERDIR	2	Entry can be set in `php.ini`, `.htaccess` or `httpd.conf`
PHP_INI_SYSTEM	4	Entry can be set in `php.ini` or `httpd.conf`
PHP_INI_ALL	7	Entry can be set anywhere

You can view the settings of the configuration values in the output of **phpinfo()**. You can also access the values of individual configuration directives using **ini_get()** or **get_cfg_var()**.

Configuration directives

Httpd Options

Table 4-2. Httpd Options

Name	Default	Changeable
async_send	"0"	PHP_INI_ALL

Language Options

Table 4-3. Language and Misc Configuration Options

Name	Default	Changeable
short_open_tag	On	PHP_INI_SYSTEM|PHP_INI_PERDIR
asp_tags	Off	PHP_INI_SYSTEM|PHP_INI_PERDIR
precision	"14"	PHP_INI_ALL
y2k_compliance	Off	PHP_INI_ALL
allow_call_time_pass_reference	On	PHP_INI_SYSTEM|PHP_INI_PERDIR
expose_php	On	PHP_INI_SYSTEM

Here is a short explanation of the configuration directives.

short_open_tag **boolean**

Tells whether the short form (`<? ?>`) of PHP's open tag should be allowed. If you want to use PHP in combination with XML, you can disable this option in order to use `<?xml ?>` inline. Otherwise, you can print it with PHP, for example: `<?php echo '<?xml version="1.0"'; ?>`. Also if disabled, you must use the long form of the PHP open tag (`<?php ?>`).

Note: This directive also affects the shorthand `<?=`, which is identical to `<? echo`. Use of this shortcut requires short_open_tag to be on.

asp_tags **boolean**

Enables the use of ASP-like <% %> tags in addition to the usual <?php ?> tags. This includes the variable-value printing shorthand of <%= $value %>. For more information, see Escaping from HTML.

Note: Support for ASP-style tags was added in 3.0.4.

precision **integer**

The number of significant digits displayed in floating point numbers.

y2k_compliance **boolean**

Enforce year 2000 compliance (will cause problems with non-compliant browsers)

allow_call_time_pass_reference **boolean**

Whether to enable the ability to force arguments to be passed by reference at function call time. This method is deprecated and is likely to be unsupported in future versions of PHP/Zend. The encouraged method of specifying which arguments should be passed by reference is in the function declaration. You're encouraged to try and turn this option Off and make sure your scripts work properly with it in order to ensure they will work with future versions of the language (you will receive a warning each time you use this feature, and the argument will be passed by value instead of by reference).

See also References Explained.

expose_php **boolean**

Decides whether PHP may expose the fact that it is installed on the server (e.g. by adding its signature to the Web server header). It is no security threat in any way, but it makes it possible to determine whether you use PHP on your server or not.

Resource Limits

Table 4-4. Resource Limits

Name	Default	Changeable
memory_limit	"8M"	PHP_INI_ALL

Here is a short explanation of the configuration directives.

memory_limit **integer**

This sets the maximum amount of memory in bytes that a script is allowed to allocate. This helps prevent poorly written scripts for eating up all available memory on a server. In order to use this directive you must have enabled it at compile time. So, your configure line would have included: `--enable-memory-limit`. Note that you have to set it to -1 if you don't want any limit for your memory.

See also: max_execution_time.

Data Handling

Table 4-5. Data Handling Configuration Options

Name	Default	Changeable
track-vars	"On"	PHP_INI_??
arg_separator.output	"&"	PHP_INI_ALL
arg_separator.input	"&"	PHP_INI_SYSTEM|PHP_INI_PERDIR
variables_order	"EGPCS"	PHP_INI_ALL
register_globals	"Off"	PHP_INI_PERDIR|PHP_INI_SYSTEM
register_argc_argv	"On"	PHP_INI_PERDIR|PHP_INI_SYSTEM
post_max_size	"8M"	PHP_INI_SYSTEM|PHP_INI_PERDIR
gpc_order	"GPC"	PHP_INI_ALL
auto_prepend_file	""	PHP_INI_SYSTEM|PHP_INI_PERDIR
auto_append_file	""	PHP_INI_SYSTEM|PHP_INI_PERDIR
default_mimetype	"text/html"	PHP_INI_ALL
default_charset	"iso-8859-1"	PHP_INI_ALL
always_populate_raw_post_data	"0"	PHP_INI_SYSTEM|PHP_INI_PERDIR
allow_webdav_methods	"0"	PHP_INI_SYSTEM|PHP_INI_PERDIR

Here is a short explanation of the configuration directives.

track_vars **boolean**

If enabled, then Environment, GET, POST, Cookie, and Server variables can be found in the global associative arrays `$_ENV`, `$_GET`, `$_POST`, `$_COOKIE`, and `$_SERVER`.

Note that as of PHP 4.0.3, track_vars is always turned on.

arg_separator.output **string**

The separator used in PHP generated URLs to separate arguments.

arg_separator.input **string**

List of separator(s) used by PHP to parse input URLs into variables.

Note: Every character in this directive is considered as separator!

variables_order **string**

Set the order of the EGPCS (Environment, GET, POST, Cookie, Server) variable parsing. The default setting of this directive is "EGPCS". Setting this to "GP", for example, will cause PHP to completely ignore environment variables, cookies and server variables, and to overwrite any GET method variables with POST-method variables of the same name.

See also register_globals.

register_globals **boolean**

Tells whether or not to register the EGPCS (Environment, GET, POST, Cookie, Server) variables as global variables. For example; if register_globals = on, the url `http://www.example.com/test.php?id=3` will produce `$id`. Or, `$DOCUMENT_ROOT` from

`$_SERVER['DOCUMENT_ROOT']`. You may want to turn this off if you don't want to clutter your scripts' global scope with user data. As of PHP 4.2.0, this directive defaults to *off*. It's preferred to go through PHP Predefined Variables instead, such as the superglobals: `$_ENV`, `$_GET`, `$_POST`, `$_COOKIE`, and `$_SERVER`. Please read the security chapter on Using register_globals for related information.

Please note that register_globals cannot be set at runtime (**ini_set()**). Although, you can use `.htaccess` if your host allows it as described above. An example `.htaccess` entry: **`php_flag register_globals on`**.

Note: register_globals is affected by the variables_order directive.

`register_argc_argv` **boolean**

Tells PHP whether to declare the argv & argc variables (that would contain the GET information).

See also command line. Also, this directive became available in PHP 4.0.0 and was always "on" before that.

`post_max_size` **integer**

Sets max size of post data allowed. This setting also affects file upload. To upload large files, this value must be larger than upload_max_filesize.

If memory limit is enabled by your configure script, memory_limit also affects file uploading. Generally speaking, memory_limit should be larger than *`post_max_size`*.

`gpc_order` **string**

Set the order of GET/POST/COOKIE variable parsing. The default setting of this directive is "GPC". Setting this to "GP", for example, will cause PHP to completely ignore cookies and to overwrite any GET method variables with POST-method variables of the same name.

Note: This option is not available in PHP 4. Use variables_order instead.

`auto_prepend_file` **string**

Specifies the name of a file that is automatically parsed before the main file. The file is included as if it was called with the **include()** function, so include_path is used.

The special value none disables auto-prepending.

`auto_append_file` **string**

Specifies the name of a file that is automatically parsed after the main file. The file is included as if it was called with the **include()** function, so include_path is used.

The special value none disables auto-appending.

Note: If the script is terminated with **exit()**, auto-append will *not* occur.

`default_mimetype` **string**

`default_charset` **string**

As of 4.0b4, PHP always outputs a character encoding by default in the Content-type: header. To disable sending of the charset, simply set it to be empty.

`always_populate_raw_post_data` **boolean**

Always populate the $HTTP_RAW_POST_DATA variable.

`allow_webdav_methods` **boolean**

Allow handling of WebDAV http requests within PHP scripts (eg. PROPFIND, PROPPATCH, MOVE, COPY, etc..) If you want to get the post data of those requests, you have to set always_populate_raw_post_data as well.

See also: magic_quotes_gpc, magic-quotes-runtime, and magic_quotes_sybase.

Paths and Directories

Table 4-6. Paths and Directories Configuration Options

Name	Default	Changeable
include_path	PHP_INCLUDE_PATH	PHP_INI_ALL
doc_root	PHP_INCLUDE_PATH	PHP_INI_SYSTEM
user_dir	NULL	PHP_INI_SYSTEM
extension_dir	PHP_EXTENSION_DIR	PHP_INI_SYSTEM
cgi.force_redirect	"1"	PHP_INI_SYSTEM
cgi.redirect_status_env	""	PHP_INI_SYSTEM
fastcgi.impersonate	"0"	PHP_INI_SYSTEM

Here is a short explanation of the configuration directives.

include_path **string**

Specifies a list of directories where the **require()**, **include()** and **fopen_with_path()** functions look for files. The format is like the system's PATH environment variable: a list of directories separated with a colon in UNIX or semicolon in Windows.

Example 4-3. UNIX include_path

```
include_path=".:/php/includes"
```

Example 4-4. Windows include_path

```
include_path=".;c:\php\includes"
```

Using a . in the include path allows for relative includes as it means the current directory.

doc_root **string**

PHP's "root directory" on the server. Only used if non-empty. If PHP is configured with safe mode, no files outside this directory are served. If PHP was not compiled with FORCE_REDIRECT, you SHOULD set doc_root if you are running php as a CGI under any web server (other than IIS) The alternative is to use the cgi.force_redirect configuration below.

user_dir **string**

The base name of the directory used on a user's home directory for PHP files, for example public_html.

extension_dir **string**

In what directory PHP should look for dynamically loadable extensions. See also: enable_dl, and **dl()**.

extension **string**

Which dynamically loadable extensions to load when PHP starts up.

cgi.force_redirect **boolean**

cgi.force_redirect is necessary to provide security running PHP as a CGI under most web servers. Left undefined, PHP turns this on by default. You can turn it off *AT YOUR OWN RISK*.

Note: Windows Users: You CAN safely turn this off for IIS, in fact, you MUST. To get OmniHTTPD or Xitami to work you MUST turn it off.

cgi.cgi.redirect_status_env **string**

If cgi.force_redirect is turned on, and you are not running under Apache or Netscape (iPlanet) web servers, you MAY need to set an environment variable name that PHP will look for to know it is OK to continue execution.

Note: Setting this variable MAY cause security issues, KNOW WHAT YOU ARE DOING FIRST.

fastcgi.impersonate **string**

FastCGI under IIS (on WINNT based OS) supports the ability to impersonate security tokens of the calling client. This allows IIS to define the security context that the request runs under. mod_fastcgi under Apache does not currently support this feature (03/17/2002) Set to 1 if running under IIS. Default is zero.

File Uploads

Table 4-7. File Uploads Configuration Options

Name	Default	Changeable
file_uploads	"1"	PHP_INI_SYSTEM
upload_tmp_dir	NULL	PHP_INI_SYSTEM
upload_max_filesize	"2M"	PHP_INI_SYSTEM|PHP_INI_PERDIR

Here is a short explanation of the configuration directives.

file_uploads **boolean**

Whether or not to allow HTTP file uploads. See also the upload_max_filesize, upload_tmp_dir, and post_max_size directives.

upload_tmp_dir **string**

The temporary directory used for storing files when doing file upload. Must be writable by whatever user PHP is running as. If not specified PHP will use the system's default.

upload_max_filesize **integer**

The maximum size of an uploaded file.

General SQL

Table 4-8. General SQL Configuration Options

Name	Default	Changeable
sql.safe_mode	"0"	PHP_INI_SYSTEM

Here is a short explanation of the configuration directives.

sql.safe_mode **boolean**

Debugger Configuration Directives

debugger.host **string**

DNS name or IP address of host used by the debugger.

debugger.port **string**

Port number used by the debugger.

debugger.enabled **boolean**

Whether the debugger is enabled.

Chapter 5. Security

PHP is a powerful language and the interpreter, whether included in a web server as a module or executed as a separate CGI binary, is able to access files, execute commands and open network connections on the server. These properties make anything run on a web server insecure by default. PHP is designed specifically to be a more secure language for writing CGI programs than Perl or C, and with correct selection of compile-time and runtime configuration options, and proper coding practices, it can give you exactly the combination of freedom and security you need.

As there are many different ways of utilizing PHP, there are many configuration options controlling its behaviour. A large selection of options guarantees you can use PHP for a lot of purposes, but it also means there are combinations of these options and server configurations that result in an insecure setup.

The configuration flexibility of PHP is equally rivalled by the code flexibility. PHP can be used to build complete server applications, with all the power of a shell user, or it can be used for simple server-side includes with little risk in a tightly controlled environment. How you build that environment, and how secure it is, is largely up to the PHP developer.

This chapter starts with some general security advice, explains the different configuration option combinations and the situations they can be safely used, and describes different considerations in coding for different levels of security.

General considerations

A completely secure system is a virtual impossibility, so an approach often used in the security profession is one of balancing risk and usability. If every variable submitted by a user required two forms of biometric validation (such as a retinal scan and a fingerprint), you would have an extremely high level of accountability. It would also take half an hour to fill out a fairly complex form, which would tend to encourage users to find ways of bypassing the security.

The best security is often inobtrusive enough to suit the requirements without the user being prevented from accomplishing their work, or over-burdening the code author with excessive complexity. Indeed, some security attacks are merely exploits of this kind of overly built security, which tends to erode over time.

A phrase worth remembering: A system is only as good as the weakest link in a chain. If all transactions are heavily logged based on time, location, transaction type, etc. but the user is only verified based on a single cookie, the validity of tying the users to the transaction log is severely weakened.

When testing, keep in mind that you will not be able to test all possibilities for even the simplest of pages. The input you may expect will be completely unrelated to the input given by a disgruntled employee, a cracker with months of time on their hands, or a housecat walking across the keyboard. This is why it's best to look at the code from a logical perspective, to discern where unexpected data can be introduced, and then follow how it is modified, reduced, or amplified.

The Internet is filled with people trying to make a name for themselves by breaking your code, crashing your site, posting inappropriate content, and otherwise making your day interesting. It doesn't matter if you have a small or large site, you are a target by simply being online, by having a server that can be connected to. Many cracking programs do not discern by size, they simply trawl massive IP blocks looking for victims. Try not to become one.

Installed as CGI binary

Possible attacks

Using PHP as a CGI binary is an option for setups that for some reason do not wish to integrate PHP as a module into server software (like Apache), or will use PHP with different kinds of CGI wrappers to create safe chroot and setuid environments for scripts. This setup usually involves installing executable PHP binary to the web server cgi-bin directory. CERT advisory CA-96.11 recommends against placing any interpreters into cgi-bin. Even if the PHP binary can be used as a standalone interpreter, PHP is designed to prevent the attacks this setup makes possible:

- Accessing system files: `http://my.host/cgi-bin/php?/etc/passwd`

  The query information in a url after the question mark (?) is passed as command line arguments to the interpreter by the CGI interface. Usually interpreters open and execute the file specified as the first argument on the command line.

  When invoked as a CGI binary, PHP refuses to interpret the command line arguments.

- Accessing any web document on server: `http://my.host/cgi-bin/php/secret/doc.html`

  The path information part of the url after the PHP binary name, `/secret/doc.html` is conventionally used to specify the name of the file to be opened and interpreted by the CGI program. Usually some web server configuration directives (Apache: Action) are used to redirect requests to documents like `http://my.host/secret/script.php` to the PHP interpreter. With this setup, the web server first checks the access permissions to the directory `/secret`, and after that creates the redirected request `http://my.host/cgi-bin/php/secret/script.php`. Unfortunately, if the request is originally given in this form, no access checks are made by web server for file `/secret/script.php`, but only for the `/cgi-bin/php` file. This way any user able to access `/cgi-bin/php` is able to access any protected document on the web server.

  In PHP, compile-time configuration option --enable-force-cgi-redirect and runtime configuration directives doc_root and user_dir can be used to prevent this attack, if the server document tree has any directories with access restrictions. See below for full the explanation of the different combinations.

Case 1: only public files served

If your server does not have any content that is not restricted by password or ip based access control, there is no need for these configuration options. If your web server does not allow you to do redirects, or the server does not have a way to communicate to the PHP binary that the request is a safely redirected request, you can specify the option --enable-force-cgi-redirect to the

configure script. You still have to make sure your PHP scripts do not rely on one or another way of calling the script, neither by directly `http://my.host/cgi-bin/php/dir/script.php` nor by redirection `http://my.host/dir/script.php`.

Redirection can be configured in Apache by using AddHandler and Action directives (see below).

Case 2: using --enable-force-cgi-redirect

This compile-time option prevents anyone from calling PHP directly with a url like `http://my.host/cgi-bin/php/secretdir/script.php`. Instead, PHP will only parse in this mode if it has gone through a web server redirect rule.

Usually the redirection in the Apache configuration is done with the following directives:

```
Action php-script /cgi-bin/php
AddHandler php-script .php
```

This option has only been tested with the Apache web server, and relies on Apache to set the non-standard CGI environment variable `REDIRECT_STATUS` on redirected requests. If your web server does not support any way of telling if the request is direct or redirected, you cannot use this option and you must use one of the other ways of running the CGI version documented here.

Case 3: setting doc_root or user_dir

To include active content, like scripts and executables, in the web server document directories is sometimes consider an insecure practice. If, because of some configuration mistake, the scripts are not executed but displayed as regular HTML documents, this may result in leakage of intellectual property or security information like passwords. Therefore many sysadmins will prefer setting up another directory structure for scripts that are accessible only through the PHP CGI, and therefore always interpreted and not displayed as such.

Also if the method for making sure the requests are not redirected, as described in the previous section, is not available, it is necessary to set up a script doc_root that is different from web document root.

You can set the PHP script document root by the configuration directive doc_root in the configuration file, or you can set the environment variable `PHP_DOCUMENT_ROOT`. If it is set, the CGI version of PHP will always construct the file name to open with this *`doc_root`* and the path information in the request, so you can be sure no script is executed outside this directory (except for *`user_dir`* below).

Another option usable here is user_dir. When user_dir is unset, only thing controlling the opened file name is *`doc_root`*. Opening an url like `http://my.host/~user/doc.php` does not result in opening a file under users home directory, but a file called `~user/doc.php` under doc_root (yes, a directory name starting with a tilde [~]).

If user_dir is set to for example `public_php`, a request like `http://my.host/~user/doc.php` will open a file called `doc.php` under the directory named `public_php` under the home directory of the user. If the home of the user is `/home/user`, the file executed is `/home/user/public_php/doc.php`.

`user_dir` expansion happens regardless of the *`doc_root`* setting, so you can control the document root and user directory access separately.

Case 4: PHP parser outside of web tree

A very secure option is to put the PHP parser binary somewhere outside of the web tree of files. In `/usr/local/bin`, for example. The only real downside to this option is that you will now have to put a line similar to:

```
#!/usr/local/bin/php
```

as the first line of any file containing PHP tags. You will also need to make the file executable. That is, treat it exactly as you would treat any other CGI script written in Perl or sh or any other common scripting language which uses the `#!` shell-escape mechanism for launching itself.

To get PHP to handle `PATH_INFO` and `PATH_TRANSLATED` information correctly with this setup, the php parser should be compiled with the --enable-discard-path configure option.

Installed as an Apache module

When PHP is used as an Apache module it inherits Apache's user permissions (typically those of the "nobody" user). This has several impacts on security and authorization. For example, if you are using PHP to access a database, unless that database has built-in access control, you will have to make the database accessable to the "nobody" user. This means a malicious script could access and modify the database, even without a username and password. It's entirely possible that a web spider could stumble across a database administrator's web page, and drop all of your databases. You can protect against this with Apache authorization, or you can design your own access model using LDAP, .htaccess files, etc. and include that code as part of your PHP scripts.

Often, once security is established to the point where the PHP user (in this case, the apache user) has very little risk attached to it, it is discovered that PHP is now prevented from writing any files to user directories. Or perhaps it has been prevented from accessing or changing databases. It has equally been secured from writing good and bad files, or entering good and bad database transactions.

A frequent security mistake made at this point is to allow apache root permissions, or to escalate apache's abilitites in some other way.

Escalating the Apache user's permissions to root is extremely dangerous and may compromise the entire system, so sudo'ing, chroot'ing, or otherwise running as root should not be considered by those who are not security professionals.

There are some simpler solutions. By using open_basedir you can control and restrict what directories are allowed to be used for PHP. You can also set up apache-only areas, to restrict all web based activity to non-user, or non-system, files.

Filesystem Security

PHP is subject to the security built into most server systems with respect to permissions on a file and directory basis. This allows you to control which files in the filesystem may be read. Care should be taken with any files which are world readable to ensure that they are safe for reading by all users who have access to that filesystem.

Since PHP was designed to allow user level access to the filesystem, it's entirely possible to write a PHP script that will allow you to read system files such as /etc/passwd, modify your ethernet connections, send massive printer jobs out, etc. This has some obvious implications, in that you need to ensure that the files that you read from and write to are the appropriate ones.

Consider the following script, where a user indicates that they'd like to delete a file in their home directory. This assumes a situation where a PHP web interface is regularly used for file management, so the Apache user is allowed to delete files in the user home directories.

Example 5-1. Poor variable checking leads to....

```
<?php
// remove a file from the user's home directory
$username = $_POST['user_submitted_name'];
$homedir = "/home/$username";
$file_to_delete = "$userfile";
unlink ("$homedir/$userfile");
echo "$file_to_delete has been deleted!";
?>
```

Since the username is postable from a user form, they can submit a username and file belonging to someone else, and delete files. In this case, you'd want to use some other form of authentication. Consider what could happen if the variables submitted were "../etc/" and "passwd". The code would then effectively read:

Example 5-2. ... A filesystem attack

```
<?php
// removes a file from anywhere on the hard drive that
// the PHP user has access to. If PHP has root access:
$username = "../etc/";
$homedir = "/home/../etc/";
$file_to_delete = "passwd";
unlink ("/home/../etc/passwd");
echo "/home/../etc/passwd has been deleted!";
?>
```

There are two important measures you should take to prevent these issues.

- Only allow limited permissions to the PHP web user binary.
- Check all variables which are submitted.

Here is an improved script:

Example 5-3. More secure file name checking

```
<?php
```

```
// removes a file from the hard drive that
// the PHP user has access to.
$username = $_SERVER['REMOTE_USER']; // using an authentication mechanisim

$homedir = "/home/$username";

$file_to_delete = basename("$userfile"); // strip paths
unlink ($homedir/$file_to_delete);

$fp = fopen("/home/logging/filedelete.log","+a"); //log the deletion
$logstring = "$username $homedir $file_to_delete";
fputs ($fp, $logstring);
fclose($fp);

echo "$file_to_delete has been deleted!";
?>
```

However, even this is not without it's flaws. If your authentication system allowed users to create their own user logins, and a user chose the login "../etc/", the system is once again exposed. For this reason, you may prefer to write a more customized check:

Example 5-4. More secure file name checking

```
<?php
$username = $_SERVER['REMOTE_USER']; // using an authentication mechanisim
$homedir = "/home/$username";

if (!ereg('^[^./][^/]*$', $userfile))
     die('bad filename'); //die, do not process

if (!ereg('^[^./][^/]*$', $username))
     die('bad username'); //die, do not process
//etc...
?>
```

Depending on your operating system, there are a wide variety of files which you should be concerned about, including device entries (/dev/ or COM1), configuration files (/etc/ files and the .ini files), well known file storage areas (/home/, My Documents), etc. For this reason, it's usually easier to create a policy where you forbid everything except for what you explicitly allow.

Database Security

Nowadays, databases are cardinal components of any web based application by enabling websites to provide varying dynamic content. Since very sensitive or secret informations can be stored in such database, you should strongly consider to protect them somehow.

To retrieve or to store any information you need to connect to the database, send a legitimate query, fetch the result, and close the connecion. Nowadays, the commonly used query language in this interaction is the Structured Query Language (SQL). See how an attacker can tamper with an SQL query.

As you can realize, PHP cannot protect your database by itself. The following sections aim to be an introduction into the very basics of how to access and manipulate databases within PHP scripts.

Keep in mind this simple rule: defence in depth. In the more place you take the more action to increase the protection of your database, the less probability of that an attacker succeeds, and exposes or abuse any stored secret information. Good design of the database schema and the application deals with your greatest fears.

Designing Databases

The first step is always to create the database, unless you want to use an existing third party's one. When a database is created, it is assigned to an owner, who executed the creation statement. Usually, only the owner (or a superuser) can do anything with the objects in that database, and in order to allow other users to use it, privileges must be granted.

Applications should never connect to the database as its owner or a superuser, because these users can execute any query at will, for example, modifying the schema (e.g. dropping tables) or deleting its entire content.

You may create different database users for every aspect of your application with very limited rights to database objects. The most required privileges should be granted only, and avoid that the same user can interact with the database in different use cases. This means that if intruders gain access to your database using one of these credentials, they can only effect as many changes as your application can.

You are encouraged not to implement all the business logic in the web application (i.e. your script), instead to do it in the database schema using views, triggers or rules. If the system evolves, new ports will be intended to open to the database, and you have to reimplement the logic in each separate database client. Over and above, triggers can be used to transparently and automatically handle fields, which often provides insight when debugging problems with your application or tracing back

transactions.

Connecting to Database

You may want to estabilish the connections over SSL to encrypt client/server communications for increased security, or you can use ssh to encrypt the network connection between clients and the database server. If either of them is done, then monitoring your traffic and gaining informations in this way will be a hard work.

Encrypted Storage Model

SSL/SSH protects data travelling from the client to the server, SSL/SSH does not protect the persistent data stored in a database. SSL is an on-the-wire protocol.

Once an attacker gains access to your database directly (bypassing the webserver), the stored sensitive data may be exposed or misused, unless the information is protected by the database itself. Encrypting the data is a good way to mitigate this threat, but very few databases offer this type of data encryption.

The easiest way to work around this problem is to first create your own encryption package, and then use it from within your PHP scripts. PHP can assist you in this case with its several extensions, such as Mcrypt and Mhash, covering a wide variety of encryption algorithms. The script encrypts the data be stored first, and decrypts it when retrieving. See the references for further examples how encryption works.

In case of truly hidden data, if its raw representation is not needed (i.e. not be displayed), hashing may be also taken into consideration. The well-known example for the hashing is storing the MD5 hash of a password in a database, instead of the password itself. See also **crypt()** and **md5()**.

Example 5-5. Using hashed password field

```
// storing password hash
$query  = sprintf("INSERT INTO users(name,pwd) VALUES('%s','%s');",
            addslashes($username), md5($password));
$result = pg_exec($connection, $query);

// querying if user submitted the right password
$query = sprintf("SELECT 1 FROM users WHERE name='%s' AND pwd='%s';",
            addslashes($username), md5($password));
$result = pg_exec($connection, $query);

if (pg_numrows($result) > 0) {
    echo "Welcome, $username!";
}
else {
    echo "Authentication failed for $username.";
}
```

SQL Injection

Many web developers are unaware of how SQL queries can be tampered with, and assume that an SQL query is a trusted command. It means that SQL queries are able to circumvent access controls, thereby bypassing standard authentication and authorization checks, and sometimes SQL queries even may allow access to host operating system level commands.

Direct SQL Command Injection is a technique where an attacker creates or alters existing SQL commands to expose hidden data, or to override valuable ones, or even to execute dangerous system level commands on the database host. This is accomplished by the application taking user input and combining it with static parameters to build a SQL query. The following examples are based on true stories, unfortunately.

Owing to the lack of input validation and connecting to the database on behalf of a superuser or the one who can create users, the attacker may create a superuser in your database.

Example 5-6. Splitting the result set into pages ... and making superusers (PostgreSQL and MySQL)

```
$offset = argv[0]; // beware, no input validation!
$query  = "SELECT id, name FROM products ORDER BY name LIMIT 20 OFFSET $offset;";
// with PostgreSQL
$result = pg_exec($conn, $query);
// with MySQL
$result = mysql_query($query);
```

Normal users click on the 'next', 'prev' links where the `$offset` is encoded into the URL. The script expects that the incoming `$offset` is decimal number. However, someone tries to break in with appending **urlencode()**'d form of the following to the URL

```
// in case of PostgreSQL
0;
insert into pg_shadow(usename,usesysid,usesuper,usecatupd,passwd)
    select 'crack', usesysid, 't','t','crack'
    from pg_shadow where usename='postgres';
--

// in case of MySQL
0;
UPDATE user SET Password=PASSWORD('crack') WHERE user='root';
FLUSH PRIVILEGES;
```

If it happened, then the script would present a superuser access to him. Note that `0;` is to supply a valid offset to the original query and to terminate it.

> **Note:** It is common technique to force the SQL parser to ignore the rest of the query written by the developer with `--` which is the comment sign in SQL.

A feasible way to gain passwords is to circumvent your search result pages. What the attacker needs only is to try if there is any submitted variable used in SQL statement which is not handled properly. These filters can be set commonly in a preceding form to customize `WHERE, ORDER BY, LIMIT` and `OFFSET` clauses in `SELECT` statements. If your database supports the `UNION` construct, the attacker may try to append an entire query to the original one to list passwords from an arbitrary table. Using encrypted password fields is strongly encouraged.

Example 5-7. Listing out articles ... and some passwords (any database server)

```
$query  = "SELECT id, name, inserted, size FROM products
                WHERE size = '$size'
                ORDER BY $order LIMIT $limit, $offset;";
$result = odbc_exec($conn, $query);
```

The static part of the query can be combined with another `SELECT` statement which reveals all passwords:

```
'
union select '1', concat(uname||'-'||passwd) as name, '1971-01-01', '0' from usertable;
--
```

If this query (playing with the `'` and `--`) were assigned to one of the variables used in `$query`, the query beast awakened.

SQL UPDATEs are also subject to attacking your database. These queries are also threatened by chopping and appending an entirely new query to it. But the attacker might fiddle with the `SET` clause. In this case some schema information must be possessed to manipulate the query successfully. This can be acquired by examing the form variable names, or just simply brute forcing. There are not so many naming convention for fields storing passwords or usernames.

Example 5-8. From resetting a password ... to gaining more privileges (any database server)

```
$query = "UPDATE usertable SET pwd='$pwd' WHERE uid='$uid';";
```

But a malicious user sumbits the value `' or uid like'%admin%'; --` to `$uid` to change the admin's password, or simply sets `$pwd` to `"hehehe', admin='yes', trusted=100 "` (with a trailing space) to gain more privileges. Then, the query will be twisted:

```
// $uid == ' or uid like'%admin%'; --
$query = "UPDATE usertable SET pwd='...' WHERE uid='' or uid like '%admin%'; --";

// $pwd == "hehehe', admin='yes', trusted=100 "
$query = "UPDATE usertable SET pwd='hehehe', admin='yes', trusted=100 WHERE ...;"
```

A frightening example how operating system level commands can be accessed on some database hosts.

Example 5-9. Attacking the database host's operating system (MSSQL Server)

```
$query  = "SELECT * FROM products WHERE id LIKE '%$prod%'";
$result = mssql_query($query);
```

If attacker submits the value `a%' exec master..xp_cmdshell 'net user test testpass /ADD' --` to `$prod`, then the `$query` will be:

```
$query  = "SELECT * FROM products
                  WHERE id LIKE '%a%'
                  exec master..xp_cmdshell 'net user test testpass /ADD'--";
$result = mssql_query($query);
```

MSSQL Server executes the SQL statements in the batch including a command to add a new user to the local accounts database. If this application were running as `sa` and the MSSQLSERVER service is running with sufficient privileges, the attacker would now have an account with which to access this machine.

> **Note:** Some of the examples above is tied to a specific database server. This does not mean that a similar attack is impossible against other products. Your database server may be so vulnerable in other manner.

Avoiding techniques

You may plead that the attacker must possess a piece of information about the database schema in most examples. You are right, but you never know when and how it can be taken out, and if it happens, your database may be exposed. If you are using an open source, or publicly available database handling package, which may belong to a content management system or forum, the intruders easily produce a copy of a piece of your code. It may be also a security risk if it is a poorly designed one.

These attacks are mainly based on exploiting the code not being written with security in mind. Never trust on any kind of input, especially which comes from the client side, even though it comes from a select box, a hidden input field or a cookie. The first example shows that such a blameless query can cause disasters.

- Never connect to the database as a superuser or as the database owner. Use always customized users with very limited privileges.
- Check if the given input has the expected data type. PHP has a wide range of input validating functions, from the simplest ones found in Variable Functions and in Character Type Functions (e.g. **is_numeric()**, **ctype_digit()** respectively) onwards the Perl compatible Regular Expressions support.
- If the application waits for numerical input, consider to verify data with **is_numeric()**, or silently change its type using **settype()**, or use its numeric representation by **sprintf()**.

  Example 5-10. A more secure way to compose a query for paging

  ```
  settype($offset, 'integer');
  $query = "SELECT id, name FROM products ORDER BY name LIMIT 20 OFFSET $offset;";

  // please note %d in the format string, using %s would be meaningless
  $query = sprintf("SELECT id, name FROM products ORDER BY name LIMIT 20 OFFSET %d;",
                   $offset);
  ```

- Quote each non numeric user input which is passed to the database with **addslashes()** or **addcslashes()**. See the first example. As the examples shows, quotes burnt into the static part of the query is not enough, and can be easily hacked.
- Do not print out any database specific information, especially about the schema, by fair means or foul. See also Error Reporting and Error Handling and Logging Functions.
- You may use stored procedures and previously defined cursors to abstract data access so that users do not directly access tables or views, but this solution has another impacts.

Besides these, you benefit from logging queries either within your script or by the database itself, if it supports. Obviously, the logging is unable to prevent any harmful attempt, but it can be helpful to trace back which application has been circumvented. The log is not useful by itself, but through the information it contains. The more detail is generally better.

Error Reporting

With PHP security, there are two sides to error reporting. One is beneficial to increasing security, the other is detrimental.

A standard attack tactic involves profiling a system by feeding it improper data, and checking for the kinds, and contexts, of the errors which are returned. This allows the system cracker to probe for information about the server, to determine possible weaknesses. For example, if an attacker had gleaned information about a page based on a prior form submission, they may attempt to override variables, or modify them:

Example 5-11. Attacking Variables with a custom HTML page

```
<form method="post" action="attacktarget?username=badfoo&password=badfoo">
<input type="hidden" name="username" value="badfoo">
<input type="hidden" name="password" value="badfoo">
</form>
```

The PHP errors which are normally returned can be quite helpful to a developer who is trying to debug a script, indicating such things as the function or file that failed, the PHP file it failed in, and the line number which the failure occured in. This is all information that can be exploited. It is not uncommon for a php developer to use **show_source()**, **highlight_string()**, or **highlight_file()** as a debugging measure, but in a live site, this can expose hidden variables, unchecked syntax, and other dangerous information. Especially dangerous is running code from known sources with built-in debugging handlers, or using common debugging techniques. If the attacker can determine what general technique you are using, they may try to brute-force a page, by sending various common debugging strings:

Example 5-12. Exploiting common debugging variables

```
<form method="post" action="attacktarget?errors=Y&amp;showerrors=1"&debug=1">
<input type="hidden" name="errors" value="Y">
```

```
<input type="hidden" name="showerrors" value="1">
<input type="hidden" name="debug" value="1">
</form>
```

Regardless of the method of error handling, the ability to probe a system for errors leads to providing an attacker with more information.

For example, the very style of a generic PHP error indicates a system is running PHP. If the attacker was looking at an .html page, and wanted to probe for the back-end (to look for known weaknesses in the system), by feeding it the wrong data they may be able to determine that a system was built with PHP.

A function error can indicate whether a system may be running a specific database engine, or give clues as to how a web page or programmed or designed. This allows for deeper investigation into open database ports, or to look for specific bugs or weaknesses in a web page. By feeding different pieces of bad data, for example, an attacker can determine the order of authentication in a script, (from the line number errors) as well as probe for exploits that may be exploited in different locations in the script.

A filesystem or general PHP error can indicate what permissions the webserver has, as well as the structure and organization of files on the web server. Developer written error code can aggravate this problem, leading to easy exploitation of formerly "hidden" information.

There are three major solutions to this issue. The first is to scrutinize all functions, and attempt to compensate for the bulk of the errors. The second is to disable error reporting entirely on the running code. The third is to use PHP's custom error handling functions to create your own error handler. Depending on your security policy, you may find all three to be applicable to your situation.

One way of catching this issue ahead of time is to make use of PHP's own **error_reporting()**, to help you secure your code and find variable usage that may be dangerous. By testing your code, prior to deployment, with E_ALL, you can quickly find areas where your variables may be open to poisoning or modification in other ways. Once you are ready for deployment, by using E_NONE, you insulate your code from probing.

Example 5-13. Finding dangerous variables with E_ALL

```
<?php
if ($username) {  // Not initialized or checked before usage
    $good_login = 1;
}
if ($good_login == 1) { // If above test fails, not initialized or checked before usage
    fpassthru ("/highly/sensitive/data/index.html");
}
?>
```

Using Register Globals

One feature of PHP that can be used to enhance security is configuring PHP with register_globals = off. By turning off the ability for any user-submitted variable to be injected into PHP code, you can reduce the amount of variable poisoning a potential attacker may inflict. They will have to take the additional time to forge submissions, and your internal variables are effectively isolated from user submitted data.

While it does slightly increase the amount of effort required to work with PHP, it has been argued that the benefits far outweigh the effort.

Example 5-14. Working with register_globals=on

```
<?php
if ($username) {  // can be forged by a user in get/post/cookies
    $good_login = 1;
}

if ($good_login == 1) { // can be forged by a user in get/post/cookies,
    fpassthru ("/highly/sensitive/data/index.html");
}
?>
```

Example 5-15. Working with register_globals = off

```
<?php
if($_COOKIE['username']){
    // can only come from a cookie, forged or otherwise
    $good_login = 1;
    fpassthru ("/highly/sensitive/data/index.html");
}
?>
```

By using this wisely, it's even possible to take preventative measures to warn when forging is being attempted. If you know

ahead of time exactly where a variable should be coming from, you can check to see if submitted data is coming from an inappropriate kind of submission. While it doesn't guarantee that data has not been forged, it does require an attacker to guess the right kind of forging.

Example 5-16. Detecting simple variable poisoning

```
<?php
if ($_COOKIE['username'] &&
    !$_POST['username'] &&
    !$_GET['username'] ) {
    // Perform other checks to validate the user name...
    $good_login = 1;
    fpassthru ("/highly/sensitive/data/index.html");
} else {
   mail("admin@example.com", "Possible breakin attempt", $_SERVER['REMOTE_ADDR']);
   echo "Security violation, admin has been alerted.";
   exit;
}
?>
```

Of course, simply turning off register_globals does not mean code is secure. For every piece of data that is submitted, it should also be checked in other ways.

User Submitted Data

The greatest weakness in many PHP programs is not inherent in the language itself, but merely an issue of code not being written with security in mind. For this reason, you should always take the time to consider the implications of a given piece of code, to ascertain the possible damage if an unexpected variable is submitted to it.

Example 5-17. Dangerous Variable Usage

```
<?php
// remove a file from the user's home directory... or maybe
// somebody else's?
unlink ($evil_var);

// Write logging of their access... or maybe an /etc/passwd entry?
fputs ($fp, $evil_var);

// Execute something trivial.. or rm -rf *?
system ($evil_var);
exec ($evil_var);

?>
```

You should always carefully examine your code to make sure that any variables being submitted from a web browser are being properly checked, and ask yourself the following questions:

- Will this script only affect the intended files?
- Can unusual or undesirable data be acted upon?
- Can this script be used in unintended ways?
- Can this be used in conjunction with other scripts in a negative manner?
- Will any transactions be adequately logged?

By adequately asking these questions while writing the script, rather than later, you prevent an unfortunate re-write when you need to increase your security. By starting out with this mindset, you won't guarantee the security of your system, but you can help improve it.

You may also want to consider turning off register_globals, magic_quotes, or other convenience settings which may confuse you as to the validity, source, or value of a given variable. Working with PHP in error_reporting(E_ALL) mode can also help warn you about variables being used before they are checked or initialized (so you can prevent unusual data from being operated upon).

Hiding PHP

In general, security by obscurity is one of the weakest forms of security. But in some cases, every little bit of extra security is desirable.

A few simple techniques can help to hide PHP, possibly slowing down an attacker who is attempting to discover weaknesses in your system. By setting expose_php = off in your `php.ini` file, you reduce the amount of information available to them.

Another tactic is to configure web servers such as apache to parse different filetypes through PHP, either with an .htaccess directive, or in the apache configuration file itself. You can then use misleading file extensions:

Example 5-18. Hiding PHP as another language

```
# Make PHP code look like other code types
AddType application/x-httpd-php .asp .py .pl
```

Or obscure it completely:

Example 5-19. Using unknown types for PHP extensions

```
# Make PHP code look like unknown types
AddType application/x-httpd-php .bop .foo .133t
```

Or hide it as html code, which has a slight performance hit because all html will be parsed through the PHP engine:

Example 5-20. Using html types for PHP extensions

```
# Make all PHP code look like html
AddType application/x-httpd-php .htm .html
```

For this to work effectively, you must rename your PHP files with the above extensions. While it is a form of security through obscurity, it's a minor preventative measure with few drawbacks.

Keeping Current

PHP, like any other large system, is under constant scrutiny and improvement. Each new version will often include both major and minor changes to enhance and repair security flaws, configuration mishaps, and other issues that will affect the overall security and stability of your system.

Like other system-level scripting languages and programs, the best approach is to update often, and maintain awareness of the latest versions and their changes.

II. Language Reference

Table of Contents

Chapter 6. Basic syntax

Escaping from HTML

When PHP parses a file, it simply passes the text of the file through until it encounters one of the special tags which tell it to start interpreting the text as PHP code. The parser then executes all the code it finds, up until it runs into a PHP closing tag, which tells the parser to just start passing the text through again. This is the mechanism which allows you to embed PHP code inside HTML: everything outside the PHP tags is left utterly alone, while everything inside is parsed as code.

There are four sets of tags which can be used to denote blocks of PHP code. Of these, only two (<?php. . .?> and <script language="php">. . .</script>) are always available; the others can be turned on or off from the `php.ini` configuration file. While the short-form tags and ASP-style tags may be convenient, they are not as portable as the longer versions. Also, if you intend to embed PHP code in XML or XHTML, you will need to use the <?php. . .?> form to conform to the XML.

The tags supported by PHP are:

Example 6-1. Ways of escaping from HTML

```
1.  <?php echo("if you want to serve XHTML or XML documents, do like this\n"); ?>

2.  <? echo ("this is the simplest, an SGML processing instruction\n"); ?>
    <?= expression ?> This is a shortcut for "<? echo expression ?>"

3.  <script language="php">
        echo ("some editors (like FrontPage) don't
              like processing instructions");
    </script>

4.  <% echo ("You may optionally use ASP-style tags"); %>
    <%= $variable; # This is a shortcut for "<% echo . . ." %>
```

The first way, <?php. . .?>, is the preferred method, as it allows the use of PHP in XML-conformant code such as XHTML.

The second way is not available always. Short tags are available only when they have been enabled. This can be done via the **short_tags()** function (PHP 3 only), by enabling the short_open_tag configuration setting in the PHP config file, or by compiling PHP with the --enable-short-tags option to **configure**. Even if it is enabled by default in php.ini-dist, use of short tags are discouraged.

The fourth way is only available if ASP-style tags have been enabled using the asp_tags configuration setting.

> **Note:** Support for ASP-style tags was added in 3.0.4.
>
> **Note:** Using short tags should be avoided when developing applications or libraries that are meant for redistribution, or deployment on PHP servers which are not under your control, because short tags may not be supported on the target server. For portable, redistributable code, be sure not to use short tags.

The closing tag for the block will include the immediately trailing newline if one is present. Also, the closing tag automatically implies a semicolon; you do not need to have a semicolon terminating the last line of a PHP block.

PHP allows you to use structures like this:

Example 6-2. Advanced escaping

```
<?php
if ($expression) {
    ?>
    <strong>This is true.</strong>
    <?php
} else {
    ?>
    <strong>This is false.</strong>
    <?php
}
?>
```

This works as expected, because when PHP hits the ?> closing tags, it simply starts outputting whatever it finds until it hits another opening tag. The example given here is contrived, of course, but for outputting large blocks of text, dropping out of PHP parsing mode is generally more efficient than sending all of the text through **echo()** or **print()** or somesuch.

Instruction separation

Instructions are separated the same as in C or Perl - terminate each statement with a semicolon.

The closing tag (?>) also implies the end of the statement, so the following are equivalent:

```
<?php
    echo "This is a test";
?>

<?php echo "This is a test" ?>
```

Comments

PHP supports 'C', 'C++' and Unix shell-style comments. For example:

```
<?php
    echo "This is a test"; // This is a one-line c++ style comment
    /* This is a multi line comment
       yet another line of comment */
    echo "This is yet another test";
    echo "One Final Test"; # This is shell-style style comment
?>
```

The "one-line" comment styles actually only comment to the end of the line or the current block of PHP code, whichever comes first.

```
<h1>This is an <?php # echo "simple";?> example.</h1>
<p>The header above will say 'This is an example'.
```

You should be careful not to nest 'C' style comments, which can happen when commenting out large blocks.

```
<?php
 /*
    echo "This is a test"; /* This comment will cause a problem */
 */
?>
```

The one-line comment styles actually only comment to the end of the line or the current block of PHP code, whichever comes first. This means that HTML code after // ?> WILL be printed: ?> skips out of the PHP mode and returns to HTML mode, and // cannot influence that.

Chapter 7. Types

Introduction

PHP supports eight primitive types.

Four scalar types:

- boolean
- integer
- floating-point number (float)
- string

Two compound types:

- array
- object

And finally two special types:

- resource
- NULL

This manual also introduces some pseudo-types for readability reasons:

- mixed
- number
- callback

You may also find some references to the type "double". Consider double the same as float, the two names exist only for historic reasons.

The type of a variable is usually not set by the programmer; rather, it is decided at runtime by PHP depending on the context in which that variable is used.

> **Note:** If you want to check out the type and value of a certain expression, use **var_dump()**.
>
> **Note:** If you simply want a human-readable representation of the type for debugging, use **gettype()**. To check for a certain type, do *not* use **gettype()**, but use the `is_type` functions. Some examples:

```
<?php
$bool = TRUE;   // a boolean
$str  = "foo";  // a string
$int  = 12;     // an integer

echo gettype($bool); // prints out "boolean"
echo gettype($str);  // prints out "string"
```

```
// If this is an integer, increment it by four
if (is_int($int)) {
    $int += 4;
}

// If $bool is a string, print it out
// (does not print out anything)
if (is_string($bool)) {
    echo "String: $bool";
}
?>
```

If you would like to force a variable to be converted to a certain type, you may either cast the variable or use the **settype()** function on it.

Note that a variable may be evaluated with different values in certain situations, depending on what type it is at the time. For more information, see the section on Type Juggling.

Booleans

This is the easiest type. A **boolean** expresses a truth value. It can be either **TRUE** or **FALSE**.

> **Note:** The boolean type was introduced in PHP 4.

Syntax

To specify a boolean literal, use either the keyword **TRUE** or **FALSE**. Both are case-insensitive.

```
<?php
$foo = True; // assign the value TRUE to $foo
?>
```

Usually you use some kind of operator which returns a **boolean** value, and then pass it on to a control structure.

```
<?php
// == is an operator which test
// equality and returns a boolean
if ($action == "show_version") {
    echo "The version is 1.23";
}

// this is not necessary...
if ($show_separators == TRUE) {
    echo "<hr>\n";
}

// ...because you can simply type
if ($show_separators) {
    echo "<hr>\n";
}
?>
```

Converting to boolean

To explicitly convert a value to **boolean**, use either the `(bool)` or the `(boolean)` cast. However, in most cases you do not need to use the cast, since a value will be automatically converted if an operator, function or control structure requires a **boolean** argument.

See also Type Juggling.

When converting to **boolean**, the following values are considered **FALSE**:

- the boolean **FALSE** itself
- the integer 0 (zero)
- the float 0.0 (zero)
- the empty string, and the string "0"
- an array with zero elements

- an object with zero member variables
- the special type NULL (including unset variables)

Every other value is considered **TRUE** (including any resource).

Warning
`-1` is considered **TRUE**, like any other non-zero (whether negative or positive) number!

```
<?php
echo gettype((bool) "");        // bool(false)
echo gettype((bool) 1);         // bool(true)
echo gettype((bool) -2);        // bool(true)
echo gettype((bool) "foo");     // bool(true)
echo gettype((bool) 2.3e5);     // bool(true)
echo gettype((bool) array(12)); // bool(true)
echo gettype((bool) array());   // bool(false)
?>
```

Integers

An **integer** is a number of the set Z = {..., -2, -1, 0, 1, 2, ...}.

See also: Arbitrary length integers and Floating point numbers

---

Syntax

Integers can be specified in decimal (10-based), hexadecimal (16-based) or octal (8-based) notation, optionally preceded by a sign (- or +).

If you use the octal notation, you must precede the number with a `0` (zero), to use hexadecimal notation precede the number with `0x`.

Example 7-1. Integer literals

```
<?php
$a = 1234; # decimal number
$a = -123; # a negative number
$a = 0123; # octal number (equivalent to 83 decimal)
$a = 0x1A; # hexadecimal number (equivalent to 26 decimal)
?>
```

Formally the possible structure for integer literals is:

```
<?php
decimal     : [1-9][0-9]*
            | 0

hexadecimal : 0[xX][0-9a-fA-F]+

octal       : 0[0-7]+

integer     : [+-]?decimal
            | [+-]?hexadecimal
            | [+-]?octal
?>
```

The size of an integer is platform-dependent, although a maximum value of about two billion is the usual value (that's 32 bits signed). PHP does not support unsigned integers.

Integer overflow

If you specify a number beyond the bounds of the **integer** type, it will be interpreted as a **float** instead. Also, if you perform an operation that results in a number beyond the bounds of the **integer** type, a **float** will be returned instead.

```
<?php
$large_number =  2147483647;
var_dump($large_number);
// output: int(2147483647)

$large_number =  2147483648;
var_dump($large_number);
```

```
// output: float(2147483648)

// this goes also for hexadecimal specified integers:
var_dump( 0x80000000 );
// output: float(2147483648)

$million = 1000000;
$large_number =  50000 * $million;
var_dump($large_number);
// output: float(50000000000)
?>
```

Warning
Unfortunately, there was a bug in PHP so that this does not always work correctly when there are negative numbers involved. For example: when you do `-50000 * $million`, the result will be `-429496728`. However, when both operands are positive there is no problem. This is solved in PHP 4.1.0.

There is no integer division operator in PHP. `1/2` yields the **float** `0.5`. You can cast the value to an integer to always round it downwards, or you can use the **round()** function.

```
<?php
var_dump(25/7);         // float(3.5714285714286)
var_dump((int) (25/7)); // int(3)
var_dump(round(25/7));  // float(4)
?>
```

Converting to integer

To explicitly convert a value to **integer**, use either the `(int)` or the `(integer)` cast. However, in most cases you do not need to use the cast, since a value will be automatically converted if an operator, function or control structure requires an **integer** argument. You can also convert a value to integer with the function **intval()**.

See also type-juggling.

From booleans

FALSE will yield `0` (zero), and **TRUE** will yield `1` (one).

From floating point numbers

When converting from float to integer, the number will be rounded *towards zero*.

If the float is beyond the boundaries of integer (usually `+/- 2.15e+9 = 2^31`), the result is undefined, since the float hasn't got enough precision to give an exact integer result. No warning, not even a notice will be issued in this case!

Warning

Never cast an unknown fraction to **integer**, as this can sometimes lead to unexpected results.
 	
`<?php`
`echo (int) ( (0.1+0.7) * 10 ); // echoes 7!`
`?>`
 	
See for more information the warning about float-precision.

From strings

See String conversion to numbers

From other types

Caution
Behaviour of converting to integer is undefined for other types. Currently, the behaviour is the same as if the value was first converted to boolean. However, do *not* rely on this behaviour, as it can change without notice.

Floating point numbers

Floating point numbers (AKA "floats", "doubles" or "real numbers") can be specified using any of the following syntaxes:

```
<?php
$a = 1.234;
$b = 1.2e3;
$c = 7E-10;
?>
```

Formally:

```
LNUM          [0-9]+
DNUM          ([0-9]*[\.]{LNUM}) | ({LNUM}[\.][0-9]*)
EXPONENT_DNUM ( ({LNUM} | {DNUM}) [eE][+-]? {LNUM})
```

The size of a float is platform-dependent, although a maximum of ~1.8e308 with a precision of roughly 14 decimal digits is a common value (that's 64 bit IEEE format).

Floating point precision
It is quite usual that simple decimal fractions like `0.1` or `0.7` cannot be converted into their internal binary counterparts without a little loss of precision. This can lead to confusing results: for example, `floor((0.1+0.7)*10)` will usually return `7` instead of the expected `8` as the result of the internal representation really being something like `7.9999999999...` This is related to the fact that it is impossible to exactly express some fractions in decimal notation with a finite number of digits. For instance, `1/3` in decimal form becomes `0.3333333. . ..` So never trust floating number results to the last digit and never compare floating point numbers for equality. If you really need higher precision, you should use the arbitrary precision math functions or gmp functions instead.

Converting to float

For information on when and how strings are converted to floats, see the section titled String conversion to numbers. For values of other types, the conversion is the same as if the value would have been converted to integer and then to float. See the Converting to integer section for more information.

Strings

A **string** is series of characters. In PHP, a character is the same as a byte, that is, there are exactly 256 different characters possible. This also implies that PHP has no native support of Unicode. See **utf8_encode()** and **utf8_decode()** for some Unicode support.

> **Note:** It is no problem for a string to become very large. There is no practical bound to the size of strings imposed by PHP, so there is no reason at all to worry about long strings.

Syntax

A string literal can be specified in three different ways.

- single quoted
- double quoted
- heredoc syntax

Single quoted

The easiest way to specify a simple string is to enclose it in single quotes (the character `'`).

To specify a literal single quote, you will need to escape it with a backslash (`\`), like in many other languages. If a backslash needs to occur before a single quote or at the end of the string, you need to double it. Note that if you try to escape any other character, the backslash will also be printed! So usually there is no need to escape the backslash itself.

> **Note:** In PHP 3, a warning will be issued at the `E_NOTICE` level when this happens.

> **Note:** Unlike the two other syntaxes, variables will *not* be expanded when they occur in single quoted strings.

```
<?php
echo 'this is a simple string';

echo 'You can also have embedded newlines in
strings this way as it is
okay to do';

// Outputs: "I'll be back"
echo 'Arnold once said: "I\'ll be back"';

// Outputs: You deleted C:\*.*?
echo 'You deleted C:\\*.*?';

// Outputs: You deleted C:\*.*?
echo 'You deleted C:\*.*?';

// Outputs: This will not expand: \n a newline
echo 'This will not expand: \n a newline';

// Outputs: Variables do not $expand $either
echo 'Variables do not $expand $either';
?>
```

Double quoted

If the string is enclosed in double-quotes ("), PHP understands more escape sequences for special characters:

Table 7-1. Escaped characters

sequence	meaning
`\n`	linefeed (LF or 0x0A (10) in ASCII)
`\r`	carriage return (CR or 0x0D (13) in ASCII)
`\t`	horizontal tab (HT or 0x09 (9) in ASCII)
`\\`	backslash
`\$`	dollar sign
`\"`	double-quote
`\[0-7]{1,3}`	the sequence of characters matching the regular expression is a character in octal notation
`\x[0-9A-Fa-f]{1,2}`	the sequence of characters matching the regular expression is a character in hexadecimal notation

Again, if you try to escape any other character, the backslash will be printed too!

But the most important feature of double-quoted strings is the fact that variable names will be expanded. See string parsing for details.

Heredoc

Another way to delimit strings is by using heredoc syntax ("<<<"). One should provide an identifier after `<<<`, then the string, and then the same identifier to close the quotation.

The closing identifier *must* begin in the first column of the line. Also, the identifier used must follow the same naming rules as any other label in PHP: it must contain only alphanumeric characters and underscores, and must start with a non-digit character or underscore.

Warning
It is very important to note that the line with the closing identifier contains no other characters, except *possibly* a semicolon (;). That means especially that the identifier *may not be indented*, and there may not be any spaces or tabs after or before the semicolon. It's also important to realize that the first character before the closing identifier must be a newline as defined by your operating system. This is `\r` on Macintosh for example.

If this rule is broken and the closing identifier is not "clean" then it's not considered to be a closing identifier and PHP will continue looking for one. If in this case a proper closing identifier is not found then a parse error will result with the line number being at the end of the script.

Heredoc text behaves just like a double-quoted string, without the double-quotes. This means that you do not need to escape quotes in your here docs, but you can still use the escape codes listed above. Variables are expanded, but the same care must be taken when expressing complex variables inside a here doc as with strings.

Example 7-2. Heredoc string quoting example

```
<?php
$str = <<<EOD
Example of string
spanning multiple lines
using heredoc syntax.
EOD;

/* More complex example, with variables. */
class foo
{
    var $foo;
    var $bar;

    function foo()
    {
        $this->foo = 'Foo';
        $this->bar = array('Bar1', 'Bar2', 'Bar3');
    }
}

$foo = new foo();
$name = 'MyName';

echo <<<EOT
My name is "$name". I am printing some $foo->foo.
Now, I am printing some {$foo->bar[1]}.
This should print a capital 'A': \x41
EOT;
?>
```

Note: Heredoc support was added in PHP 4.

Variable parsing

When a string is specified in double quotes or with heredoc, variables are parsed within it.

There are two types of syntax, a simple one and a complex one. The simple syntax is the most common and convenient, it provides a way to parse a variable, an **array** value, or an object property.

The complex syntax was introduced in PHP 4, and can be recognised by the curly braces surrounding the expression.

Simple syntax

If a dollar sign (`$`) is encountered, the parser will greedily take as much tokens as possible to form a valid variable name. Enclose the variable name in curly braces if you want to explicitly specify the end of the name.

```
<?php
$beer = 'Heineken';
echo "$beer's taste is great"; // works, "'" is an invalid character for varnames
echo "He drank some $beers";   // won't work, 's' is a valid character for varnames
echo "He drank some ${beer}s"; // works
echo "He drank some {$beer}s"; // works
?>
```

Similarly, you can also have an **array** index or an object property parsed. With array indices, the closing square bracket (`]`) marks the end of the index. For object properties the same rules apply as to simple variables, though with object properties there doesn't exist a trick like the one with variables.

```
<?php
// These examples are specific to using arrays inside of strings.
// When outside of a string, always quote your array string keys
// and do not use {braces} when outside of strings either.

// Let's show all errors
error_reporting(E_ALL);
```

```
$fruits = array('strawberry' => 'red', 'banana' => 'yellow');

// Works but note that this works differently outside string-quotes
echo "A banana is $fruits[banana].";

// Works
echo "A banana is {$fruits['banana']}.";

// Works but PHP looks for a constant named banana first
// as described below.
echo "A banana is {$fruits[banana]}.";

// Won't work, use braces.  This results in a parse error.
echo "A banana is $fruits['banana'].";

// Works
echo "A banana is " . $fruits['banana'] . ".";

// Works
echo "This square is $square->width meters broad.";

// Won't work. For a solution, see the complex syntax.
echo "This square is $square->width00 centimeters broad.";
?>
```

For anything more complex, you should use the complex syntax.

Complex (curly) syntax

This isn't called complex because the syntax is complex, but because you can include complex expressions this way.

In fact, you can include any value that is in the namespace in strings with this syntax. You simply write the expression the same way as you would outside the string, and then include it in { and }. Since you can't escape '{', this syntax will only be recognised when the $ is immediately following the {. (Use "{\$" or "\{$" to get a literal "{$"). Some examples to make it clear:

```
<?php
// Let's show all errors
error_reporting(E_ALL);

$great = 'fantastic';

// Won't work, outputs: This is { fantastic}
echo "This is { $great}";

// Works, outputs: This is fantastic
echo "This is {$great}";
echo "This is ${great}";

// Works
echo "This square is {$square->width}00 centimeters broad.";

// Works
echo "This works: {$arr[4][3]}";

// This is wrong for the same reason as $foo[bar] is wrong
// outside a string.  In otherwords, it will still work but
// because PHP first looks for a constant named foo, it will
// throw an error of level E_NOTICE (undefined constant).
echo "This is wrong: {$arr[foo][3]}";

// Works.  When using multi-dimensional arrays, always use
// braces around arrays when inside of strings
echo "This works: {$arr['foo'][3]}";

// Works.
echo "This works: " . $arr['foo'][3];

echo "You can even write {$obj->values[3]->name}";

echo "This is the value of the var named $name: {${$name}}";
?>
```

String access by character

Characters within strings may be accessed by specifying the zero-based offset of the desired character after the string in curly braces.

> **Note:** For backwards compatibility, you can still use array-braces for the same purpose. However, this syntax is deprecated as of PHP 4.

Example 7-3. Some string examples

```
<?php
// Get the first character of a string
$str = 'This is a test.';
$first = $str{0};

// Get the third character of a string
$third = $str{2};

// Get the last character of a string.
$str = 'This is still a test.';
$last = $str{strlen($str)-1};
?>
```

Useful functions and operators

Strings may be concatenated using the '.' (dot) operator. Note that the '+' (addition) operator will not work for this. Please see String operators for more information.

There are a lot of useful functions for string modification.

See the string functions section for general functions, the regular expression functions for advanced find&replacing (in two tastes: Perl and POSIX extended).

There are also functions for URL-strings, and functions to encrypt/decrypt strings (mcrypt and mhash).

Finally, if you still didn't find what you're looking for, see also the character type functions.

Converting to string

You can convert a value to a string using the `(string)` cast, or the **strval()** function. String conversion is automatically done in the scope of an expression for you where a string is needed. This happens when you use the **echo()** or **print()** functions, or when you compare a variable value to a string. Reading the manual sections on Types and Type Juggling will make the following clearer. See also **settype()**.

A **boolean** **TRUE** value is converted to the string `"1"`, the **FALSE** value is represented as `""` (empty string). This way you can convert back and forth between boolean and string values.

An **integer** or a floating point number (**float**) is converted to a string representing the number with its digits (including the exponent part for floating point numbers).

Arrays are always converted to the string `"Array"`, so you cannot dump out the contents of an **array** with **echo()** or **print()** to see what is inside them. To view one element, you'd do something like `echo $arr['foo']`. See below for tips on dumping/viewing the entire contents.

Objects are always converted to the string `"Object"`. If you would like to print out the member variable values of an **object** for debugging reasons, read the paragraphs below. If you would like to find out the class name of which an object is an instance of, use **get_class()**.

Resources are always converted to strings with the structure `"Resource id #1"` where `1` is the unique number of the **resource** assigned by PHP during runtime. If you would like to get the type of the resource, use **get_resource_type()**.

NULL is always converted to an empty string.

As you can see above, printing out the arrays, objects or resources does not provide you any useful information about the values themselfs. Look at the functions **print_r()** and **var_dump()** for better ways to print out values for debugging.

You can also convert PHP values to strings to store them permanently. This method is called serialization, and can be done with the function **serialize()**. You can also serialize PHP values to XML structures, if you have WDDX support in your PHP setup.

String conversion to numbers

When a string is evaluated as a numeric value, the resulting value and type are determined as follows.

The string will evaluate as a **float** if it contains any of the characters '.', 'e', or 'E'. Otherwise, it will evaluate as an integer.

The value is given by the initial portion of the string. If the string starts with valid numeric data, this will be the value used. Otherwise, the value will be o (zero). Valid numeric data is an optional sign, followed by one or more digits (optionally containing a decimal point), followed by an optional exponent. The exponent is an 'e' or 'E' followed by one or more digits.

```
<?php
$foo = 1 + "10.5";                // $foo is float (11.5)
$foo = 1 + "-1.3e3";              // $foo is float (-1299)
$foo = 1 + "bob-1.3e3";           // $foo is integer (1)
$foo = 1 + "bob3";                // $foo is integer (1)
$foo = 1 + "10 Small Pigs";       // $foo is integer (11)
$foo = 4 + "10.2 Little Piggies"; // $foo is float (14.2)
$foo = "10.0 pigs " + 1;          // $foo is float (11)
$foo = "10.0 pigs " + 1.0;        // $foo is float (11)
?>
```

For more information on this conversion, see the Unix manual page for strtod(3).

If you would like to test any of the examples in this section, you can cut and paste the examples and insert the following line to see for yourself what's going on:

```
<?php
echo "\$foo==$foo; type is " . gettype ($foo) . "<br />\n";
?>
```

Do not expect to get the code of one character by converting it to integer (as you would do in C for example). Use the functions **ord()** and **chr()** to convert between charcodes and characters.

Arrays

An array in PHP is actually an ordered map. A map is a type that maps *values* to *keys*. This type is optimized in several ways, so you can use it as a real array, or a list (vector), hashtable (which is an implementation of a map), dictionary, collection, stack, queue and probably more. Because you can have another PHP-array as a value, you can also quite easily simulate trees.

Explanation of those structures is beyond the scope of this manual, but you'll find at least one example for each of those structures. For more information we refer you to external literature about this broad topic.

Syntax

Specifying with array()

An **array** can be created by the **array()** language-construct. It takes a certain number of comma-separated *key => value* pairs.

```
array( [key =>] value
     , ...
     )
// key is either string or nonnegative integer
// value can be anything


<?php
$arr = array("foo" => "bar", 12 => true);

echo $arr["foo"]; // bar
echo $arr[12];    // 1
?>
```

A key is either an **integer** or a **string**. If a key is the standard representation of an **integer**, it will be interpreted as such (i.e. "8" will be interpreted as 8, while "08" will be interpreted as "08"). There are no different indexed and associative array types in PHP, there is only one array type, which can both contain integer and string indices.

A value can be of any PHP type.

```
<?php
$arr = array("somearray" => array(6 => 5, 13 => 9, "a" => 42));

echo $arr["somearray"][6];    // 5
echo $arr["somearray"][13];   // 9
echo $arr["somearray"]["a"];  // 42
?>
```

If you omit a key, the maximum of the integer-indices is taken, and the new key will be that maximum + 1. As integers can be negative, this is also true for negative indices. Having e.g. the highest index being -6 will result in -5 being the new key. If no integer-indices exist yet, the key will be 0 (zero). If you specify a key that already has a value assigned to it, that value will be overwritten.

```
<?php
// This array is the same as ...
```

```
array(5 => 43, 32, 56, "b" => 12);

// ...this array
array(5 => 43, 6 => 32, 7 => 56, "b" => 12);
?>
```

Using **TRUE** as a key will evalute to **integer** `1` as key. Using **FALSE** as a key will evalute to **integer** `0` as key. Using `NULL` as a key will evaluate to an empty string. Using an emptry string as key will create (or overwrite) a key with an empty string and its value, it is not the same as using empty brackets.

You cannot use arrays or objects as keys. Doing so will result in a warning: `Illegal offset type`.

Creating/modifying with square-bracket syntax

You can also modify an existing array, by explicitly setting values in it.

This is done by assigning values to the array while specifying the key in brackets. You can also omit the key, add an empty pair of brackets ("`[]`") to the variable-name in that case.

```
$arr[key] = value;
$arr[] = value;
// key is either string or nonnegative integer
// value can be anything
```

If `$arr` doesn't exist yet, it will be created. So this is also an alternative way to specify an array. To change a certain value, just assign a new value to an element specified with its key. If you want to remove a key/value pair, you need to **unset()** it.

```
<?php
$arr = array(5 => 1, 12 => 2);

$arr[] = 56;    // This is the same as $arr[13] = 56;
                // at this point of the script

$arr["x"] = 42; // This adds a new element to
                // the array with key "x"

unset($arr[5]); // This removes the element from the array

unset($arr);    // This deletes the whole array
?>
```

Useful functions

There are quite some useful function for working with arrays, see the array functions section.

> **Note:** The **unset()** function allows unsetting keys of an array. Be aware that the array will NOT be reindexed. If you only use "usual integer indices" (starting from zero, increasing by one), you can achive the reindex effect by using **array_values()**.
>
> ```
> <?php
> $a = array(1 => 'one', 2 => 'two', 3 => 'three');
> unset($a[2]);
> /* will produce an array that would have been defined as
>    $a = array(1 => 'one', 3 => 'three');
>    and NOT
>    $a = array(1 => 'one', 2 =>'three');
> */
>
> $b = array_values($a);
> // Now b is array(1 => 'one', 2 =>'three')
> ?>
> ```

The foreach control structure exists specifically for arrays. It provides an easy way to traverse an array.

Array do's and don'ts

Why is `$foo[bar]` wrong?

You should always use quotes around an associative array index. For example, use $foo['bar'] and not $foo[bar]. But why is $foo[bar] wrong? You might have seen the following syntax in old scripts:

```
<?php
$foo[bar] = 'enemy';
```

```
echo $foo[bar];
// etc
?>
```

This is wrong, but it works. Then, why is it wrong? The reason is that this code has an undefined constant (bar) rather than a string ('bar' - notice the quotes), and PHP may in future define constants which, unfortunately for your code, have the same name. It works, because the undefined constant gets converted to a string of the same name automatically for backward compatibility reasons.

More examples to demonstrate this fact:

```
<?php
// Let's show all errors
error_reporting(E_ALL);

$arr = array('fruit' => 'apple', 'veggie' => 'carrot');

// Correct
print $arr['fruit'];  // apple
print $arr['veggie']; // carrot

// Incorrect.  This works but also throws a PHP error of
// level E_NOTICE because of an undefined constant named fruit
//
// Notice: Use of undefined constant fruit - assumed 'fruit' in...
print $arr[fruit];    // apple

// Let's define a constant to demonstrate what's going on.  We
// will assign value 'veggie' to a constant named fruit.
define('fruit','veggie');

// Notice the difference now
print $arr['fruit'];  // apple
print $arr[fruit];    // carrot

// The following is okay as it's inside a string.  Constants are not
// looked for within strings so no E_NOTICE error here
print "Hello $arr[fruit]";      // Hello apple

// With one exception, braces surrounding arrays within strings
// allows constants to be looked for
print "Hello {$arr[fruit]}";    // Hello carrot
print "Hello {$arr['fruit']}";  // Hello apple

// This will not work, results in a parse error such as:
// Parse error: parse error, expecting T_STRING' or T_VARIABLE' or T_NUM_STRING'
// This of course applies to using autoglobals in strings as well
print "Hello $arr['fruit']";
print "Hello $_GET['foo']";

// Concatenation is another option
print "Hello " . $arr['fruit']; // Hello apple
?>
```

When you turn **error_reporting()** up to show **E_NOTICE** level errors (such as setting it to **E_ALL**) then you will see these errors. By default, error_reporting is turned down to not show them.

As stated in the syntax section, there must be an expression between the square brackets ('[' and ']'). That means that you can write things like this:

```
<?php
echo $arr[somefunc($bar)];
?>
```

This is an example of using a function return value as the array index. PHP also knows about constants, as you may have seen the E_* ones before.

```
<?php
$error_descriptions[E_ERROR]   = "A fatal error has occured";
$error_descriptions[E_WARNING] = "PHP issued a warning";
$error_descriptions[E_NOTICE]  = "This is just an informal notice";
?>
```

Note that E_ERROR is also a valid identifier, just like bar in the first example. But the last example is in fact the same as writing:

```
<?php
$error_descriptions[1] = "A fatal error has occured";
$error_descriptions[2] = "PHP issued a warning";
$error_descriptions[8] = "This is just an informal notice";
?>
```

because E_ERROR equals 1, etc.

As we already explained in the above examples, $foo[bar] still works but is wrong. It works, because bar is due to its syntax expected to be a constant expression. However, in this case no constant with the name bar exists. PHP now assumes that you

meant `bar` literally, as the string `"bar"`, but that you forgot to write the quotes.

So why is it bad then?

At some point in the future, the PHP team might want to add another constant or keyword, or you may introduce another constant into your application, and then you get in trouble. For example, you already cannot use the words `empty` and `default` this way, since they are special reserved keywords.

> **Note:** To reiterate, inside a double-quoted **string**, it's valid to not surround array indexes with quotes so `"$foo[bar]"` is valid. See the above examples for details on why as well as the section on variable parsing in strings.

Converting to array

For any of the types: **integer**, **float**, **string**, **boolean** and **resource**, if you convert a value to an **array**, you get an array with one element (with index o), which is the scalar value you started with.

If you convert an **object** to an array, you get the properties (member variables) of that object as the array's elements. The keys are the member variable names.

If you convert a **`NULL`** value to an array, you get an empty array.

Examples

The array type in PHP is very versatile, so here will be some examples to show you the full power of arrays.

```
<?php
// this
$a = array( 'color' => 'red',
            'taste' => 'sweet',
            'shape' => 'round',
            'name'  => 'apple',
                       4        // key will be 0
          );

// is completely equivalent with
$a['color'] = 'red';
$a['taste'] = 'sweet';
$a['shape'] = 'round';
$a['name']  = 'apple';
$a[]        = 4;        // key will be 0

$b[] = 'a';
$b[] = 'b';
$b[] = 'c';
// will result in the array array(0 => 'a' , 1 => 'b' , 2 => 'c'),
// or simply array('a', 'b', 'c')
?>
```

Example 7-4. Using array()

```
<?php
// Array as (property-)map
$map = array( 'version'    => 4,
              'OS'         => 'Linux',
              'lang'       => 'english',
              'short_tags' => true
            );

// strictly numerical keys
$array = array( 7,
                8,
                0,
                156,
                -10
              );
// this is the same as array(0 => 7, 1 => 8, ...)

$switching = array(         10, // key = 0
                    5    =>  6,
                    3    =>  7,
                    'a'  =>  4,
                             11, // key = 6 (maximum of integer-indices was 5)
                    '8'  =>  2, // key = 8 (integer!)
                    '02' => 77, // key = '02'
                    0    => 12  // the value 10 will be overwritten by 12
                  );
```

```
// empty array
$empty = array();
?>
```

Example 7-5. Collection

```
<?php
$colors = array('red', 'blue', 'green', 'yellow');

foreach ($colors as $color) {
    echo "Do you like $color?\n";
}

/* output:
Do you like red?
Do you like blue?
Do you like green?
Do you like yellow?
*/
?>
```

Note that it is currently not possible to change the values of the array directly in such a loop. A workaround is the following:

Example 7-6. Collection

```
<?php
foreach ($colors as $key => $color) {
    // won't work:
    //$color = strtoupper($color);

    // works:
    $colors[$key] = strtoupper($color);
}
print_r($colors);

/* output:
Array
(
    [0] => RED
    [1] => BLUE
    [2] => GREEN
    [3] => YELLOW
)
*/
?>
```

This example creates a one-based array.

Example 7-7. One-based index

```
<?php
$firstquarter  = array(1 => 'January', 'February', 'March');
print_r($firstquarter);

/* output:
Array
(
    [1] => 'January'
    [2] => 'February'
    [3] => 'March'
)
*/
```

Example 7-8. Filling an array

```
// fill an array with all items from a directory
$handle = opendir('.');
while ($file = readdir($handle)) {
    $files[] = $file;
}
closedir($handle);
?>
```

Arrays are ordered. You can also change the order using various sorting-functions. See the array functions section for more information. You can count the number of items in an array using the **count()** function.

Example 7-9. Sorting array

```
<?php
sort($files);
print_r($files);
?>
```

Because the value of an array can be everything, it can also be another array. This way you can make recursive and multi-dimensional arrays.

Example 7-10. Recursive and multi-dimensional arrays

```
<?php
$fruits = array ( "fruits"  => array ( "a" => "orange",
                                       "b" => "banana",
                                       "c" => "apple"
                                     ),
                  "numbers" => array ( 1,
                                       2,
                                       3,
                                       4,
                                       5,
                                       6,
                                     ),
                  "holes"   => array (      "first",
                                       5 => "second",
                                            "third"
                                     )
                );

// Some examples to address values in the array above
echo $fruits["holes"][5];    // prints "second"
echo $fruits["fruits"]["a"]; // prints "orange"
unset($fruits["holes"][0]);  // remove "first"

// Create a new multi-dimensional array
$juices["apple"]["green"] = "good";
?>
```

You should be aware, that array assignment always involves value copying. You need to use the reference operator to copy an array by reference.

```
<?php
$arr1 = array(2, 3);
$arr2 = $arr1;
$arr2[] = 4; // $arr2 is changed,
             // $arr1 is still array(2,3)

$arr3 = &$arr1;
$arr3[] = 4; // now $arr1 and $arr3 are the same
?>
```

Objects

Object Initialization

To initialize an object, you use the `new` statement to instantiate the object to a variable.

```
<?php
class foo
{
    function do_foo()
    {
        echo "Doing foo.";
    }
}

$bar = new foo;
$bar->do_foo();
?>
```

For a full discussion, please read the section Classes and Objects.

Converting to object

If an object is converted to an object, it is not modified. If a value of any other type is converted to an object, a new instace of the `stdClass` built in class is created. If the value was null, the new instance will be empty. For any other value, a member variable named `scalar` will contain the value.

```
<?php
$obj = (object) 'ciao';
echo $obj-&gt;scalar;  // outputs 'ciao'
?>
```

Resource

A resource is a special variable, holding a reference to an external resource. Resources are created and used by special functions. See the appendix for a listing of all these functions and the corresponding resource types.

> **Note:** The resource type was introduced in PHP 4

Converting to resource

As resource types hold special handlers to opened files, database connections, image canvas areas and the like, you cannot convert any value to a resource.

Freeing resources

Due to the reference-counting system introduced with PHP4's Zend-engine, it is automatically detected when a resource is no longer referred to (just like Java). When this is the case, all resources that were in use for this resource are made free by the garbage collector. For this reason, it is rarely ever necessary to free the memory manually by using some free_result function.

> **Note:** Persistent database links are special, they are *not* destroyed by the garbage collector. See also the section about persistent connections.

NULL

The special **NULL** value represents that a variable has no value. **NULL** is the only possible value of type NULL.

> **Note:** The null type was introduced in PHP 4

A variable is considered to be **NULL** if

- it has been assigned the constant **NULL**.
- it has not been set to any value yet.
- it has been **unset()**.

Syntax

There is only one value of type **NULL**, and that is the case-insensitive keyword **NULL**.

```
<?php
$var = NULL;

?>
```

See also **is_null()** and **unset()**.

Pseudo-types used in this documentation

mixed

`mixed` indicates that a parameter may accept multiple (but not necesseraly all) types.

gettype() for example will accept all PHP types, while **str_replace()** will accept strings and arrays.

number

`number` indicates that a parameter can be either **integer** or **float**.

---

callback

Some functions like **call_user_function()** or **usort()** accept user defined callback functions as a parameter. Callback functions can not only be simple functions but also object methods including static class methods.

A PHP function is simply passed by its name as a string. You can pass any builtin or user defined function with the exception of **array()**, **echo()**, **empty()**, **eval()**, **exit()**, **isset()**, **list()**, **print()** and **unset()**.

A method of an instantiated object is passed as an array containing an object as the element with index 0 and a method name as the element with index 1.

Static class methods can also be passed without instantiating an object of that class by passing the class name instead of an object as the element with index 0.

Example 7-11. Callback function examples

```
<?php

// simple callback example
function foobar() {
    echo "hello world!";
}
call_user_function("foobar");

// method callback examples
class foo {
  function bar() {
    echo "hello world!";
  }
}

$foo = new foo;

call_user_function(array($foo, "bar")); // object method call

call_user_function(array("foo", "bar")); // static class method call

?>
```

Type Juggling

PHP does not require (or support) explicit type definition in variable declaration; a variable's type is determined by the context in which that variable is used. That is to say, if you assign a string value to variable *$var*, *$var* becomes a string. If you then assign an integer value to *$var*, it becomes an integer.

An example of PHP's automatic type conversion is the addition operator '+'. If any of the operands is a float, then all operands are evaluated as floats, and the result will be a float. Otherwise, the operands will be interpreted as integers, and the result will also be an integer. Note that this does NOT change the types of the operands themselves; the only change is in how the operands are evaluated.

```
<?php
$foo = "0";  // $foo is string (ASCII 48)
<!-- bad example, no real operator (must be used with variable, modifies it too)
$foo++;      // $foo is the string "1" (ASCII 49)
-->
$foo += 2;   // $foo is now an integer (2)
$foo = $foo + 1.3;  // $foo is now a float (3.3)
$foo = 5 + "10 Little Piggies"; // $foo is integer (15)
$foo = 5 + "10 Small Pigs";     // $foo is integer (15)
<!--

TODO: explain ++/- - behaviour with strings

examples:

++'001' = '002'
++'abc' = 'abd'
++'xyz' = 'xza'
++'9.9' = '9.0'
++'-3'  = '-4'
- -'9'   = 8 (integer!)
- -'5.5' = '5.5'
- -'-9'  = -10 (integer)
- -'09'  = 8 (integer)
- -'abc' = 'abc'
```

```
-->
?>
```

If the last two examples above seem odd, see String conversion to numbers.

If you wish to force a variable to be evaluated as a certain type, see the section on Type casting. If you wish to change the type of a variable, see **settype()**.

If you would like to test any of the examples in this section, you can use the **var_dump()** function.

> **Note:** The behaviour of an automatic conversion to array is currently undefined.
>
> ```
> <?php
> $a = "1";     // $a is a string
> $a[0] = "f";  // What about string offsets? What happens?
> ?>
> ```
>
> Since PHP (for historical reasons) supports indexing into strings via offsets using the same syntax as array indexing, the example above leads to a problem: should $a become an array with its first element being "f", or should "f" become the first character of the string $a?
>
> The current versions of PHP interpret the second assignment as a string offset identification, so $a becomes "f", the result of this automatic conversion however should be considered undefined. PHP 4 introduced the new curly bracket syntax to access characters in string, use this syntax instead of the one presented above:
>
> ```
> <?php
> $a    = "abc"; // $a is a string
> $a{1} = "f";   // $a is now "afc"
> ?>
> ```
>
> See the section titled String access by character for more informaton.

Type Casting

Type casting in PHP works much as it does in C: the name of the desired type is written in parentheses before the variable which is to be cast.

```
<?php
$foo = 10;   // $foo is an integer
$bar = (boolean) $foo;   // $bar is a boolean
?>
```

The casts allowed are:

- (int), (integer) - cast to integer
- (bool), (boolean) - cast to boolean
- (float), (double), (real) - cast to float
- (string) - cast to string
- (array) - cast to array
- (object) - cast to object

Note that tabs and spaces are allowed inside the parentheses, so the following are functionally equivalent:

```
<?php
$foo = (int) $bar;
$foo = ( int ) $bar;
?>
```

> **Note:** Instead of casting a variable to string, you can also enclose the variable in double quotes.
>
> ```
> <?php
> $foo = 10;            // $foo is an integer
> $str = "$foo";        // $str is a string
> $fst = (string) $foo; // $fst is also a string
>
> // This prints out that "they are the same"
> if ($fst === $str) {
>     echo "they are the same";
> }
> ?>
> ```

It may not be obvious exactly what will happen when casting between certain types. For more info, see these sections:

- Converting to boolean
- Converting to integer
- Converting to float
- Converting to string
- Converting to array
- Converting to object
- Converting to resource

Chapter 8. Variables

Basics

Variables in PHP are represented by a dollar sign followed by the name of the variable. The variable name is case-sensitive.

Variable names follow the same rules as other labels in PHP. A valid variable name starts with a letter or underscore, followed by any number of letters, numbers, or underscores. As a regular expression, it would be expressed thus: '[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*'

> **Note:** For our purposes here, a letter is a-z, A-Z, and the ASCII characters from 127 through 255 (0x7f-0xff).

```
<?php
$var = "Bob";
$Var = "Joe";
echo "$var, $Var";      // outputs "Bob, Joe"

$4site = 'not yet';     // invalid; starts with a number
$_4site = 'not yet';    // valid; starts with an underscore
$täyte = 'mansikka';    // valid; 'ä' is ASCII 228.
?>
```

In PHP 3, variables are always assigned by value. That is to say, when you assign an expression to a variable, the entire value of the original expression is copied into the destination variable. This means, for instance, that after assigning one variable's value to another, changing one of those variables will have no effect on the other. For more information on this kind of assignment, see the chapter on Expressions.

PHP 4 offers another way to assign values to variables: assign by reference. This means that the new variable simply references (in other words, "becomes an alias for" or "points to") the original variable. Changes to the new variable affect the original, and vice versa. This also means that no copying is performed; thus, the assignment happens more quickly. However, any speedup will likely be noticed only in tight loops or when assigning large arrays or objects.

To assign by reference, simply prepend an ampersand (&) to the beginning of the variable which is being assigned (the source variable). For instance, the following code snippet outputs 'My name is Bob' twice:

```
<?php
$foo = 'Bob';              // Assign the value 'Bob' to $foo
$bar = &$foo;              // Reference $foo via $bar.
$bar = "My name is $bar";  // Alter $bar...
echo $bar;
echo $foo;                 // $foo is altered too.
?>
```

One important thing to note is that only named variables may be assigned by reference.

```
<?php
$foo = 25;
$bar = &$foo;      // This is a valid assignment.
$bar = &(24 * 7);  // Invalid; references an unnamed expression.

function test()
{
   return 25;
}

$bar = &test();    // Invalid.
?>
```

PHP follows Perl's convention when dealing with arithmetic operations on character variables and not C's. For example, in Perl

'Z'+1 turns into 'AA', while in C 'Z'+1 turns into '[' { ord('Z') == 90, ord('[') == 91).

Example 8-1. Arithmetic Operations on Character Variables

```
<?php
$i = 'W';
for($n=0; $n<6; $n++)
  echo ++$i . "\n";

/*
  Produces the output similar to the following:

X
Y
Z
AA
AB
AC

*/
?>
```

Predefined variables

PHP provides a large number of predefined variables to any script which it runs. Many of these variables, however, cannot be fully documented as they are dependent upon which server is running, the version and setup of the server, and other factors. Some of these variables will not be available when PHP is run on the command line. For a listing of these variables, please see the section on Reserved Predefined Variables.

Warning
In PHP 4.2.0 and later, the default value for the PHP directive register_globals is *off*. This is a major change in PHP. Having register_globals *off* affects the set of predefined variables available in the global scope. For example, to get `DOCUMENT_ROOT` you'll use `$_SERVER['DOCUMENT_ROOT']` instead of `$DOCUMENT_ROOT`, or `$_GET['id']` from the URL `http://www.example.com/test.php?id=3` instead of `$id`, or `$_ENV['HOME']` instead of `$HOME`. For related information on this change, read the configuration entry for register_globals, the security chapter on Using Register Globals , as well as the PHP 4.1.0 and 4.2.0 Release Announcements. Using the available PHP Reserved Predefined Variables, like the superglobal arrays, is preferred.

From version 4.1.0 onward, PHP provides an additional set of predefined arrays containing variables from the web server (if applicable), the environment, and user input. These new arrays are rather special in that they are automatically global--i.e., automatically available in every scope. For this reason, they are often known as 'autoglobals' or 'superglobals'. (There is no mechanism in PHP for user-defined superglobals.) The superglobals are listed below; however, for a listing of their contents and further discussion on PHP predefined variables and their natures, please see the section Reserved Predefined Variables. Also, you'll notice how the older predefined variables (`$HTTP_*_VARS`) still exist.

> **Variable variables:** Superglobals cannot be used as variable variables.

If certain variables in variables_order are not set, their appropriate PHP predefined arrays are also left empty.

PHP Superglobals

$GLOBALS

> Contains a reference to every variable which is currently available within the global scope of the script. The keys of this array are the names of the global variables. `$GLOBALS` has existed since PHP 3.

$_SERVER

> Variables set by the web server or otherwise directly related to the execution environment of the current script. Analogous to the old `$HTTP_SERVER_VARS` array (which is still available, but deprecated).

$_GET

> Variables provided to the script via HTTP GET. Analogous to the old `$HTTP_GET_VARS` array (which is still available, but deprecated).

$_POST

> Variables provided to the script via HTTP POST. Analogous to the old `$HTTP_POST_VARS` array (which is still available, but deprecated).

$_COOKIE

Variables provided to the script via HTTP cookies. Analogous to the old $HTTP_COOKIE_VARS array (which is still available, but deprecated).

$_FILES

Variables provided to the script via HTTP post file uploads. Analogous to the old $HTTP_POST_FILES array (which is still available, but deprecated). See POST method uploads for more information.

$_ENV

Variables provided to the script via the environment. Analogous to the old $HTTP_ENV_VARS array (which is still available, but deprecated).

$_REQUEST

Variables provided to the script via any user input mechanism, and which therefore cannot be trusted. The presence and order of variable inclusion in this array is defined according to the variables_order configuration directive. This array has no direct analogue in versions of PHP prior to 4.1.0. See also **import_request_variables()**.

Note: When running on the command line , this will *not* include the argv and argc entries; these are present in the $_SERVER array.

$_SESSION

Variables which are currently registered to a script's session. Analogous to the old $HTTP_SESSION_VARS array (which is still available, but deprecated). See the Session handling functions section for more information.

Variable scope

The scope of a variable is the context within which it is defined. For the most part all PHP variables only have a single scope. This single scope spans included and required files as well. For example:

```
<?php
$a = 1;
include "b.inc";
?>
```

Here the $a variable will be available within the included b.inc script. However, within user-defined functions a local function scope is introduced. Any variable used inside a function is by default limited to the local function scope. For example:

```
<?php
$a = 1; /* global scope */

function Test()
{
    echo $a; /* reference to local scope variable */
}

Test();
?>
```

This script will not produce any output because the echo statement refers to a local version of the $a variable, and it has not been assigned a value within this scope. You may notice that this is a little bit different from the C language in that global variables in C are automatically available to functions unless specifically overridden by a local definition. This can cause some problems in that people may inadvertently change a global variable. In PHP global variables must be declared global inside a function if they are going to be used in that function. An example:

```
<?php
$a = 1;
$b = 2;

function Sum()
{
    global $a, $b;

    $b = $a + $b;
}

Sum();
echo $b;
?>
```

The above script will output "3". By declaring $a and $b global within the function, all references to either variable will refer to the global version. There is no limit to the number of global variables that can be manipulated by a function.

A second way to access variables from the global scope is to use the special PHP-defined $GLOBALS array. The previous example

can be rewritten as:

```
<?php
$a = 1;
$b = 2;

function Sum()
{
    $GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"];
}

Sum();
echo $b;
?>
```

The `$GLOBALS` array is an associative array with the name of the global variable being the key and the contents of that variable being the value of the array element. Notice how `$GLOBALS` exists in any scope, this is because $GLOBALS is a superglobal. Here's an example demonstrating the power of superglobals:

```
<?php
function test_global()
{
    // Most predefined variables aren't "super" and require
    // 'global' to be available to the functions local scope.
    global $HTTP_POST_VARS;

    print $HTTP_POST_VARS['name'];

    // Superglobals are available in any scope and do
    // not require 'global'.  Superglobals are available
    // as of PHP 4.1.0
    print $_POST['name'];
}
?>
```

Another important feature of variable scoping is the *static* variable. A static variable exists only in a local function scope, but it does not lose its value when program execution leaves this scope. Consider the following example:

```
<?php
function Test ()
{
    $a = 0;
    echo $a;
    $a++;
}
?>
```

This function is quite useless since every time it is called it sets `$a` to `0` and prints "0". The `$a++` which increments the variable serves no purpose since as soon as the function exits the `$a` variable disappears. To make a useful counting function which will not lose track of the current count, the `$a` variable is declared static:

```
<?php
function Test()
{
    static $a = 0;
    echo $a;
    $a++;
}
?>
```

Now, every time the Test() function is called it will print the value of `$a` and increment it.

Static variables also provide one way to deal with recursive functions. A recursive function is one which calls itself. Care must be taken when writing a recursive function because it is possible to make it recurse indefinitely. You must make sure you have an adequate way of terminating the recursion. The following simple function recursively counts to 10, using the static variable `$count` to know when to stop:

```
<?php
function Test()
{
    static $count = 0;

    $count++;
    echo $count;
    if ($count < 10) {
        Test ();
    }
    $count--;
}
?>
```

The Zend Engine 1, driving PHP4, implements the `static` and `global` modifier for variables in terms of references. For example, a true global variable imported inside a function scope with the `global` statement actually creates a reference to the global variable. This can lead to unexpected behaviour which the following example addresses:

```
<?php
function test_global_ref() {
    global $obj;
    $obj = &new stdclass;
}

function test_global_noref() {
    global $obj;
    $obj = new stdclass;
}

test_global_ref();
var_dump($obj);
test_global_noref();
var_dump($obj);
?>
```

Executing this example will result in the following output:

```
NULL
object(stdClass)(0) {
}
```

A similar behaviour applies to the `static` statement. References are not stored statically:

```
<?php
function &get_instance_ref() {
    static $obj;

    echo "Static object: ";
    var_dump($obj);
    if (!isset($obj)) {
        // Assign a reference to the static variable
        $obj = &new stdclass;
    }
    $obj->property++;
    return $obj;
}

function &get_instance_noref() {
    static $obj;

    echo "Static object: ";
    var_dump($obj);
    if (!isset($obj)) {
        // Assign the object to the static variable
        $obj = new stdclass;
    }
    $obj->property++;
    return $obj;
}

$obj1 = get_instance_ref();
$still_obj1 = get_instance_ref();
echo "\n";
$obj2 = get_instance_noref();
$still_obj2 = get_instance_noref();
?>
```

Executing this example will result in the following output:

```
Static object: NULL
Static object: NULL

Static object: NULL
Static object: object(stdClass)(1) {
  ["property"]=>
  int(1)
}
```

This example demonstrates that when assigning a reference to a static variable, it's not *remembered* when you call the `&get_instance_ref()` function a second time.

Variable variables

Sometimes it is convenient to be able to have variable variable names. That is, a variable name which can be set and used dynamically. A normal variable is set with a statement such as:

```
<?php
$a = "hello";
?>
```

A variable variable takes the value of a variable and treats that as the name of a variable. In the above example, *hello*, can be

used as the name of a variable by using two dollar signs. i.e.

```
<?php
$$a = "world";
?>
```

At this point two variables have been defined and stored in the PHP symbol tree: `$a` with contents "hello" and `$hello` with contents "world". Therefore, this statement:

```
<?php
echo "$a ${$a}";
?>
```

produces the exact same output as:

```
<?php
echo "$a $hello";
?>
```

i.e. they both produce: `hello world`.

In order to use variable variables with arrays, you have to resolve an ambiguity problem. That is, if you write `$$a[1]` then the parser needs to know if you meant to use `$a[1]` as a variable, or if you wanted `$$a` as the variable and then the [1] index from that variable. The syntax for resolving this ambiguity is: `${$a[1]}` for the first case and `${$a}[1]` for the second.

Warning
Please note that variable variables cannot be used with PHP's Superglobal arrays. This means you cannot do things like `${$_GET}`. If you are looking for a way to handle availability of superglobals and the old `HTTP_*_VARS`, you might want to try referencing them.

Variables from outside PHP

HTML Forms (GET and POST)

When a form is submitted to a PHP script, the information from that form is automatically made available to the script. There are many ways to access this information, for example:

Example 8-2. A simple HTML form

```
<form action="foo.php" method="post">
    Name:  <input type="text" name="username"><br>
    Email: <input type="text" name="email"><br>
    <input type="submit" name="submit" value="Submit me!">
</form>
```

Depending on your particular setup and personal preferences, there are many ways to access data from your HTML forms. Some examples are:

Example 8-3. Accessing data from a simple POST HTML form

```
<?php
// Available since PHP 4.1.0

   print $_POST['username'];
   print $_REQUEST['username'];

   import_request_variables('p', 'p_');
   print $p_username;

// Available since PHP 3.

   print $HTTP_POST_VARS['username'];

// Available if the PHP directive register_globals = on.  As of
// PHP 4.2.0 the default value of register_globals = off.
// Using/relying on this method is not preferred.

   print $username;
?>
```

Using a GET form is similar except you'll use the appropriate GET predefined variable instead. GET also applies to the QUERY_STRING (the information after the '?' in an URL). So, for example, `http://www.example.com/test.php?id=3` contains GET data which is accessible with `$_GET['id']`. See also $_REQUEST and **import_request_variables()**.

Note: Superglobal arrays, like $_POST and $_GET, became available in PHP 4.1.0

As shown, before PHP 4.2.0 the default value for register_globals was *on*. And, in PHP 3 it was always on. The PHP community is encouraging all to not rely on this directive as it's preferred to assume it's *off* and code accordingly.

Note: The magic_quotes_gpc configuration directive affects Get, Post and Cookie values. If turned on, value (It's "PHP!") will automagically become (It\'s \"PHP!\"). Escaping is needed for DB insertion. See also **addslashes()**, **stripslashes()** and magic_quotes_sybase.

PHP also understands arrays in the context of form variables (see the related faq). You may, for example, group related variables together, or use this feature to retrieve values from a multiple select input. For example, let's post a form to itself and upon submission display the data:

Example 8-4. More complex form variables

```
<?php
if ($HTTP_POST_VARS['action'] == 'submitted') {
    print '<pre>';

    print_r($HTTP_POST_VARS);
    print '<a href="'. $HTTP_SERVER_VARS['PHP_SELF'] .'">Please try again</a>';

    print '</pre>';
} else {
?>
<form action="<?php echo $HTTP_SERVER_VARS['PHP_SELF']; ?>" method="post">
    Name:  <input type="text" name="personal[name]"><br>
    Email: <input type="text" name="personal[email]"><br>
    Beer: <br>
    <select multiple name="beer[]">
        <option value="warthog">Warthog</option>
        <option value="guinness">Guinness</option>
        <option value="stuttgarter">Stuttgarter Schwabenbräu</option>
    </select><br>
    <input type="hidden" name="action" value="submitted">
    <input type="submit" name="submit" value="submit me!">
</form>
<?php
}
?>
```

In PHP 3, the array form variable usage is limited to single-dimensional arrays. In PHP 4, no such restriction applies.

IMAGE SUBMIT variable names

When submitting a form, it is possible to use an image instead of the standard submit button with a tag like:

```
<input type="image" src="image.gif" name="sub">
```

When the user clicks somewhere on the image, the accompanying form will be transmitted to the server with two additional variables, sub_x and sub_y. These contain the coordinates of the user click within the image. The experienced may note that the actual variable names sent by the browser contains a period rather than an underscore, but PHP converts the period to an underscore automatically.

HTTP Cookies

PHP transparently supports HTTP cookies as defined by Netscape's Spec. Cookies are a mechanism for storing data in the remote browser and thus tracking or identifying return users. You can set cookies using the **setcookie()** function. Cookies are part of the HTTP header, so the SetCookie function must be called before any output is sent to the browser. This is the same restriction as for the **header()** function. Cookie data is then available in the appropriate cookie data arrays, such as $_COOKIE, $HTTP_COOKIE_VARS as well as in $_REQUEST. See the **setcookie()** manual page for more details and examples.

If you wish to assign multiple values to a single cookie variable, you may assign it as an array. For example:

```
<?php
  setcookie("MyCookie[foo]", "Testing 1", time()+3600);
  setcookie("MyCookie[bar]", "Testing 2", time()+3600);
?>
```

That will create two seperate cookies although MyCookie will now be a single array in your script. If you want to set just one cookie with multiple values, consider using **serialize()** or **explode()** on the value first.

Note that a cookie will replace a previous cookie by the same name in your browser unless the path or domain is different. So,

for a shopping cart application you may want to keep a counter and pass this along. i.e.

Example 8-5. A setcookie() example

```
<?php
$count++;
setcookie("count", $count, time()+3600);
setcookie("Cart[$count]", $item, time()+3600);
?>
```

Dots in incoming variable names

Typically, PHP does not alter the names of variables when they are passed into a script. However, it should be noted that the dot (period, full stop) is not a valid character in a PHP variable name. For the reason, look at it:

```
<?php
$varname.ext;  /* invalid variable name */
?>
```

Now, what the parser sees is a variable named `$varname`, followed by the string concatenation operator, followed by the barestring (i.e. unquoted string which doesn't match any known key or reserved words) 'ext'. Obviously, this doesn't have the intended result.

For this reason, it is important to note that PHP will automatically replace any dots in incoming variable names with underscores.

Determining variable types

Because PHP determines the types of variables and converts them (generally) as needed, it is not always obvious what type a given variable is at any one time. PHP includes several functions which find out what type a variable is, such as: **gettype()**, **is_array()**, **is_float()**, **is_int()**, **is_object()**, and **is_string()**. See also the chapter on Types.

Chapter 9. Constants

A constant is a identifier (name) for a simple value. As the name suggests, that value cannot change during the execution of the script (except the magic constants which aren't actually constants). A constant is case-sensitive by default. By convention constant identifiers are always uppercase.

The name of a constant follows the same rules as any label in PHP. A valid constant name starts with a letter or underscore, followed by any number of letters, numbers, or underscores. As a regular expression, it would be expressed thus:
`[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*`

> **Note:** For our purposes here, a letter is a-z, A-Z, and the ASCII characters from 127 through 255 (0x7f-0xff).

The scope of a constant is global--you can access it anywhere in your script without regard to scope.

Syntax

You can define a constant by using the **define()**-function. Once a constant is defined, it can never be changed or undefined.

Only scalar data (**boolean**, **integer**, **float** and **string**) can be contained in constants.

You can get the value of a constant by simply specifying its name. Unlike with variables, you should *not* prepend a constant with a `$`. You can also use the function **constant()**, to read a constant's value, if you are to obtain the constant's name dynamically. Use **get_defined_constants()** to get a list of all defined constants.

> **Note:** Constants and (global) variables are in a different namespace. This implies that for example **TRUE** and `$TRUE` are generally different.

If you use an undefined constant, PHP assumes that you mean the name of the constant itself. A notice will be issued when this happens. Use the **defined()**-function if you want to know if a constant is set.

These are the differences between constants and variables:

- Constants do not have a dollar sign ($) before them;

- Constants may only be defined using the **define()** function, not by simple assignment;
- Constants may be defined and accessed anywhere without regard to variable scoping rules;
- Constants may not be redefined or undefined once they have been set; and
- Constants may only evaluate to scalar values.

Example 9-1. Defining Constants

```
<?php
define("CONSTANT", "Hello world.");
echo CONSTANT; // outputs "Hello world."
echo Constant; // outputs "Constant" and issues a notice.
?>
```

Predefined constants

PHP provides a large number of predefined constants to any script which it runs. Many of these constants, however, are created by various extensions, and will only be present when those extensions are available, either via dynamic loading or because they have been compiled in.

There are four magical constants that change depending on where they're used. For example, the value of **__LINE__** depends on the line that it's used on in your script. These special constants are case-insensitive and are as follows:

Table 9-1. A few "magical" PHP "constants"

Name	Description
__LINE__	The current line number of the file.
__FILE__	The full path and filename of the file.
__FUNCTION__	The function name. This was added in PHP 4.3.0
__CLASS__	The class name. This was added in PHP 4.3.0

A list of predefined constants is available in the section Reserved predefined constants.

Chapter 10. Expressions

Expressions are the most important building stones of PHP. In PHP, almost anything you write is an expression. The simplest yet most accurate way to define an expression is "anything that has a value".

The most basic forms of expressions are constants and variables. When you type "$a = 5", you're assigning '5' into $a. '5', obviously, has the value 5, or in other words '5' is an expression with the value of 5 (in this case, '5' is an integer constant).

After this assignment, you'd expect $a's value to be 5 as well, so if you wrote $b = $a, you'd expect it to behave just as if you wrote $b = 5. In other words, $a is an expression with the value of 5 as well. If everything works right, this is exactly what will happen.

Slightly more complex examples for expressions are functions. For instance, consider the following function:

```
function foo ()
{
    return 5;
}
```

Assuming you're familiar with the concept of functions (if you're not, take a look at the chapter about functions), you'd assume that typing `$c = foo()` is essentially just like writing `$c = 5`, and you're right. Functions are expressions with the value of their return value. Since foo() returns 5, the value of the expression 'foo()' is 5. Usually functions don't just return a static value but compute something.

Of course, values in PHP don't have to be integers, and very often they aren't. PHP supports three scalar value types: integer values, floating point values and string values (scalar values are values that you can't 'break' into smaller pieces, unlike arrays, for instance). PHP also supports two composite (non-scalar) types: arrays and objects. Each of these value types can be assigned into variables or returned from functions.

So far, users of PHP/FI 2 shouldn't feel any change. However, PHP takes expressions much further, in the same way many other

languages do. PHP is an expression-oriented language, in the sense that almost everything is an expression. Consider the example we've already dealt with, '$a = 5'. It's easy to see that there are two values involved here, the value of the integer constant '5', and the value of $a which is being updated to 5 as well. But the truth is that there's one additional value involved here, and that's the value of the assignment itself. The assignment itself evaluates to the assigned value, in this case 5. In practice, it means that '$a = 5', regardless of what it does, is an expression with the value 5. Thus, writing something like '$b = ($a = 5)' is like writing '$a = 5; $b = 5;' (a semicolon marks the end of a statement). Since assignments are parsed in a right to left order, you can also write '$b = $a = 5'.

Another good example of expression orientation is pre- and post-increment and decrement. Users of PHP/FI 2 and many other languages may be familiar with the notation of variable++ and variable--. These are increment and decrement operators. In PHP/FI 2, the statement '$a++' has no value (is not an expression), and thus you can't assign it or use it in any way. PHP enhances the increment/decrement capabilities by making these expressions as well, like in C. In PHP, like in C, there are two types of increment - pre-increment and post-increment. Both pre-increment and post-increment essentially increment the variable, and the effect on the variable is idential. The difference is with the value of the increment expression. Pre-increment, which is written '++$variable', evaluates to the incremented value (PHP increments the variable before reading its value, thus the name 'pre-increment'). Post-increment, which is written '$variable++' evaluates to the original value of $variable, before it was incremented (PHP increments the variable after reading its value, thus the name 'post-increment').

A very common type of expressions are comparison expressions. These expressions evaluate to either 0 or 1, meaning **FALSE** or **TRUE** (respectively). PHP supports > (bigger than), >= (bigger than or equal to), == (equal), != (not equal), < (smaller than) and <= (smaller than or equal to). These expressions are most commonly used inside conditional execution, such as `if` statements.

The last example of expressions we'll deal with here is combined operator-assignment expressions. You already know that if you want to increment $a by 1, you can simply write '$a++' or '++$a'. But what if you want to add more than one to it, for instance 3? You could write '$a++' multiple times, but this is obviously not a very efficient or comfortable way. A much more common practice is to write '$a = $a + 3'. '$a + 3' evaluates to the value of $a plus 3, and is assigned back into $a, which results in incrementing $a by 3. In PHP, as in several other languages like C, you can write this in a shorter way, which with time would become clearer and quicker to understand as well. Adding 3 to the current value of $a can be written '$a += 3'. This means exactly "take the value of $a, add 3 to it, and assign it back into $a". In addition to being shorter and clearer, this also results in faster execution. The value of '$a += 3', like the value of a regular assignment, is the assigned value. Notice that it is NOT 3, but the combined value of $a plus 3 (this is the value that's assigned into $a). Any two-place operator can be used in this operator-assignment mode, for example '$a -= 5' (subtract 5 from the value of $a), '$b *= 7' (multiply the value of $b by 7), etc.

There is one more expression that may seem odd if you haven't seen it in other languages, the ternary conditional operator:

```
$first ? $second : $third
```

If the value of the first subexpression is **TRUE** (non-zero), then the second subexpression is evaluated, and that is the result of the conditional expression. Otherwise, the third subexpression is evaluated, and that is the value.

The following example should help you understand pre- and post-increment and expressions in general a bit better:

```
function double($i)
{
    return $i*2;
}
$b = $a = 5;        /* assign the value five into the variable $a and $b */
$c = $a++;          /* post-increment, assign original value of $a
                       (5) to $c */
$e = $d = ++$b;     /* pre-increment, assign the incremented value of
                       $b (6) to $d and $e */

/* at this point, both $d and $e are equal to 6 */

$f = double($d++);  /* assign twice the value of $d <emphasis>before</emphasis>
                       the increment, 2*6 = 12 to $f */
$g = double(++$e);  /* assign twice the value of $e <emphasis>after</emphasis>
                       the increment, 2*7 = 14 to $g */
$h = $g += 10;      /* first, $g is incremented by 10 and ends with the
                       value of 24. the value of the assignment (24) is
                       then assigned into $h, and $h ends with the value
                       of 24 as well. */
```

In the beginning of the chapter we said that we'll be describing the various statement types, and as promised, expressions can be statements. However, not every expression is a statement. In this case, a statement has the form of 'expr' ';' that is, an expression followed by a semicolon. In '$b=$a=5;', $a=5 is a valid expression, but it's not a statement by itself. '$b=$a=5;' however is a valid statement.

One last thing worth mentioning is the truth value of expressions. In many events, mainly in conditional execution and loops, you're not interested in the specific value of the expression, but only care about whether it means **TRUE** or **FALSE**. The constants **TRUE** and **FALSE** (case-insensitive) are the two possible boolean values. When necessary, an expression is automatically converted to boolean. See the section about type-casting for details about how.

PHP provides a full and powerful implementation of expressions, and documenting it entirely goes beyond the scope of this manual. The above examples should give you a good idea about what expressions are and how you can construct useful expressions. Throughout the rest of this manual we'll write *expr* to indicate any valid PHP expression.

Chapter 11. Operators

Operator Precedence

The precedence of an operator specifies how "tightly" it binds two expressions together. For example, in the expression `1 + 5 * 3`, the answer is `16` and not `18` because the multiplication ("*") operator has a higher precedence than the addition ("+") operator. Parentheses may be used to force precedence, if necessary. For instance: `(1 + 5) * 3` evaluates to `18`.

The following table lists the precedence of operators with the lowest-precedence operators listed first.

Table 11-1. Operator Precedence

Associativity	Operators
left	,
left	or
left	xor
left	and
right	print
left	= += -= *= /= .= %= &= |= ^= ~= <<= >>=
left	? :
left	||
left	&&
left	|
left	^
left	&
non-associative	== != === !==
non-associative	< <= > >=
left	<< >>
left	+ - .
left	* / %
right	! ~ ++ -- (int) (float) (string) (array) (object) @
right	[
non-associative	new

> **Note:** Although `!` has a higher precedence than `=`, PHP will still allow expressions similar to the following: `if (!$a = foo())`, in which case the output from `foo()` is put into `$a`.

Arithmetic Operators

Remember basic arithmetic from school? These work just like those.

Table 11-2. Arithmetic Operators

Example	Name	Result
$a + $b	Addition	Sum of $a and $b.
$a - $b	Subtraction	Difference of $a and $b.
$a * $b	Multiplication	Product of $a and $b.
$a / $b	Division	Quotient of $a and $b.
$a % $b	Modulus	Remainder of $a divided by $b.

The division operator ("/") returns a float value anytime, even if the two operands are integers (or strings that get converted to integers).

Assignment Operators

The basic assignment operator is "=". Your first inclination might be to think of this as "equal to". Don't. It really means that the the left operand gets set to the value of the expression on the rights (that is, "gets set to").

The value of an assignment expression is the value assigned. That is, the value of "$a = 3" is 3. This allows you to do some tricky things:

```
$a = ($b = 4) + 5; // $a is equal to 9 now, and $b has been set to 4.
```

In addition to the basic assignment operator, there are "combined operators" for all of the binary arithmetic and string operators that allow you to use a value in an expression and then set its value to the result of that expression. For example:

```
$a = 3;
$a += 5; // sets $a to 8, as if we had said: $a = $a + 5;
$b = "Hello ";
$b .= "There!"; // sets $b to "Hello There!", just like $b = $b . "There!";
```

Note that the assignment copies the original variable to the new one (assignment by value), so changes to one will not affect the other. This may also have relevance if you need to copy something like a large array inside a tight loop. PHP 4 supports assignment by reference, using the `$var = &$othervar;` syntax, but this is not possible in PHP 3. 'Assignment by reference' means that both variables end up pointing at the same data, and nothing is copied anywhere. To learn more about references, please read References explained.

Bitwise Operators

Bitwise operators allow you to turn specific bits within an integer on or off. If both the left- and right-hand parameters are strings, the bitwise operator will operate on the characters in this string.

```
<?php
    echo 12 ^ 9; // Outputs '5'

    echo "12" ^ "9"; // Outputs the Backspace character (ascii 8)
                     // ('1' (ascii 49)) ^ ('9' (ascii 57)) = #8

    echo "hallo" ^ "hello"; // Outputs the ascii values #0 #4 #0 #0 #0
                            // 'a' ^ 'e' = #4
?>
```

Table 11-3. Bitwise Operators

Example	Name	Result
$a & $b	And	Bits that are set in both $a and $b are set.
$a | $b	Or	Bits that are set in either $a or $b are set.
$a ^ $b	Xor	Bits that are set in $a or $b but not both are set.
~ $a	Not	Bits that are set in $a are not set, and vice versa.
$a << $b	Shift left	Shift the bits of $a $b steps to the left (each step means "multiply by two")
$a >> $b	Shift right	Shift the bits of $a $b steps to the right (each step means "divide by two")

Comparison Operators

Comparison operators, as their name implies, allow you to compare two values.

Table 11-4. Comparison Operators

Example	Name	Result
$a == $b	Equal	**TRUE** if $a is equal to $b.
$a === $b	Identical	**TRUE** if $a is equal to $b, and they are of the same type. (PHP 4 only)
$a != $b	Not equal	**TRUE** if $a is not equal to $b.
$a <> $b	Not equal	**TRUE** if $a is not equal to $b.
$a !== $b	Not identical	**TRUE** if $a is not equal to $b, or they are not of the same type. (PHP 4 only)

Example	Name	Result
$a < $b	Less than	**TRUE** if $a is strictly less than $b.
$a > $b	Greater than	**TRUE** if $a is strictly greater than $b.
$a <= $b	Less than or equal to	**TRUE** if $a is less than or equal to $b.
$a >= $b	Greater than or equal to	**TRUE** if $a is greater than or equal to $b.

Another conditional operator is the "?:" (or ternary) operator, which operates as in C and many other languages.

```
(expr1) ? (expr2) : (expr3);
```

This expression evaluates to *expr2* if *expr1* evaluates to **TRUE**, and *expr3* if *expr1* evaluates to **FALSE**.

Error Control Operators

PHP supports one error control operator: the at sign (@). When prepended to an expression in PHP, any error messages that might be generated by that expression will be ignored.

If the track_errors feature is enabled, any error message generated by the expression will be saved in the variable $php_errormsg. This variable will be overwritten on each error, so check early if you want to use it.

```
<?php
/* Intentional file error */
$my_file = @file ('non_existent_file') or
    die ("Failed opening file: error was '$php_errormsg'");

// this works for any expression, not just functions:
$value = @$cache[$key];
// will not issue a notice if the index $key doesn't exist.

?>
```

> **Note:** The @-operator works only on expressions. A simple rule of thumb is: if you can take the value of something, you can prepend the @ operator to it. For instance, you can prepend it to variables, function and **include()** calls, constants, and so forth. You cannot prepend it to function or class definitions, or conditional structures such as `if` and `foreach`, and so forth.

See also **error_reporting()**.

> **Note:** The "@" error-control operator prefix will not disable messages that are the result of parse errors.

Warning
Currently the "@" error-control operator prefix will even disable error reporting for critical errors that will terminate script execution. Among other things, this means that if you use "@" to suppress errors from a certain function and either it isn't available or has been mistyped, the script will die right there with no indication as to why.

Execution Operators

PHP supports one execution operator: backticks (``). Note that these are not single-quotes! PHP will attempt to execute the contents of the backticks as a shell command; the output will be returned (i.e., it won't simply be dumped to output; it can be assigned to a variable).

```
$output = `ls -al`;
echo "<pre>$output</pre>";
```

> **Note:** The backtick operator is disabled when safe mode is enabled or **shell_exec()** is disabled.

See also **escapeshellcmd()**, **exec()**, **passthru()**, **popen()**, **shell_exec()**, and **system()**.

Incrementing/Decrementing Operators

PHP supports C-style pre- and post-increment and decrement operators.

Table 11-5. Increment/decrement Operators

Example	Name	Effect
++$a	Pre-increment	Increments $a by one, then returns $a.
$a++	Post-increment	Returns $a, then increments $a by one.
--$a	Pre-decrement	Decrements $a by one, then returns $a.
$a--	Post-decrement	Returns $a, then decrements $a by one.

Here's a simple example script:

```
<?php
echo "<h3>Postincrement</h3>";
$a = 5;
echo "Should be 5: " . $a++ . "<br />\n";
echo "Should be 6: " . $a . "<br />\n";

echo "<h3>Preincrement</h3>";
$a = 5;
echo "Should be 6: " . ++$a . "<br />\n";
echo "Should be 6: " . $a . "<br />\n";

echo "<h3>Postdecrement</h3>";
$a = 5;
echo "Should be 5: " . $a-- . "<br />\n";
echo "Should be 4: " . $a . "<br />\n";

echo "<h3>Predecrement</h3>";
$a = 5;
echo "Should be 4: " . --$a . "<br />\n";
echo "Should be 4: " . $a . "<br />\n";
?>
```

Logical Operators

Table 11-6. Logical Operators

Example	Name	Result
$a and $b	And	**TRUE** if both $a and $b are **TRUE**.
$a or $b	Or	**TRUE** if either $a or $b is **TRUE**.
$a xor $b	Xor	**TRUE** if either $a or $b is **TRUE**, but not both.
! $a	Not	**TRUE** if $a is not **TRUE**.
$a && $b	And	**TRUE** if both $a and $b are **TRUE**.
$a || $b	Or	**TRUE** if either $a or $b is **TRUE**.

The reason for the two different variations of "and" and "or" operators is that they operate at different precedences. (See Operator Precedence.)

String Operators

There are two string operators. The first is the concatenation operator ('.'), which returns the concatenation of its right and left arguments. The second is the concatenating assignment operator ('.='), which appends the argument on the right side to the argument on the left side. Please read Assignment Operators for more information.

```
$a = "Hello ";
$b = $a . "World!"; // now $b contains "Hello World!"

$a = "Hello ";
$a .= "World!";     // now $a contains "Hello World!"
```

Array Operators

The only array operator in PHP is the + operator. It appends the right handed array to the left handed, whereas duplicated keys are NOT overwritten.

```
$a = array("a" => "apple", "b" => "banana");
$b = array("a" =>"pear", "b" => "strawberry", "c" => "cherry");
```

```
$c = $a + $b;

var_dump($c);

array(3) {
  ["a"]=>
  string(5) "apple"
  ["b"]=>
  string(6) "banana"
  ["c"]=>
  string(6) "cherry"
}
```

Chapter 12. Control Structures

Any PHP script is built out of a series of statements. A statement can be an assignment, a function call, a loop, a conditional statement of even a statement that does nothing (an empty statement). Statements usually end with a semicolon. In addition, statements can be grouped into a statement-group by encapsulating a group of statements with curly braces. A statement-group is a statement by itself as well. The various statement types are described in this chapter.

---

if

The `if` construct is one of the most important features of many languages, PHP included. It allows for conditional execution of code fragments. PHP features an `if` structure that is similar to that of C:

```
if (expr)
    statement
```

As described in the section about expressions, *expr* is evaluated to its Boolean value. If *expr* evaluates to **TRUE**, PHP will execute *statement*, and if it evaluates to **FALSE** - it'll ignore it. More information about what values evaluate to **FALSE** can be found in the 'Converting to boolean' section.

The following example would display `a is bigger than b` if *$a* is bigger than *$b*:

```
if ($a > $b)
    print "a is bigger than b";
```

Often you'd want to have more than one statement to be executed conditionally. Of course, there's no need to wrap each statement with an `if` clause. Instead, you can group several statements into a statement group. For example, this code would display `a is bigger than b` if *$a* is bigger than *$b*, and would then assign the value of *$a* into *$b*:

```
if ($a > $b) {
    print "a is bigger than b";
    $b = $a;
}
```

`If` statements can be nested indefinitely within other `if` statements, which provides you with complete flexibility for conditional execution of the various parts of your program.

---

else

Often you'd want to execute a statement if a certain condition is met, and a different statement if the condition is not met. This is what `else` is for. `else` extends an `if` statement to execute a statement in case the expression in the `if` statement evaluates to **FALSE**. For example, the following code would display `a is bigger than b` if *$a* is bigger than *$b*, and `a is NOT bigger than b` otherwise:

```
if ($a > $b) {
    print "a is bigger than b";
} else {
    print "a is NOT bigger than b";
}
```

The `else` statement is only executed if the `if` expression evaluated to **FALSE**, and if there were any `elseif` expressions - only if they evaluated to **FALSE** as well (see elseif).

---

elseif

elseif, as its name suggests, is a combination of if and else. Like else, it extends an if statement to execute a different statement in case the original if expression evaluates to FALSE. However, unlike else, it will execute that alternative expression only if the elseif conditional expression evaluates to TRUE. For example, the following code would display a is bigger than b, a equal to b or a is smaller than b:

```
if ($a > $b) {
    print "a is bigger than b";
} elseif ($a == $b) {
    print "a is equal to b";
} else {
    print "a is smaller than b";
}
```

There may be several elseifs within the same if statement. The first elseif expression (if any) that evaluates to TRUE would be executed. In PHP, you can also write 'else if' (in two words) and the behavior would be identical to the one of 'elseif' (in a single word). The syntactic meaning is slightly different (if you're familiar with C, this is the same behavior) but the bottom line is that both would result in exactly the same behavior.

The elseif statement is only executed if the preceding if expression and any preceding elseif expressions evaluated to FALSE, and the current elseif expression evaluated to TRUE.

Alternative syntax for control structures

PHP offers an alternative syntax for some of its control structures; namely, if, while, for, foreach, and switch. In each case, the basic form of the alternate syntax is to change the opening brace to a colon (:) and the closing brace to endif;, endwhile;, endfor;, endforeach;, or endswitch;, respectively.

```
<?php if ($a == 5): ?>
A is equal to 5
<?php endif; ?>
```

In the above example, the HTML block "A is equal to 5" is nested within an if statement written in the alternative syntax. The HTML block would be displayed only if $a is equal to 5.

The alternative syntax applies to else and elseif as well. The following is an if structure with elseif and else in the alternative format:

```
if ($a == 5):
    print "a equals 5";
    print "...";
elseif ($a == 6):
    print "a equals 6";
    print "!!!";
else:
    print "a is neither 5 nor 6";
endif;
```

See also while, for, and if for further examples.

---

while

while loops are the simplest type of loop in PHP. They behave just like their C counterparts. The basic form of a while statement is:

```
while (expr) statement
```

The meaning of a while statement is simple. It tells PHP to execute the nested statement(s) repeatedly, as long as the while expression evaluates to TRUE. The value of the expression is checked each time at the beginning of the loop, so even if this value changes during the execution of the nested statement(s), execution will not stop until the end of the iteration (each time PHP runs the statements in the loop is one iteration). Sometimes, if the while expression evaluates to FALSE from the very beginning, the nested statement(s) won't even be run once.

Like with the if statement, you can group multiple statements within the same while loop by surrounding a group of statements with curly braces, or by using the alternate syntax:

```
while (expr): statement ... endwhile;
```

The following examples are identical, and both print numbers from 1 to 10:

```
/* example 1 */

$i = 1;
while ($i <= 10) {
    print $i++;  /* the printed value would be
                    $i before the increment
                    (post-increment) */
}

/* example 2 */

$i = 1;
while ($i <= 10):
    print $i;
    $i++;
endwhile;
```

---

do..while

`do..while` loops are very similar to `while` loops, except the truth expression is checked at the end of each iteration instead of in the beginning. The main difference from regular `while` loops is that the first iteration of a `do..while` loop is guaranteed to run (the truth expression is only checked at the end of the iteration), whereas it's may not necessarily run with a regular `while` loop (the truth expression is checked at the beginning of each iteration, if it evaluates to **FALSE** right from the beginning, the loop execution would end immediately).

There is just one syntax for `do..while` loops:

```
$i = 0;
do {
   print $i;
} while ($i>0);
```

The above loop would run one time exactly, since after the first iteration, when truth expression is checked, it evaluates to **FALSE** ($i is not bigger than 0) and the loop execution ends.

Advanced C users may be familiar with a different usage of the `do..while` loop, to allow stopping execution in the middle of code blocks, by encapsulating them with `do..while`(0), and using the break statement. The following code fragment demonstrates this:

```
do {
    if ($i < 5) {
        print "i is not big enough";
        break;
    }
    $i *= $factor;
    if ($i < $minimum_limit) {
        break;
    }
    print "i is ok";

     ...process i...

} while(0);
```

Don't worry if you don't understand this right away or at all. You can code scripts and even powerful scripts without using this 'feature'.

---

for

`for` loops are the most complex loops in PHP. They behave like their C counterparts. The syntax of a `for` loop is:

```
for (expr1; expr2; expr3) statement
```

The first expression (*expr1*) is evaluated (executed) once unconditionally at the beginning of the loop.

In the beginning of each iteration, *expr2* is evaluated. If it evaluates to **TRUE**, the loop continues and the nested statement(s) are executed. If it evaluates to **FALSE**, the execution of the loop ends.

At the end of each iteration, *expr3* is evaluated (executed).

Each of the expressions can be empty. *expr2* being empty means the loop should be run indefinitely (PHP implicitly considers it as **TRUE**, like C). This may not be as useless as you might think, since often you'd want to end the loop using a conditional break

statement instead of using the `for` truth expression.

Consider the following examples. All of them display numbers from 1 to 10:

```
/* example 1 */

for ($i = 1; $i <= 10; $i++) {
    print $i;
}

/* example 2 */

for ($i = 1;;$i++) {
    if ($i > 10) {
        break;
    }
    print $i;
}

/* example 3 */

$i = 1;
for (;;) {
    if ($i > 10) {
        break;
    }
    print $i;
    $i++;
}

/* example 4 */

for ($i = 1; $i <= 10; print $i, $i++);
```

Of course, the first example appears to be the nicest one (or perhaps the fourth), but you may find that being able to use empty expressions in `for` loops comes in handy in many occasions.

PHP also supports the alternate "colon syntax" for `for` loops.

```
for (expr1; expr2; expr3): statement; ...; endfor;
```

Other languages have a `foreach` statement to traverse an array or hash. PHP 3 has no such construct; PHP 4 does (see foreach). In PHP 3, you can combine while with the **list()** and **each()** functions to achieve the same effect. See the documentation for these functions for an example.

`foreach`

PHP 4 (not PHP 3) includes a `foreach` construct, much like Perl and some other languages. This simply gives an easy way to iterate over arrays. `foreach` works only on arrays, and will issue an error when you try to use it on a variable with a different data type or an uninitialized variables. There are two syntaxes; the second is a minor but useful extension of the first:

```
foreach(array_expression as $value) statement
foreach(array_expression as $key => $value) statement
```

The first form loops over the array given by `array_expression`. On each loop, the value of the current element is assigned to `$value` and the internal array pointer is advanced by one (so on the next loop, you'll be looking at the next element).

The second form does the same thing, except that the current element's key will be assigned to the variable `$key` on each loop.

> **Note:** When `foreach` first starts executing, the internal array pointer is automatically reset to the first element of the array. This means that you do not need to call **reset()** before a `foreach` loop.
>
> **Note:** Also note that `foreach` operates on a copy of the specified array, not the array itself, therefore the array pointer is not modified as with the **each()** construct and changes to the array element returned are not reflected in the original array. However, the internal pointer of the original array *is* advanced with the processing of the array. Assuming the foreach loop runs to completion, the array's internal pointer will be at the end of the array.
>
> **Note:** `foreach` does not support the ability to suppress error messages using '@'.

You may have noticed that the following are functionally identical:

```
reset ($arr);
while (list(, $value) = each ($arr)) {
    echo "Value: $value<br>\n";
}

foreach ($arr as $value) {
    echo "Value: $value<br>\n";
```

```
}
```

The following are also functionally identical:

```
reset ($arr);
while (list($key, $value) = each ($arr)) {
    echo "Key: $key; Value: $value<br>\n";
}

foreach ($arr as $key => $value) {
    echo "Key: $key; Value: $value<br>\n";
}
```

Some more examples to demonstrate usages:

```
/* foreach example 1: value only */

$a = array (1, 2, 3, 17);

foreach ($a as $v) {
   print "Current value of \$a: $v.\n";
}

/* foreach example 2: value (with key printed for illustration) */

$a = array (1, 2, 3, 17);

$i = 0; /* for illustrative purposes only */

foreach($a as $v) {
    print "\$a[$i] => $v.\n";
    $i++;
}

/* foreach example 3: key and value */

$a = array (
    "one" => 1,
    "two" => 2,
    "three" => 3,
    "seventeen" => 17
);

foreach($a as $k => $v) {
    print "\$a[$k] => $v.\n";
}

/* foreach example 4: multi-dimensional arrays */

$a[0][0] = "a";
$a[0][1] = "b";
$a[1][0] = "y";
$a[1][1] = "z";

foreach($a as $v1) {
    foreach ($v1 as $v2) {
        print "$v2\n";
    }
}

/* foreach example 5: dynamic arrays */

foreach(array(1, 2, 3, 4, 5) as $v) {
    print "$v\n";
}
```

`break`

`break` ends execution of the current `for`, `foreach` `while`, `do..while` or `switch` structure.

`break` accepts an optional numeric argument which tells it how many nested enclosing structures are to be broken out of.

```
$arr = array ('one', 'two', 'three', 'four', 'stop', 'five');
while (list (, $val) = each ($arr)) {
    if ($val == 'stop') {
        break;    /* You could also write 'break 1;' here. */
    }
    echo "$val<br>\n";
}

/* Using the optional argument. */

$i = 0;
while (++$i) {
    switch ($i) {
    case 5:
        echo "At 5<br>\n";
```

```
        break 1;  /* Exit only the switch. */
    case 10:
        echo "At 10; quitting<br>\n";
        break 2;  /* Exit the switch and the while. */
    default:
        break;
    }
}
```

---

continue

`continue` is used within looping structures to skip the rest of the current loop iteration and continue execution at the beginning of the next iteration.

`continue` accepts an optional numeric argument which tells it how many levels of enclosing loops it should skip to the end of.

```
while (list ($key, $value) = each ($arr)) {
    if (!($key % 2)) { // skip odd members
        continue;
    }
    do_something_odd ($value);
}

$i = 0;
while ($i++ < 5) {
    echo "Outer<br>\n";
    while (1) {
        echo "  Middle<br>\n";
        while (1) {
            echo "  Inner<br>\n";
            continue 3;
        }
        echo "This never gets output.<br>\n";
    }
    echo "Neither does this.<br>\n";
}
```

---

switch

The `switch` statement is similar to a series of IF statements on the same expression. In many occasions, you may want to compare the same variable (or expression) with many different values, and execute a different piece of code depending on which value it equals to. This is exactly what the `switch` statement is for.

The following two examples are two different ways to write the same thing, one using a series of `if` statements, and the other using the `switch` statement:

```
if ($i == 0) {
    print "i equals 0";
}
if ($i == 1) {
    print "i equals 1";
}
if ($i == 2) {
    print "i equals 2";
}

switch ($i) {
    case 0:
        print "i equals 0";
        break;
    case 1:
        print "i equals 1";
        break;
    case 2:
        print "i equals 2";
        break;
}
```

It is important to understand how the `switch` statement is executed in order to avoid mistakes. The `switch` statement executes line by line (actually, statement by statement). In the beginning, no code is executed. Only when a `case` statement is found with a value that matches the value of the `switch` expression does PHP begin to execute the statements. PHP continues to execute the statements until the end of the `switch` block, or the first time it sees a `break` statement. If you don't write a `break` statement at the end of a case's statement list, PHP will go on executing the statements of the following case. For example:

```
switch ($i) {
    case 0:
        print "i equals 0";
    case 1:
        print "i equals 1";
```

```
    case 2:
        print "i equals 2";
}
```

Here, if $i is equal to 0, PHP would execute all of the print statements! If $i is equal to 1, PHP would execute the last two print statements. You would get the expected behavior ('i equals 2' would be displayed) only if $i is equal to 2. Thus, it is important not to forget `break` statements (even though you may want to avoid supplying them on purpose under certain circumstances).

In a `switch` statement, the condition is evaluated only once and the result is compared to each `case` statement. In an `elseif` statement, the condition is evaluated again. If your condition is more complicated than a simple compare and/or is in a tight loop, a `switch` may be faster.

The statement list for a case can also be empty, which simply passes control into the statement list for the next case.

```
switch ($i) {
    case 0:
    case 1:
    case 2:
        print "i is less than 3 but not negative";
        break;
    case 3:
        print "i is 3";
}
```

A special case is the default case. This case matches anything that wasn't matched by the other cases, and should be the last `case` statement. For example:

```
switch ($i) {
    case 0:
        print "i equals 0";
        break;
    case 1:
        print "i equals 1";
        break;
    case 2:
        print "i equals 2";
        break;
    default:
        print "i is not equal to 0, 1 or 2";
}
```

The `case` expression may be any expression that evaluates to a simple type, that is, integer or floating-point numbers and strings. Arrays or objects cannot be used here unless they are dereferenced to a simple type.

The alternative syntax for control structures is supported with switches. For more information, see Alternative syntax for control structures .

```
switch ($i):
    case 0:
        print "i equals 0";
        break;
    case 1:
        print "i equals 1";
        break;
    case 2:
        print "i equals 2";
        break;
    default:
        print "i is not equal to 0, 1 or 2";
endswitch;
```

`declare`

The `declare` construct is used to set execution directives for a block of code. The syntax of `declare` is similar to the syntax of other flow control constructs:

```
declare (directive) statement
```

The `directive` section allows the behavior of the `declare` block to be set. Currently only one directive is recognized: the `ticks` directive. (See below for more information on the ticks directive)

The `statement` part of the `declare` block will be executed -- how it is executed and what side effects occur during execution may depend on the directive set in the `directive` block.

Ticks

A tick is an event that occurs for every *N* low-level statements executed by the parser within the `declare` block. The value for *N* is specified using `ticks=N` within the `declare` blocks's `directive` section.

The event(s) that occur on each tick are specified using the **register_tick_function()**. See the example below for more details. Note that more than one event can occur for each tick.

Example 12-1. Profile a section of PHP code

```
<?php
// A function that records the time when it is called
function profile ($dump = FALSE)
{
    static $profile;

    // Return the times stored in profile, then erase it
    if ($dump) {
        $temp = $profile;
        unset ($profile);
        return ($temp);
    }

    $profile[] = microtime ();
}

// Set up a tick handler
register_tick_function("profile");

// Initialize the function before the declare block
profile ();

// Run a block of code, throw a tick every 2nd statement
declare (ticks=2) {
    for ($x = 1; $x < 50; ++$x) {
        echo similar_text (md5($x), md5($x*$x)), "<br />;";
    }
}

// Display the data stored in the profiler
print_r (profile (TRUE));
?>
```

The example profiles the PHP code within the 'declare' block, recording the time at which every second low-level statement in the block was executed. This information can then be used to find the slow areas within particular segments of code. This process can be performed using other methods: using ticks is more convenient and easier to implement.

Ticks are well suited for debugging, implementing simple multitasking, backgrounded I/O and many other tasks.

See also **register_tick_function()** and **unregister_tick_function()**.

---

return

If called from within a function, the **return()** statement immediately ends execution of the current function, and returns its argument as the value of the function call. **return()** will also end the execution of an **eval()** statement or script file.

If called from the global scope, then execution of the current script file is ended. If the current script file was **include()**ed or **require()**ed, then control is passed back to the calling file. Furthermore, if the current script file was **include()**ed, then the value given to **return()** will be returned as the value of the **include()** call. If **return()** is called from within the main script file, then script execution ends. If the current script file was named by the auto_prepend_file or auto_append_file configuration options in the configuration file, then that script file's execution is ended.

For more information, see Returning values.

> **Note:** Note that since **return()** is a language construct and not a function, the parentheses surrounding its arguments are *not* required--in fact, it is more common to leave them out than to use them, although it doesn't matter one way or the other.

---

require()

The **require()** statement includes and evaluates the specific file.

require() includes and evaluates a specific file. Detailed information on how this inclusion works is described in the documentation for **include()**.

require() and **include()** are identical in every way except how they handle failure. **include()** produces a Warning while **require()** results in a Fatal Error. In other words, don't hesitate to use **require()** if you want a missing file to halt processing of the page.

include() does not behave this way, the script will continue regardless. Be sure to have an appropriate include_path setting as well.

Example 12-2. Basic require() examples

```
<?php

require 'prepend.php';

require $somefile;

require ('somefile.txt');

?>
```

See the **include()** documentation for more examples.

> **Note:** Prior to PHP 4.0.2, the following applies: **require()** will always attempt to read the target file, even if the line it's on never executes. The conditional statement won't affect **require()**. However, if the line on which the **require()** occurs is not executed, neither will any of the code in the target file be executed. Similarly, looping structures do not affect the behaviour of **require()**. Although the code contained in the target file is still subject to the loop, the **require()** itself happens only once.

Warning
Windows versions of PHP prior to PHP 4.3 do not support accessing remote files via this function, even if allow_url_fopen is enabled.

See also **include()**, **require_once()**, **include_once()**, **eval()**, **file()**, **readfile()**, **virtual()** and include_path.

---

include()

The **include()** statement includes and evaluates the specified file.

The documentation below also applies to **require()**. The two constructs are identical in every way except how they handle failure. **include()** produces a Warning while **require()** results in a Fatal Error. In other words, use **require()** if you want a missing file to halt processing of the page. **include()** does not behave this way, the script will continue regardless. Be sure to have an appropriate include_path setting as well.

When a file is included, the code it contains inherits the variable scope of the line on which the include occurs. Any variables available at that line in the calling file will be available within the called file, from that point forward.

Example 12-3. Basic include() example

```
vars.php
<?php

$color = 'green';
$fruit = 'apple';

?>

test.php
<?php

echo "A $color $fruit"; // A

include 'vars.php';

echo "A $color $fruit"; // A green apple

?>
```

If the include occurs inside a function within the calling file, then all of the code contained in the called file will behave as though it had been defined inside that function. So, it will follow the variable scope of that function.

Example 12-4. Including within functions

```
<?php

function foo()
{
global $color;
```

```
    include 'vars.php';

    echo "A $color $fruit";
}

/* vars.php is in the scope of foo() so     *
 * $fruit is NOT available outside of this  *
 * scope.  $color is because we declared it *
 * as global.                               */

foo();                    // A green apple
echo "A $color $fruit";   // A green

?>
```

When a file is included, parsing drops out of PHP mode and into HTML mode at the beginning of the target file, and resumes again at the end. For this reason, any code inside the target file which should be executed as PHP code must be enclosed within valid PHP start and end tags.

If "URL fopen wrappers" are enabled in PHP (which they are in the default configuration), you can specify the file to be included using an URL (via HTTP or other supported wrapper - see Appendix I for a list of protocols) instead of a local pathname. If the target server interprets the target file as PHP code, variables may be passed to the included file using an URL request string as used with HTTP GET. This is not strictly speaking the same thing as including the file and having it inherit the parent file's variable scope; the script is actually being run on the remote server and the result is then being included into the local script.

Warning
Windows versions of PHP prior to PHP 4.3 do not support accessing remote files via this function, even if allow_url_fopen is enabled.

Example 12-5. include() through HTTP

```
<?php

/* This example assumes that www.example.com is configured to parse .php *
 * files and not .txt files. Also, 'Works' here means that the variables *
 * $foo and $bar are available within the included file.                  */

// Won't work; file.txt wasn't handled by www.example.com as PHP
include 'http://www.example.com/file.txt?foo=1&bar=2';

// Won't work; looks for a file named 'file.php?foo=1&bar=2' on the
// local filesystem.
include 'file.php?foo=1&bar=2';

// Works.
include 'http://www.example.com/file.php?foo=1&bar=2';

$foo = 1;
$bar = 2;
include 'file.txt';  // Works.
include 'file.php';  // Works.

?>
```

See also Remote files, **fopen()** and **file()** for related information.

Because **include()** and **require()** are special language constructs, you must enclose them within a statement block if it's inside a conditional block.

Example 12-6. include() and conditional blocks

```
<?php

// This is WRONG and will not work as desired.
if ($condition)
    include $file;
else
    include $other;


// This is CORRECT.
if ($condition) {
    include $file;
} else {
    include $other;
}

?>
```

Handling Returns: It is possible to execute a **return()** statement inside an included file in order to terminate processing in that file and return to the script which called it. Also, it's possible to return values from included files. You can take the value of the include call as you would a normal function.

Note: In PHP 3, the return may not appear inside a block unless it's a function block, in which case the **return()** applies to that function and not the whole file.

Example 12-7. include() and the return() statement

```
return.php
<?php

$var = 'PHP';

return $var;

?>

noreturn.php
<?php

$var = 'PHP';

?>

testreturns.php
<?php

$foo = include 'return.php';

echo $foo; // prints 'PHP'

$bar = include 'noreturn.php';

echo $bar; // prints 1

?>
```

`$bar` is the value `1` because the include was successful. Notice the difference between the above examples. The first uses **return()** within the included file while the other does not. A few other ways to "include" files into variables are with **fopen()**, **file()** or by using **include()** along with Output Control Functions.

See also **require()**, **require_once()**, **include_once()**, **readfile()**, **virtual()**, and include_path.

---

require_once()

The **require_once()** statement includes and evaluates the specified file during the execution of the script. This is a behavior similar to the **require()** statement, with the only difference being that if the code from a file has already been included, it will not be included again. See the documentation for **require()** for more information on how this statement works.

require_once() should be used in cases where the same file might be included and evaluated more than once during a particular execution of a script, and you want to be sure that it is included exactly once to avoid problems with function redefinitions, variable value reassignments, etc.

For examples on using **require_once()** and **include_once()**, look at the PEAR code included in the latest PHP source code distributions.

Note: **require_once()** was added in PHP 4.0.1pl2

Note: Be aware, that the behaviour of **require_once()** and **include_once()** may not be what you expect on a non case sensitive operating system (such as Windows).

Example 12-8. require_once() is case sensitive

```
require_once("a.php"); // this will include a.php
require_once("A.php"); // this will include a.php again on Windows!
```

Warning
Windows versions of PHP prior to PHP 4.3 do not support accessing remote files via this function, even if allow_url_fopen is enabled.

See also: **require()**, **include()**, **include_once()**, **get_required_files()**, **get_included_files()**, **readfile()**, and **virtual()**.

---

include_once()

The **include_once()** statement includes and evaluates the specified file during the execution of the script. This is a behavior

similar to the **include()** statement, with the only difference being that if the code from a file has already been included, it will not be included again. As the name suggests, it will be included just once.

include_once() should be used in cases where the same file might be included and evaluated more than once during a particular execution of a script, and you want to be sure that it is included exactly once to avoid problems with function redefinitions, variable value reassignments, etc.

For more examples on using **require_once()** and **include_once()**, look at the PEAR code included in the latest PHP source code distributions.

> **Note:** **include_once()** was added in PHP 4.0.1pl2
>
> **Note:** Be aware, that the behaviour of **include_once()** and **require_once()** may not be what you expect on a non case sensitive operating system (such as Windows).
>
> **Example 12-9. include_once() is case sensitive**

```
include_once("a.php"); // this will include a.php
include_once("A.php"); // this will include a.php again on Windows!
```

Warning
Windows versions of PHP prior to PHP 4.3 do not support accessing remote files via this function, even if allow_url_fopen is enabled.

See also **include()**, **require()**, **require_once()**, **get_required_files()**, **get_included_files()**, **readfile()**, and **virtual()**.

Chapter 13. Functions

User-defined functions

A function may be defined using syntax such as the following:

```
function foo ($arg_1, $arg_2, ..., $arg_n)
{
    echo "Example function.\n";
    return $retval;
}
```

Any valid PHP code may appear inside a function, even other functions and class definitions.

In PHP 3, functions must be defined before they are referenced. No such requirement exists in PHP 4. *Except* when a function is conditionally defined such as shown in the two examples below.

When a function is defined in a conditional manner such as the two examples shown. Its definition must be processed *prior* to being called.

Example 13-1. Conditional functions

```
<?php

$makefoo = true;

/* We can't call foo() from here
   since it doesn't exist yet,
   but we can call bar() */

bar();

if ($makefoo) {
  function foo ()
  {
    echo "I don't exist until program execution reaches me.\n";
  }
}

/* Now we can safely call foo()
   since $makefoo evaluated to true */

if ($makefoo) foo();

function bar() {
{
  echo "I exist immediately upon program start.\n";
}
```

```
?>
```

Example 13-2. Functions within functions

```
<?php
function foo()
{
  function bar()
  {
    echo "I don't exist until foo() is called.\n";
  }
}

/* We can't call bar() yet
   since it doesn't exist. */

foo();

/* Now we can call bar(),
   foo()'s processesing has
   made it accessable. */

bar();

?>
```

PHP does not support function overloading, nor is it possible to undefine or redefine previously-declared functions.

PHP 3 does not support variable numbers of arguments to functions, although default arguments are supported (see Default argument values for more information). PHP 4 supports both: see Variable-length argument lists and the function references for **func_num_args()**, **func_get_arg()**, and **func_get_args()** for more information.

Function arguments

Information may be passed to functions via the argument list, which is a comma-delimited list of variables and/or constants.

PHP supports passing arguments by value (the default), passing by reference, and default argument values. Variable-length argument lists are supported only in PHP 4 and later; see Variable-length argument lists and the function references for **func_num_args()**, **func_get_arg()**, and **func_get_args()** for more information. A similar effect can be achieved in PHP 3 by passing an array of arguments to a function:

```
function takes_array($input)
{
    echo "$input[0] + $input[1] = ", $input[0]+$input[1];
}
```

Making arguments be passed by reference

By default, function arguments are passed by value (so that if you change the value of the argument within the function, it does not get changed outside of the function). If you wish to allow a function to modify its arguments, you must pass them by reference.

If you want an argument to a function to always be passed by reference, you can prepend an ampersand (&) to the argument name in the function definition:

```
function add_some_extra(&$string)
{
    $string .= 'and something extra.';
}
$str = 'This is a string, ';
add_some_extra($str);
echo $str;    // outputs 'This is a string, and something extra.'
```

Default argument values

A function may define C++-style default values for scalar arguments as follows:

```
function makecoffee ($type = "cappuccino")
{
    return "Making a cup of $type.\n";
}
echo makecoffee ();
```

```
echo makecoffee ("espresso");
```

The output from the above snippet is:

```
Making a cup of cappuccino.
Making a cup of espresso.
```

The default value must be a constant expression, not (for example) a variable or class member.

Note that when using default arguments, any defaults should be on the right side of any non-default arguments; otherwise, things will not work as expected. Consider the following code snippet:

```
function makeyogurt ($type = "acidophilus", $flavour)
{
    return "Making a bowl of $type $flavour.\n";
}

echo makeyogurt ("raspberry");   // won't work as expected
```

The output of the above example is:

```
Warning: Missing argument 2 in call to makeyogurt() in
/usr/local/etc/httpd/htdocs/php3test/functest.html on line 41
Making a bowl of raspberry .
```

Now, compare the above with this:

```
function makeyogurt ($flavour, $type = "acidophilus")
{
    return "Making a bowl of $type $flavour.\n";
}

echo makeyogurt ("raspberry");   // works as expected
```

The output of this example is:

```
Making a bowl of acidophilus raspberry.
```

Variable-length argument lists

PHP 4 has support for variable-length argument lists in user-defined functions. This is really quite easy, using the **func_num_args()**, **func_get_arg()**, and **func_get_args()** functions.

No special syntax is required, and argument lists may still be explicitly provided with function definitions and will behave as normal.

Returning values

Values are returned by using the optional return statement. Any type may be returned, including lists and objects. This causes the function to end its execution immediately and pass control back to the line from which it was called. See **return()** for more information.

```
function square ($num)
{
    return $num * $num;
}
echo square (4);   // outputs '16'.
```

You can't return multiple values from a function, but similar results can be obtained by returning a list.

```
function small_numbers()
{
    return array (0, 1, 2);
}
list ($zero, $one, $two) = small_numbers();
```

To return a reference from a function, you have to use the reference operator & in both the function declaration and when assigning the returned value to a variable:

```
function &returns_reference()
{
    return $someref;
}
```

```
$newref =& returns_reference();
```

For more information on references, please check out References Explained.

---

old_function

The old_function statement allows you to declare a function using a syntax identical to PHP/FI2 (except you must replace 'function' with 'old_function'.

This is a deprecated feature, and should only be used by the PHP/FI2->PHP 3 convertor.

Warning
Functions declared as old_function cannot be called from PHP's internal code. Among other things, this means you can't use them in functions such as usort(), array_walk(), and register_shutdown_function(). You can get around this limitation by writing a wrapper function (in normal PHP 3 form) to call the old_function.

Variable functions

PHP supports the concept of variable functions. This means that if a variable name has parentheses appended to it, PHP will look for a function with the same name as whatever the variable evaluates to, and will attempt to execute it. Among other things, this can be used to implement callbacks, function tables, and so forth.

Variable functions won't work with language constructs such as echo(), print(), unset(), isset(), empty(), include(), require() and the like. You need to use your own wrapper function to utilize any of these constructs as variable functions.

Example 13-3. Variable function example

```
<?php
function foo()
{
    echo "In foo()<br>\n";
}

function bar($arg = '')
{
    echo "In bar(); argument was '$arg'.<br>\n";
}

// This is a wrapper function around echo
function echoit($string)
{
    echo $string;
}

$func = 'foo';
$func();        // This calls foo()

$func = 'bar';
$func('test');  // This calls bar()

$func = 'echoit';
$func('test');  // This calls echoit()
?>
```

You can also call an object's method by using the variable functions feature.

Example 13-4. Variable method example

```
<?php
class Foo
{
    function Var()
    {
        $name = 'Bar';
        $this->$name(); // This calls the Bar() method
    }

    function Bar()
    {
        echo "This is Bar";
    }
}

$foo = new Foo();
$funcname = "Var";
$foo->$varname();   // This calls $foo->Var()
```

```
?>
```

See also **call_user_func()**, variable variables and **function_exists()**.

Chapter 14. Classes and Objects

class

A class is a collection of variables and functions working with these variables. A class is defined using the following syntax:

```
<?php
class Cart
{
    var $items;  // Items in our shopping cart

    // Add $num articles of $artnr to the cart

    function add_item ($artnr, $num)
    {
        $this->items[$artnr] += $num;
    }

    // Take $num articles of $artnr out of the cart

    function remove_item ($artnr, $num)
    {
        if ($this->items[$artnr] > $num) {
            $this->items[$artnr] -= $num;
            return true;
        } else {
            return false;
        }
    }
}
?>
```

This defines a class named Cart that consists of an associative array of articles in the cart and two functions to add and remove items from this cart.

Caution
The following cautionary notes are valid for PHP 4. The name `stdClass` is used interally by Zend and is reserved. You cannot have a class named `stdClass` in PHP. The function names `__sleep` and `__wakeup` are magical in PHP classes. You cannot have functions with these names in any of your classes unless you want the magic functionality associated with them. See below for more information. PHP reserves all function names starting with __ as magical. It is recommended that you do not use function names with __ in PHP unless you want some documented magic functionality.

Note: In PHP 4, only constant initializers for `var` variables are allowed. To initialize variables with non-constant values, you need an initialization function which is called automatically when an object is being constructed from the class. Such a function is called a constructor (see below).

```
<?php
/* None of these will work in PHP 4. */
class Cart
{
    var $todays_date = date("Y-m-d");
    var $name = $firstname;
    var $owner = 'Fred ' . 'Jones';
    var $items = array("VCR", "TV");
}

/* This is how it should be done. */
class Cart
{
    var $todays_date;
    var $name;
    var $owner;
    var $items;

    function Cart()
    {
        $this->todays_date = date("Y-m-d");
        $this->name = $GLOBALS['firstname'];
        /* etc. . . */
    }
}
?>
```

Classes are types, that is, they are blueprints for actual variables. You have to create a variable of the desired type with the `new` operator.

```
<?php
$cart = new Cart;
$cart->add_item("10", 1);

$another_cart = new Cart;
$another_cart->add_item("0815", 3);
```

This creates the objects $cart and $another_cart, both of the class Cart. The function add_item() of the $cart object is being called to add 1 item of article number 10 to the $cart. 3 items of article number 0815 are being added to $another_cart.

Both, $cart and $another_cart, have functions add_item(), remove_item() and a variable items. These are distinct functions and variables. You can think of the objects as something similar to directories in a filesystem. In a filesystem you can have two different files README.TXT, as long as they are in different directories. Just like with directories where you'll have to type the full pathname in order to reach each file from the toplevel directory, you have to specify the complete name of the function you want to call: In PHP terms, the toplevel directory would be the global namespace, and the pathname separator would be ->. Thus, the names $cart->items and $another_cart->items name two different variables. Note that the variable is named $cart->items, not $cart->$items, that is, a variable name in PHP has only a single dollar sign.

```
// correct, single $
$cart->items = array("10" => 1);

// invalid, because $cart->$items becomes $cart->""
$cart->$items = array("10" => 1);

// correct, but may or may not be what was intended:
// $cart->$myvar becomes $cart->items
$myvar = 'items';
$cart->$myvar = array("10" => 1);
```

Within a class definition, you do not know under which name the object will be accessible in your program: at the time the Cart class was written, it was unknown that the object will be named $cart or $another_cart later. Thus, you cannot write $cart->items within the Cart class itself. Instead, in order to be able to access it's own functions and variables from within a class, one can use the pseudo-variable $this which can be read as 'my own' or 'current object'. Thus, '$this->items[$artnr] += $num' can be read as 'add $num to the $artnr counter of my own items array' or 'add $num to the $artnr counter of the items array within the current object'.

> **Note:** There are some nice functions to handle classes and objects. You might want to take a look at the Class/Object Functions

---

extends

Often you need classes with similar variables and functions to another existing class. In fact, it is good practice to define a generic class which can be used in all your projects and adapt this class for the needs of each of your specific projects. To facilitate this, classes can be extensions of other classes. The extended or derived class has all variables and functions of the base class (this is called 'inheritance' despite the fact that nobody died) and what you add in the extended definition. It is not possible to substract from a class, that is, to undefine any existing functions or variables. An extended class is always dependent on a single base class, that is, multiple inheritance is not supported. Classes are extended using the keyword 'extends'.

```
class Named_Cart extends Cart
{
    var $owner;

    function set_owner ($name)
    {
        $this->owner = $name;
    }
}
```

This defines a class Named_Cart that has all variables and functions of Cart plus an additional variable $owner and an additional function set_owner(). You create a named cart the usual way and can now set and get the carts owner. You can still use normal cart functions on named carts:

```
$ncart = new Named_Cart;    // Create a named cart
$ncart->set_owner("kris");  // Name that cart
print $ncart->owner;        // print the cart owners name
$ncart->add_item("10", 1);  // (inherited functionality from cart)
```

This is also called a "parent-child" relationship. You create a class, parent, and use `extends` to create a new class *based* on the parent class: the child class. You can even use this new child class and create another class based on this child class.

> **Note:** Classes must be defined before they are used! If you want the class `Named_Cart` to extend the class `Cart`, you will have to define the class `Cart` first. If you want to create another class called `Yellow_named_cart` based on the class

`Named_Cart` you have to define `Named_Cart` first. To make it short: the order in which the classes are defined is important.

Constructors

Caution
In PHP 3 and PHP 4 constructors behave differently. The PHP 4 semantics are strongly preferred.

Constructors are functions in a class that are automatically called when you create a new instance of a class with `new`. In PHP 3, a function becomes a constructor when it has the same name as the class. In PHP 4, a function becomes a constructor, when it has the same name as the class it is defined in - the difference is subtle, but crucial (see below).

```
// Works in PHP 3 and PHP 4.
class Auto_Cart extends Cart
{
    function Auto_Cart()
    {
        $this->add_item ("10", 1);
    }
}
```

This defines a class Auto_Cart that is a Cart plus a constructor which initializes the cart with one item of article number "10" each time a new Auto_Cart is being made with "new". Constructors can take arguments and these arguments can be optional, which makes them much more useful. To be able to still use the class without parameters, all parameters to constructors should be made optional by providing default values.

```
// Works in PHP 3 and PHP 4.
class Constructor_Cart extends Cart
{
    function Constructor_Cart($item = "10", $num = 1)
    {
        $this->add_item ($item, $num);
    }
}

// Shop the same old boring stuff.

$default_cart = new Constructor_Cart;

// Shop for real...

$different_cart = new Constructor_Cart("20", 17);
```

You also can use the `@` operator to *mute* errors occuring in the constructor, e.g. `@new`.

Caution
In PHP 3, derived classes and constructors have a number of limitations. The following examples should be read carefully to understand these limitations.

```
class A
{
    function A()
    {
      echo "I am the constructor of A.<br>\n";
    }
}

class B extends A
{
    function C()
    {
        echo "I am a regular function.<br>\n";
    }
}

// no constructor is being called in PHP 3.
$b = new B;
```

In PHP 3, no constructor is being called in the above example. The rule in PHP 3 is: 'A constructor is a function of the same name as the class.'. The name of the class is B, and there is no function called B() in class B. Nothing happens.

This is fixed in PHP 4 by introducing another rule: If a class has no constructor, the constructor of the base class is being called, if it exists. The above example would have printed 'I am the constructor of A.
' in PHP 4.

```
class A
{
    function A()
    {
```

```
        echo "I am the constructor of A.<br>\n";
    }

    function B()
    {
        echo "I am a regular function named B in class A.<br>\n";
        echo "I am not a constructor in A.<br>\n";
    }
}

class B extends A
{
    function C()
    {
        echo "I am a regular function.<br>\n";
    }
}

// This will call B() as a constructor.
$b = new B;
```

In PHP 3, the function B() in class A will suddenly become a constructor in class B, although it was never intended to be. The rule in PHP 3 is: 'A constructor is a function of the same name as the class.'. PHP 3 does not care if the function is being defined in class B, or if it has been inherited.

This is fixed in PHP 4 by modifying the rule to: 'A constructor is a function of the same name as the class it is being defined in.'. Thus in PHP 4, the class B would have no constructor function of its own and the constructor of the base class would have been called, printing 'I am the constructor of A.
'.

Caution
Neither PHP 3 nor PHP 4 call constructors of the base class automatically from a constructor of a derived class. It is your responsibility to propagate the call to constructors upstream where appropriate.

> **Note:** There are no destructors in PHP 3 or PHP 4. You may use **register_shutdown_function()** instead to simulate most effects of destructors.

Destructors are functions that are called automatically when an object is destroyed, either with **unset()** or by simply going out of scope. There are no destructors in PHP.

::

Caution
The following is valid for PHP 4 only.

Sometimes it is useful to refer to functions and variables in base classes or to refer to functions in classes that have not yet any instances. The :: operator is being used for this.

```
class A
{
    function example()
    {
        echo "I am the original function A::example().<br>\n";
    }
}

class B extends A
{
    function example()
    {
        echo "I am the redefined function B::example().<br>\n";
        A::example();
    }
}

// there is no object of class A.
// this will print
//   I am the original function A::example().<br>
A::example();

// create an object of class B.
$b = new B;

// this will print
//   I am the redefined function B::example().<br>
//   I am the original function A::example().<br>
$b->example();
```

The above example calls the function example() in class A, but there is no object of class A, so that we cannot write $a->example() or similar. Instead we call example() as a 'class function', that is, as a function of the class itself, not any object of that class.

There are class functions, but there are no class variables. In fact, there is no object at all at the time of the call. Thus, a class function may not use any object variables (but it can use local and global variables), and it may no use $this at all.

In the above example, class B redefines the function example(). The original definition in class A is shadowed and no longer available, unless you are refering specifically to the implementation of example() in class A using the ::-operator. Write A::example() to do this (in fact, you should be writing parent::example(), as shown in the next section).

In this context, there is a current object and it may have object variables. Thus, when used from WITHIN an object function, you may use $this and object variables.

`parent`

You may find yourself writing code that refers to variables and functions in base classes. This is particularly true if your derived class is a refinement or specialisation of code in your base class.

Instead of using the literal name of the base class in your code, you should be using the special name `parent`, which refers to the name of your base class as given in the `extends` declaration of your class. By doing this, you avoid using the name of your base class in more than one place. Should your inheritance tree change during implementation, the change is easily made by simply changing the `extends` declaration of your class.

```
class A
{
    function example()
    {
        echo "I am A::example() and provide basic functionality.<br>\n";
    }
}

class B extends A
{
    function example()
    {
        echo "I am B::example() and provide additional functionality.<br>\n";
        parent::example();
    }
}

$b = new B;

// This will call B::example(), which will in turn call A::example().
$b->example();
```

Serializing objects - objects in sessions

> **Note:** In PHP 3, objects will lose their class association throughout the process of serialization and unserialization. The resulting variable is of type object, but has no class and no methods, thus it is pretty useless (it has become just like an array with a funny syntax).

Caution
The following information is valid for PHP 4 only.

serialize() returns a string containing a byte-stream representation of any value that can be stored in PHP. **unserialize()** can use this string to recreate the original variable values. Using serialize to save an object will save all variables in an object. The functions in an object will not be saved, only the name of the class.

In order to be able to **unserialize()** an object, the class of that object needs to be defined. That is, if you have an object $a of class A on page1.php and serialize this, you'll get a string that refers to class A and contains all values of variabled contained in $a. If you want to be able to unserialize this on page2.php, recreating $a of class A, the definition of class A must be present in page2.php. This can be done for example by storing the class defintion of class A in an include file and including this file in both page1.php and page2.php.

```
classa.inc:
  class A
  {
      var $one = 1;

      function show_one()
      {
          echo $this->one;
      }
  }

page1.php:
  include("classa.inc");
```

```
  $a = new A;
  $s = serialize($a);
  // store $s somewhere where page2.php can find it.
  $fp = fopen("store", "w");
  fputs($fp, $s);
  fclose($fp);

page2.php:
  // this is needed for the unserialize to work properly.
  include("classa.inc");

  $s = implode("", @file("store"));
  $a = unserialize($s);

  // now use the function show_one() of the $a object.
  $a->show_one();
```

If you are using sessions and use **session_register()** to register objects, these objects are serialized automatically at the end of each PHP page, and are unserialized automatically on each of the following pages. This basically means that these objects can show up on any of your pages once they become part of your session.

It is strongly recommended that you include the class definitions of all such registered objects on all of your pages, even if you do not actually use these classes on all of your pages. If you don't and an object is being unserialized without its class definition being present, it will lose its class association and become an object of class `stdClass` without any functions available at all, that is, it will become quite useless.

So if in the example above $a became part of a session by running `session_register("a")`, you should include the file `classa.inc` on all of your pages, not only page1.php and page2.php.

The magic functions `__sleep` and `__wakeup`

serialize() checks if your class has a function with the magic name `__sleep`. If so, that function is being run prior to any serialization. It can clean up the object and is supposed to return an array with the names of all variables of that object that should be serialized.

The intended use of `__sleep` is to close any database connections that object may have, committing pending data or perform similar cleanup tasks. Also, the function is useful if you have very large objects which need not be saved completely.

Conversely, **unserialize()** checks for the presence of a function with the magic name `__wakeup`. If present, this function can reconstruct any resources that object may have.

The intended use of `__wakeup` is to reestablish any database connections that may have been lost during serialization and perform other reinitialization tasks.

References inside the constructor

Creating references within the constructor can lead to confusing results. This tutorial-like section helps you to avoid problems.

```
class Foo
{
    function Foo($name)
    {
        // create a reference inside the global array $globalref
        global $globalref;
        $globalref[] = &$this;
        // set name to passed value
        $this->setName($name);
        // and put it out
        $this->echoName();
    }

    function echoName()
    {
        echo "<br>",$this->name;
    }

    function setName($name)
    {
        $this->name = $name;
    }
}
```

Let us check out if there is a difference between `$bar1` which has been created using the copy `=` operator and `$bar2` which has been created using the reference `=&` operator...

```
$bar1 = new Foo('set in constructor');
$bar1->echoName();
$globalref[0]->echoName();

/* output:
set in constructor
set in constructor
set in constructor */

$bar2 =& new Foo('set in constructor');
$bar2->echoName();
$globalref[1]->echoName();

/* output:
set in constructor
set in constructor
set in constructor */
```

Apparently there is no difference, but in fact there is a very significant one: `$bar1` and `$globalref[0]` are _NOT_ referenced, they are NOT the same variable. This is because "new" does not return a reference by default, instead it returns a copy.

> **Note:** There is no performance loss (since PHP 4 and up use reference counting) returning copies instead of references. On the contrary it is most often better to simply work with copies instead of references, because creating references takes some time where creating copies virtually takes no time (unless none of them is a large array or object and one of them gets changed and the other(s) one(s) subsequently, then it would be wise to use references to change them all concurrently).

To prove what is written above let us watch the code below.

```
// now we will change the name. what do you expect?
// you could expect that both $bar1 and $globalref[0] change their names...
$bar1->setName('set from outside');

// as mentioned before this is not the case.
$bar1->echoName();
$globalref[0]->echoName();

/* output:
set from outside
set in constructor */

// let us see what is different with $bar2 and $globalref[1]
$bar2->setName('set from outside');

// luckily they are not only equal, they are the same variable
// thus $bar2->name and $globalref[1]->name are the same too
$bar2->echoName();
$globalref[1]->echoName();

/* output:
set from outside
set from outside */
```

Another final example, try to understand it.

```
class A
{
    function A($i)
    {
        $this->value = $i;
        // try to figure out why we do not need a reference here
        $this->b = new B($this);
    }

    function createRef()
    {
        $this->c = new B($this);
    }

    function echoValue()
    {
        echo "<br>","class ",get_class($this),': ',$this->value;
    }
}


class B
{
    function B(&$a)
    {
        $this->a = &$a;
    }

    function echoValue()
    {
        echo "<br>","class ",get_class($this),': ',$this->a->value;
    }
}

// try to undestand why using a simple copy here would yield
```

```
// in an undesired result in the *-marked line
$a =& new A(10);
$a->createRef();

$a->echoValue();
$a->b->echoValue();
$a->c->echoValue();

$a->value = 11;

$a->echoValue();
$a->b->echoValue(); // *
$a->c->echoValue();

/*
output:
class A: 10
class B: 10
class B: 10
class A: 11
class B: 11
class B: 11
*/
```

Chapter 15. References Explained

What References Are

References in PHP are a means to access the same variable content by different names. They are not like C pointers, they are symbol table aliases. Note that in PHP, variable name and variable content are different, so the same content can have different names. The most close analogy is with Unix filenames and files - variable names are directory entries, while variable contents is the file itself. References can be thought of as hardlinking in Unix filesystem.

What References Do

PHP references allow you to make two variables to refer to the same content. Meaning, when you do:

```
$a =& $b
```

it means that `$a` and `$b` point to the same variable.

> **Note:** `$a` and `$b` are completely equal here, that's not `$a` is pointing to `$b` or vice versa, that's `$a` and `$b` pointing to the same place.

The same syntax can be used with functions, that return references, and with `new` operator (in PHP 4.0.4 and later):

```
$bar =& new fooclass();
$foo =& find_var ($bar);
```

> **Note:** Not using the `&` operator causes a copy of the object to be made. If you use `$this` in the class it will operate on the current instance of the class. The assignment without `&` will copy the instance (i.e. the object) and `$this` will operate on the copy, which is not always what is desired. Usually you want to have a single instance to work with, due to performance and memory consumption issues.
>
> While you can use the `@` operator to *mute* any errors in the constructor when using it as `@new`, this does not work when using the `&new` statement. This is a limitation of the Zend Engine and will therefore result in a parser error.

The second thing references do is to pass variables by-reference. This is done by making a local variable in a function and a variable in the calling scope reference to the same content. Example:

```
function foo (&$var)
{
    $var++;
}

$a=5;
foo ($a);
```

will make `$a` to be 6. This happens because in the function `foo` the variable `$var` refers to the same content as `$a`. See also more detailed explanations about passing by reference.

The third thing reference can do is return by reference.

What References Are Not

As said before, references aren't pointers. That means, the following construct won't do what you expect:

```
function foo (&$var)
{
    $var =& $GLOBALS["baz"];
}
foo($bar);
```

What happens is that `$var` in `foo` will be bound with `$bar` in caller, but then it will be re-bound with `$GLOBALS["baz"]`. There's no way to bind `$bar` in the calling scope to something else using the reference mechanism, since `$bar` is not available in the function `foo` (it is represented by `$var`, but `$var` has only variable contents and not name-to-value binding in the calling symbol table).

Passing by Reference

You can pass variable to function by reference, so that function could modify its arguments. The syntax is as follows:

```
function foo (&$var)
{
    $var++;
}

$a=5;
foo ($a);
// $a is 6 here
```

Note that there's no reference sign on function call - only on function definition. Function definition alone is enough to correctly pass the argument by reference.

Following things can be passed by reference:

- Variable, i.e. `foo($a)`
- New statement, i.e. `foo(new foobar())`
- Reference, returned from a function, i.e.:

  ```
  function &bar()
  {
      $a = 5;
      return $a;
  }
  foo(bar());
  ```

  See also explanations about returning by reference.

Any other expression should not be passed by reference, as the result is undefined. For example, the following examples of passing by reference are invalid:

```
function bar() // Note the missing &
{
    $a = 5;
    return $a;
}
foo(bar());

foo($a = 5) // Expression, not variable
foo(5) // Constant, not variable
```

These requirements are for PHP 4.0.4 and later.

Returning References

Returning by-reference is useful when you want to use a function to find which variable a reference should be bound to. When returning references, use this syntax:

```
function &find_var ($param)
```

```
{
    ...code...
    return $found_var;
}

$foo =& find_var ($bar);
$foo->x = 2;
```

In this example, the property of the object returned by the `find_var` function would be set, not the copy, as it would be without using reference syntax.

> **Note:** Unlike parameter passing, here you have to use `&` in both places - to indicate that you return by-reference, not a copy as usual, and to indicate that reference binding, rather than usual assignment, should be done for `$foo`.

Unsetting References

When you unset the reference, you just break the binding between variable name and variable content. This does not mean that variable content will be destroyed. For example:

```
$a = 1;
$b =& $a;
unset ($a);
```

won't unset `$b`, just `$a`.

Again, it might be useful to think about this as analogous to Unix **unlink** call.

Spotting References

Many syntax constructs in PHP are implemented via referencing mechanisms, so everything told above about reference binding also apply to these constructs. Some constructs, like passing and returning by-reference, are mentioned above. Other constructs that use references are:

`global` References

When you declare variable as **global $var** you are in fact creating reference to a global variable. That means, this is the same as:

```
$var =& $GLOBALS["var"];
```

That means, for example, that unsetting `$var` won't unset global variable.

`$this`

In an object method, `$this` is always reference to the caller object.

III. Features

Table of Contents

Chapter 16. HTTP authentication with PHP

The HTTP Authentication hooks in PHP are only available when it is running as an Apache module and is hence not available in the CGI version. In an Apache module PHP script, it is possible to use the **header()** function to send an "Authentication Required" message to the client browser causing it to pop up a Username/Password input window. Once the user has filled in a username and a password, the URL containing the PHP script will be called again with the predefined variables PHP_AUTH_USER, PHP_AUTH_PW, and PHP_AUTH_TYPE set to the user name, password and authentication type respectively. These predefined variables are found in the $_SERVER and $HTTP_SERVER_VARS arrays. Only "Basic" authentication is supported. See the **header()** function for more information.

> **PHP Version Note:** Autoglobals, such as $_SERVER, became available in PHP version 4.1.0. $HTTP_SERVER_VARS has been available since PHP 3.

An example script fragment which would force client authentication on a page is as follows:

Example 16-1. HTTP Authentication example

```
<?php
  if (!isset($_SERVER['PHP_AUTH_USER'])) {
    header('WWW-Authenticate: Basic realm="My Realm"');
    header('HTTP/1.0 401 Unauthorized');
    echo 'Text to send if user hits Cancel button';
    exit;
  } else {
    echo "<p>Hello {$_SERVER['PHP_AUTH_USER']}.</p>";
    echo "<p>You entered {$_SERVER['PHP_AUTH_PW']} as your password.</p>";
  }
?>
```

> **Compatibility Note:** Please be careful when coding the HTTP header lines. In order to guarantee maximum compatibility with all clients, the keyword "Basic" should be written with an uppercase "B", the realm string must be enclosed in double (not single) quotes, and exactly one space should precede the *401* code in the *HTTP/1.0 401* header line.

Instead of simply printing out PHP_AUTH_USER and PHP_AUTH_PW, as done in the above example, you may want to check the username and password for validity. Perhaps by sending a query to a database, or by looking up the user in a dbm file.

Watch out for buggy Internet Explorer browsers out there. They seem very picky about the order of the headers. Sending the *WWW-Authenticate* header before the `HTTP/1.0 401` header seems to do the trick for now.

As of PHP 4.3.0, in order to prevent someone from writing a script which reveals the password for a page that was authenticated through a traditional external mechanism, the PHP_AUTH variables will not be set if external authentication is enabled for that particular page and safe mode is enabled. Regardless, REMOTE_USER can be used to identify the externally-authenticated user. So, you can use `$_SERVER['REMOTE_USER']`.

> **Configuration Note:** PHP uses the presence of an `AuthType` directive to determine whether external authentication is in effect.

Note, however, that the above does not prevent someone who controls a non-authenticated URL from stealing passwords from authenticated URLs on the same server.

Both Netscape Navigator and Internet Explorer will clear the local browser window's authentication cache for the realm upon receiving a server response of 401. This can effectively "log out" a user, forcing them to re-enter their username and password. Some people use this to "time out" logins, or provide a "log-out" button.

Example 16-2. HTTP Authentication example forcing a new name/password

```
<?php
  function authenticate() {
    header('WWW-Authenticate: Basic realm="Test Authentication System"');
    header('HTTP/1.0 401 Unauthorized');
    echo "You must enter a valid login ID and password to access this resource\n";
    exit;
  }

  if (!isset($_SERVER['PHP_AUTH_USER']) ||
      ($_POST['SeenBefore'] == 1 && $_POST['OldAuth'] == $_SERVER['PHP_AUTH_USER'])) {
   authenticate();
  }
  else {
   echo "<p>Welcome: {$_SERVER['PHP_AUTH_USER']}<br>";
   echo "Old: {$_REQUEST['OldAuth']}";
   echo "<form action='{$_SERVER['PHP_SELF']}' METHOD='POST'>\n";
   echo "<input type='hidden' name='SeenBefore' value='1'>\n";
   echo "<input type='hidden' name='OldAuth' value='{$_SERVER['PHP_AUTH_USER']}'>\n";
   echo "<input type='submit' value='Re Authenticate'>\n";
   echo "</form></p>\n";
  }
?>
```

This behavior is not required by the HTTP Basic authentication standard, so you should never depend on this. Testing with Lynx has shown that Lynx does not clear the authentication credentials with a 401 server response, so pressing back and then forward again will open the resource as long as the credential requirements haven't changed. The user can press the '_' key to clear their authentication information, however.

Also note that this does not work using Microsoft's IIS server and the CGI version of PHP due to a limitation of IIS.

> **Note:** If safe mode is enabled, the uid of the script is added to the `realm` part of the `WWW-Authenticate` header.

Chapter 17. Cookies

PHP transparently supports HTTP cookies. Cookies are a mechanism for storing data in the remote browser and thus tracking or identifying return users. You can set cookies using the **setcookie()** function. Cookies are part of the HTTP header, so **setcookie()** must be called before any output is sent to the browser. This is the same limitation that **header()** has. You can use the output buffering functions to delay the script output until you have decided whether or not to set any cookies or send any headers.

Any cookies sent to you from the client will automatically be turned into a PHP variable just like GET and POST method data, depending on the `register_globals` and `variables_order` configuration variables. If you wish to assign multiple values to a single cookie, just add *[]* to the cookie name.

In PHP 4.1.0 and later, the `$_COOKIE` auto-global array will always be set with any cookies sent from the client. `$HTTP_COOKIE_VARS` is also set in earlier versions of PHP when the `track_vars` configuration variable is set.

For more details, including notes on browser bugs, see the **setcookie()** function.

Chapter 18. Handling file uploads

POST method uploads

PHP is capable of receiving file uploads from any RFC-1867 compliant browser (which includes Netscape Navigator 3 or later, Microsoft Internet Explorer 3 with a patch from Microsoft, or later without a patch). This feature lets people upload both text and binary files. With PHP's authentication and file manipulation functions, you have full control over who is allowed to upload and what is to be done with the file once it has been uploaded.

> **Related Configurations Note:** See also the file_uploads, upload_max_filesize, upload_tmp_dir, and post_max_size directives in `php.ini`

Note that PHP also supports PUT-method file uploads as used by Netscape Composer and W3C's Amaya clients. See the PUT Method Support for more details.

A file upload screen can be built by creating a special form which looks something like this:

Example 18-1. File Upload Form

```
<form enctype="multipart/form-data" action="_URL_" method="post">
<input type="hidden" name="MAX_FILE_SIZE" value="30000">
Send this file: <input name="userfile" type="file">
<input type="submit" value="Send File">
</form>
```

The _URL_ should point to a PHP file. The MAX_FILE_SIZE hidden field must precede the file input field and its value is the maximum filesize accepted. The value is in bytes.

Warning
The MAX_FILE_SIZE is advisory to the browser. It is easy to circumvent this maximum. So don't count on it that the browser obeys your wish! The PHP-settings for maximum-size, however, cannot be fooled.

The Variables defined for uploaded files differs depending on the PHP version and configuration. The autoglobal $_FILES exists as of PHP 4.1.0 The `$HTTP_POST_FILES` array has existed since PHP 4.0.0. These arrays will contain all your uploaded file information. Using `$_FILES` is preferred. If the PHP directive register_globals is *on*, related variable names will also exist. register_globals defaults to *off* as of PHP 4.2.0.

The contents of $_FILES from our example script is as follows. Note that this assumes the use of the file upload name *userfile*, as used in the example script above.

```
$_FILES['userfile']['name']
```

The original name of the file on the client machine.

`$_FILES['userfile']['type']`

The mime type of the file, if the browser provided this information. An example would be `"image/gif"`.

`$_FILES['userfile']['size']`

The size, in bytes, of the uploaded file.

`$_FILES['userfile']['tmp_name']`

The temporary filename of the file in which the uploaded file was stored on the server.

`$_FILES['userfile']['error']`

The error code associated with this file upload. *['error']* was added in PHP 4.2.0

Note: In PHP versions prior 4.1.0 this was named `$HTTP_POST_FILES` and it's not an autoglobal variable like `$_FILES` is. PHP 3 does not support `$HTTP_POST_FILES`.

When register_globals is turned *on* in `php.ini`, additional variables are available. For example, `$userfile_name` will equal `$_FILES['userfile']['name']`, `$userfile_type` will equal `$_FILES['userfile']['type']`, etc. Keep in mind that as of PHP 4.2.0, register_globals defaults to off. It's preferred to not rely on this directive.

Files will by default be stored in the server's default temporary directory, unless another location has been given with the upload_tmp_dir directive in `php.ini`. The server's default directory can be changed by setting the environment variable `TMPDIR` in the environment in which PHP runs. Setting it using **putenv()** from within a PHP script will not work. This environment variable can also be used to make sure that other operations are working on uploaded files, as well.

Example 18-2. Validating file uploads

See also the function entries for **is_uploaded_file()** and **move_uploaded_file()** for further information. The following example will process the file upload that came from a form.

```
<?php
// In PHP earlier then 4.1.0, $HTTP_POST_FILES should be used instead of $_FILES.
// In PHP earlier then 4.0.3, use copy() and is_uploaded_file() instead of move_uploaded_file

$uploaddir = '/var/www/uploads/';

print "<pre>";
if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploaddir . $_FILES['userfile']['name'])) {
    print "File is valid, and was successfully uploaded.  Here's some more debugging info:\n";
    print_r($_FILES);
} else {
    print "Possible file upload attack!  Here's some debugging info:\n";
    print_r($_FILES);
}

?>
```

The PHP script which receives the uploaded file should implement whatever logic is necessary for determining what should be done with the uploaded file. You can for example use the `$_FILES['userfile']['size']` variable to throw away any files that are either too small or too big. You could use the `$_FILES['userfile']['type']` variable to throw away any files that didn't match a certain type criteria. As of PHP 4.2.0, you could use `$_FILES['userfile']['error']` and plan your logic according to the error codes. Whatever the logic, you should either delete the file from the temporary directory or move it elsewhere.

The file will be deleted from the temporary directory at the end of the request if it has not been moved away or renamed.

Error Messages Explained

Since PHP 4.2.0, PHP returns an appropriate error code along with the file array. The error code can be found in the *['error']* segment of the file array that is created during the file upload by PHP. In otherwords, the error might be found in `$_FILES['userfile']['error']`.

`UPLOAD_ERR_OK`

Value: 0; There is no error, the file uploaded with success.

`UPLOAD_ERR_INI_SIZE`

Value: 1; The uploaded file exceeds the upload_max_filesize directive in `php.ini`.

`UPLOAD_ERR_FORM_SIZE`

Value: 2; The uploaded file exceeds the *MAX_FILE_SIZE* directive that was specified in the html form.

`UPLOAD_ERR_PARTIAL`

Value: 3; The uploaded file was only partially uploaded.

`UPLOAD_ERR_NO_FILE`

Value: 4; No file was uploaded.

Note: These became PHP constants in PHP 4.3.0

Common Pitfalls

The `MAX_FILE_SIZE` item cannot specify a file size greater than the file size that has been set in the upload_max_filesize ini-setting. The default is 2 Megabytes.

If memory limit is enabled, larger memory_limit may be needed. Make sure to set memory_limit large enough.

If max_execution_time is set too small, script execution may be exceeded the value. Make sure to set `max_execution_time` large enough.

If post_max_size set too small, large files cannot be uploaded. Make sure to set `post_max_size` large enough.

Not validating which file you operate on may mean that users can access sensitive information in other directories.

Please note that the CERN httpd seems to strip off everything starting at the first whitespace in the content-type mime header it gets from the client. As long as this is the case, CERN httpd will not support the file upload feature.

Due to the large amount of directory listing styles we cannot guarantee that files with exotic names (like containing spaces) are handled properly.

Uploading multiple files

Multiple files can be uploaded using different `name` for `input`.

It is also possible to upload multiple files simultaneously and have the information organized automatically in arrays for you. To do so, you need to use the same array submission syntax in the HTML form as you do with multiple selects and checkboxes:

Note: Support for multiple file uploads was added in version 3.0.10.

Example 18-3. Uploading multiple files

```
<form action="file-upload.php" method="post" enctype="multipart/form-data">
  Send these files:<br>
  <input name="userfile[]" type="file"><br>
  <input name="userfile[]" type="file"><br>
  <input type="submit" value="Send files">
</form>
```

When the above form is submitted, the arrays `$_FILES['userfile']`, `$_FILES['userfile']['name']`, and `$_FILES['userfile']['size']` will be initialized (as well as in $HTTP_POST_FILES for PHP version prior 4.1.0). When `register_globals` is on, globals for uploaded files are also initialized. Each of these will be a numerically indexed array of the appropriate values for the submitted files.

For instance, assume that the filenames `/home/test/review.html` and `/home/test/xwp.out` are submitted. In this case, `$_FILES['userfile']['name'][0]` would contain the value `review.html`, and `$_FILES['userfile']['name'][1]` would contain the value `xwp.out`. Similarly, `$_FILES['userfile']['size'][0]` would contain `review.html`'s filesize, and so forth.

`$_FILES['userfile']['name'][0]`, `$_FILES['userfile']['tmp_name'][0]`, `$_FILES['userfile']['size'][0]`, and `$_FILES['userfile']['type'][0]` are also set.

PUT method support

PUT method support has changed between PHP 3 and PHP 4. In PHP 4, one should use the standard input stream to read the

contents of an HTTP PUT.

Example 18-4. Saving HTTP PUT files with PHP 4

```
<?php
/* PUT data comes in on the stdin stream */
$putdata = fopen("php://stdin","r");

/* Open a file for writting */
$fp = fopen("myputfile.ext","w");

/* Read the data 1kb at a time
   and write to the file */
while ($data = fread($putdata,1024))
  fwrite($fp,$data);

/* Close the streams */
fclose($fp);
fclose($putdata);
?>
```

> **Note:** All documentation below applies to PHP 3 only.

PHP provides support for the HTTP PUT method used by clients such as Netscape Composer and W3C Amaya. PUT requests are much simpler than a file upload and they look something like this:

```
PUT /path/filename.html HTTP/1.1
```

This would normally mean that the remote client would like to save the content that follows as: /path/filename.html in your web tree. It is obviously not a good idea for Apache or PHP to automatically let everybody overwrite any files in your web tree. So, to handle such a request you have to first tell your web server that you want a certain PHP script to handle the request. In Apache you do this with the *Script* directive. It can be placed almost anywhere in your Apache configuration file. A common place is inside a <Directory> block or perhaps inside a <Virtualhost> block. A line like this would do the trick:

```
Script PUT /put.php
```

This tells Apache to send all PUT requests for URIs that match the context in which you put this line to the put.php script. This assumes, of course, that you have PHP enabled for the .php extension and PHP is active.

Inside your put.php file you would then do something like this:

```
<?php copy($PHP_UPLOADED_FILE_NAME,$DOCUMENT_ROOT.$REQUEST_URI); ?>
```

This would copy the file to the location requested by the remote client. You would probably want to perform some checks and/or authenticate the user before performing this file copy. The only trick here is that when PHP sees a PUT-method request it stores the uploaded file in a temporary file just like those handled but the POST-method. When the request ends, this temporary file is deleted. So, your PUT handling PHP script has to copy that file somewhere. The filename of this temporary file is in the $PHP_PUT_FILENAME variable, and you can see the suggested destination filename in the $REQUEST_URI (may vary on non-Apache web servers). This destination filename is the one that the remote client specified. You do not have to listen to this client. You could, for example, copy all uploaded files to a special uploads directory.

Chapter 19. Using remote files

As long as *allow_url_fopen* is enabled in php.ini, you can use HTTP and FTP URLs with most of the functions that take a filename as a parameter. In addition, URLs can be used with the **include()**, **include_once()**, **require()** and **require_once()** statements. See Appendix I for more information about the protocols supported by PHP.

> **Note:** In PHP 4.0.3 and older, in order to use URL wrappers, you were required to configure PHP using the configure option `--enable-url-fopen-wrapper`.

> **Note:** The Windows versions of PHP earlier than PHP 4.3 did not support remote file accessing for the following functions: **include()**, **include_once()**, **require()**, **require_once()**, and the imagecreatefromXXX functions in the Reference XLI, *Image functions* extension.

For example, you can use this to open a file on a remote web server, parse the output for the data you want, and then use that data in a database query, or simply to output it in a style matching the rest of your website.

Example 19-1. Getting the title of a remote page

```
<?php
```

```
$file = fopen ("http://www.example.com/", "r");
if (!$file) {
    echo "<p>Unable to open remote file.\n";
    exit;
}
while (!feof ($file)) {
    $line = fgets ($file, 1024);
    /* This only works if the title and its tags are on one line */
    if (eregi ("<title>(.*)</title>", $line, $out)) {
        $title = $out[1];
        break;
    }
}
fclose($file);
?>
```

You can also write to files on an FTP server (provided that you have connected as a user with the correct access rights). You can only create new files using this method; if you try to overwrite a file that already exists, the **fopen()** call will fail.

To connect as a user other than 'anonymous', you need to specify the username (and possibly password) within the URL, such as 'ftp://user:password@ftp.example.com/path/to/file'. (You can use the same sort of syntax to access files via HTTP when they require Basic authentication.)

Example 19-2. Storing data on a remote server

```
<?php
$file = fopen ("ftp://ftp.example.com/incoming/outputfile", "w");
if (!$file) {
    echo "<p>Unable to open remote file for writing.\n";
    exit;
}
/* Write the data here. */
fputs ($file, $_SERVER['HTTP_USER_AGENT'] . "\n");
fclose ($file);
?>
```

> **Note:** You might get the idea from the example above that you can use this technique to write to a remote log file. Unfortunately that would not work because the **fopen()** call will fail if the remote file already exists. To do distributed logging like that, you should take a look at **syslog()**.

Chapter 20. Connection handling

> **Note:** The following applies to 3.0.7 and later.

Internally in PHP a connection status is maintained. There are 3 possible states:

- 0 - NORMAL
- 1 - ABORTED
- 2 - TIMEOUT

When a PHP script is running normally the NORMAL state, is active. If the remote client disconnects the ABORTED state flag is turned on. A remote client disconnect is usually caused by the user hitting his STOP button. If the PHP-imposed time limit (see **set_time_limit()**) is hit, the TIMEOUT state flag is turned on.

You can decide whether or not you want a client disconnect to cause your script to be aborted. Sometimes it is handy to always have your scripts run to completion even if there is no remote browser receiving the output. The default behaviour is however for your script to be aborted when the remote client disconnects. This behaviour can be set via the ignore_user_abort `php.ini` directive as well as through the corresponding "php_value ignore_user_abort" Apache .conf directive or with the **ignore_user_abort()** function. If you do not tell PHP to ignore a user abort and the user aborts, your script will terminate. The one exception is if you have registered a shutdown function using **register_shutdown_function()**. With a shutdown function, when the remote user hits his STOP button, the next time your script tries to output something PHP will detect that the connection has been aborted and the shutdown function is called. This shutdown function will also get called at the end of your script terminating normally, so to do something different in case of a client disconnect you can use the **connection_aborted()** function. This function will return **TRUE** if the connection was aborted.

Your script can also be terminated by the built-in script timer. The default timeout is 30 seconds. It can be changed using the max_execution_time `php.ini` directive or the corresponding "php_value max_execution_time" Apache .conf directive as well as with the **set_time_limit()** function. When the timer expires the script will be aborted and as with the above client disconnect case, if a shutdown function has been registered it will be called. Within this shutdown function you can check to see if a timeout caused the shutdown function to be called by calling the **connection_timeout()** function. This function will return **TRUE** if a timeout caused the shutdown function to be called.

One thing to note is that both the ABORTED and the TIMEOUT states can be active at the same time. This is possible if you tell PHP to ignore user aborts. PHP will still note the fact that a user may have broken the connection, but the script will keep running. If it then hits the time limit it will be aborted and your shutdown function, if any, will be called. At this point you will find that **connection_timeout()** and **connection_aborted()** return **TRUE**. You can also check both states in a single call by using the **connection_status()**. This function returns a bitfield of the active states. So, if both states are active it would return 3, for example.

Chapter 21. Persistent Database Connections

Persistent connections are SQL links that do not close when the execution of your script ends. When a persistent connection is requested, PHP checks if there's already an identical persistent connection (that remained open from earlier) - and if it exists, it uses it. If it does not exist, it creates the link. An 'identical' connection is a connection that was opened to the same host, with the same username and the same password (where applicable).

> **Note:** There are other extensions that provide persistent connections, such as the IMAP extension.

People who aren't thoroughly familiar with the way web servers work and distribute the load may mistake persistent connects for what they're not. In particular, they do *not* give you an ability to open 'user sessions' on the same SQL link, they do *not* give you an ability to build up a transaction efficiently, and they don't do a whole lot of other things. In fact, to be extremely clear about the subject, persistent connections don't give you *any* functionality that wasn't possible with their non-persistent brothers.

Why?

This has to do with the way web servers work. There are three ways in which your web server can utilize PHP to generate web pages.

The first method is to use PHP as a CGI "wrapper". When run this way, an instance of the PHP interpreter is created and destroyed for every page request (for a PHP page) to your web server. Because it is destroyed after every request, any resources that it acquires (such as a link to an SQL database server) are closed when it is destroyed. In this case, you do not gain anything from trying to use persistent connections -- they simply don't persist.

The second, and most popular, method is to run PHP as a module in a multiprocess web server, which currently only includes Apache. A multiprocess server typically has one process (the parent) which coordinates a set of processes (its children) who actually do the work of serving up web pages. When each request comes in from a client, it is handed off to one of the children that is not already serving another client. This means that when the same client makes a second request to the server, it may be serviced by a different child process than the first time. What a persistent connection does for you in this case it make it so each child process only needs to connect to your SQL server the first time that it serves a page that makes use of such a connection. When another page then requires a connection to the SQL server, it can reuse the connection that child established earlier.

The last method is to use PHP as a plug-in for a multithreaded web server. Currently PHP 4 has support for ISAPI, WSAPI, and NSAPI (on Windows), which all allow PHP to be used as a plug-in on multithreaded servers like Netscape FastTrack (iPlanet), Microsoft's Internet Information Server (IIS), and O'Reilly's WebSite Pro. The behavior is essentially the same as for the multiprocess model described before. Note that SAPI support is not available in PHP 3.

If persistent connections don't have any added functionality, what are they good for?

The answer here is extremely simple -- efficiency. Persistent connections are good if the overhead to create a link to your SQL server is high. Whether or not this overhead is really high depends on many factors. Like, what kind of database it is, whether or not it sits on the same computer on which your web server sits, how loaded the machine the SQL server sits on is and so forth. The bottom line is that if that connection overhead is high, persistent connections help you considerably. They cause the child process to simply connect only once for its entire lifespan, instead of every time it processes a page that requires connecting to the SQL server. This means that for every child that opened a persistent connection will have its own open persistent connection to the server. For example, if you had 20 different child processes that ran a script that made a persistent connection to your SQL server, you'd have 20 different connections to the SQL server, one from each child.

Note, however, that this can have some drawbacks if you are using a database with connection limits that are exceeded by persistent child connections. If your database has a limit of 16 simultaneous connections, and in the course of a busy server session, 17 child threads attempt to connect, one will not be able to. If there are bugs in your scripts which do not allow the connections to shut down (such as infinite loops), the database with only 16 connections may be rapidly swamped. Check your database documentation for information on handling abandoned or idle connections.

Warning
There are a couple of additional caveats to keep in mind when using persistent connections. One is that when using table locking on a persistent connection, if the script for whatever reason cannot release the lock, then subsequent scripts using the same connection will block indefinitely and may require that you either restart the httpd server or the database server. Another is that when using transactions, a transaction block will also carry over to the next script which uses that connection if script execution ends before the transaction block does. In either case, you can use **register_shutdown_function()** to register a

simple cleanup function to unlock your tables or roll back your transactions. Better yet, avoid the problem entirely by not using persistent connections in scripts which use table locks or transactions (you can still use them elsewhere).

An important summary. Persistent connections were designed to have one-to-one mapping to regular connections. That means that you should *always* be able to replace persistent connections with non-persistent connections, and it won't change the way your script behaves. It *may* (and probably will) change the efficiency of the script, but not its behavior!

See also **fbsql_pconnect()**, **ibase_pconnect()**, **ifx_pconnect()**, **imap_popen()**, **ingres_pconnect()**, **msql_pconnect()**, **mssql_pconnect()**, **mysql_pconnect()**, **OCIPLogon()**, **odbc_pconnect()**, **Ora_pLogon()**, **pfsockopen()**, **pg_pconnect()**, and **sybase_pconnect()**.

Chapter 22. Safe Mode

The PHP safe mode is an attempt to solve the shared-server security problem. It is architecturally incorrect to try to solve this problem at the PHP level, but since the alternatives at the web server and OS levels aren't very realistic, many people, especially ISP's, use safe mode for now.

Security and Safe Mode

Table 22-1. Security and Safe Mode Configuration Directives

Name	Default	Changeable
safe_mode	"0"	PHP_INI_SYSTEM
safe_mode_gid	"0"	PHP_INI_SYSTEM
safe_mode_include_dir	NULL	PHP_INI_SYSTEM
safe_mode_exec_dir	""	PHP_INI_SYSTEM
safe_mode_allowed_env_vars	PHP_	PHP_INI_SYSTEM
safe_mode_protected_env_vars	LD_LIBRARY_PATH	PHP_INI_SYSTEM
open_basedir	NULL	PHP_INI_SYSTEM
disable_functions	""	PHP_INI_SYSTEM

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Here is a short explanation of the configuration directives.

safe_mode **boolean**

Whether to enable PHP's safe mode. Read the Security and chapter for more information.

safe_mode_gid **boolean**

By default, Safe Mode does a UID compare check when opening files. If you want to relax this to a GID compare, then turn on safe_mode_gid. Whether to use UID (**FALSE**) or GID (**TRUE**) checking upon file access.

safe_mode_include_dir **string**

UID/GID checks are bypassed when including files from this directory and its subdirectories (directory must also be in include_path or full path must including).

As of PHP 4.2.0, this directive can take a semi-colon separated path in a similar fashion to the include_path directive, rather than just a single directory.

The restriction specified is actually a prefix, not a directory name. This means that "safe_mode_include_dir = /dir/incl" also allows access to "/dir/include" and "/dir/incls" if they exist. When you want to restrict access to only the specified directory, end with a slash. For example: "safe_mode_include_dir = /dir/incl/"

safe_mode_exec_dir **string**

If PHP is used in safe mode, **system()** and the other functions executing system programs refuse to start programs that are not in this directory.

safe_mode_allowed_env_vars **string**

Setting certain environment variables may be a potential security breach. This directive contains a comma-delimited list of

prefixes. In Safe Mode, the user may only alter environment variables whose names begin with the prefixes supplied here. By default, users will only be able to set environment variables that begin with PHP_ (e.g. PHP_FOO=BAR).

Note: If this directive is empty, PHP will let the user modify ANY environment variable!

safe_mode_protected_env_vars **string**

This directive contains a comma-delimited list of environment variables that the end user won't be able to change using **putenv()**. These variables will be protected even if safe_mode_allowed_env_vars is set to allow to change them.

open_basedir **string**

Limit the files that can be opened by PHP to the specified directory-tree, including the file itself. This directive is *NOT* affected by whether Safe Mode is turned On or Off.

When a script tries to open a file with, for example, fopen or gzopen, the location of the file is checked. When the file is outside the specified directory-tree, PHP will refuse to open it. All symbolic links are resolved, so it's not possible to avoid this restriction with a symlink.

The special value . indicates that the directory in which the script is stored will be used as base-directory.

Under Windows, separate the directories with a semicolon. On all other systems, separate the directories with a colon. As an Apache module, open_basedir paths from parent directories are now automatically inherited.

The restriction specified with open_basedir is actually a prefix, not a directory name. This means that "open_basedir = /dir/incl" also allows access to "/dir/include" and "/dir/incls" if they exist. When you want to restrict access to only the specified directory, end with a slash. For example: "open_basedir = /dir/incl/"

Note: Support for multiple directories was added in 3.0.7.

The default is to allow all files to be opened.

disable_functions **string**

This directive allows you to disable certain functions for security reasons. It takes on a comma-dilimited list of function names. disable_functions is not affected by Safe Mode.

This directive must be set in `php.ini` For example, you cannot set this in `httpd.conf`.

See also: register_globals, display_errors, and log_errors

When safe_mode is on, PHP checks to see if the owner of the current script matches the owner of the file to be operated on by a file function. For example:

```
-rw-rw-r--    1 rasmus   rasmus       33 Jul  1 19:20 script.php
-rw-r--r--    1 root     root       1116 May 26 18:01 /etc/passwd
```

Running this script.php

```
<?php
 readfile('/etc/passwd');
?>
```

results in this error when safe mode is enabled:

```
Warning: SAFE MODE Restriction in effect. The script whose uid is 500 is not
allowed to access /etc/passwd owned by uid 0 in /docroot/script.php on line 2
```

However, there may be environments where a strict `UID` check is not appropriate and a relaxed `GID` check is sufficient. This is supported by means of the safe_mode_gid switch. Setting it to `On` performs the relaxed `GID` checking, setting it to `Off` (the default) performs `UID` checking.

If instead of safe_mode, you set an open_basedir directory then all file operations will be limited to files under the specified directory For example (Apache httpd.conf example):

```
<Directory /docroot>
  php_admin_value open_basedir /docroot
</Directory>
```

If you run the same script.php with this open_basedir setting then this is the result:

```
Warning: open_basedir restriction in effect. File is in wrong directory in
/docroot/script.php on line 2
```

You can also disable individual functions. Note that the disable_functions directive can not be used outside of the `php.ini` file which means that you cannot disable functions on a per-virtualhost or per-directory basis in your httpd.conf file. If we add this to our `php.ini` file:

```
disable_functions readfile,system
```

Then we get this output:

```
Warning: readfile() has been disabled for security reasons in
```

```
/docroot/script.php on line 2
```

Functions restricted/disabled by safe mode

This is a still probably incomplete and possibly incorrect listing of the functions limited by safe mode.

Table 22-2. Safe mode limited functions

Function	Limitations
dbmopen()	Checks whether the files or directories you are about to operate on have the same UID (owner) as the script that is being executed.
dbase_open()	Checks whether the files or directories you are about to operate on have the same UID (owner) as the script that is being executed.
filepro()	Checks whether the files or directories you are about to operate on have the same UID (owner) as the script that is being executed.
filepro_rowcount()	Checks whether the files or directories you are about to operate on have the same UID (owner) as the script that is being executed.
filepro_retrieve()	Checks whether the files or directories you are about to operate on have the same UID (owner) as the script that is being executed.
ifx_*()	sql_safe_mode restrictions, (!= safe mode)
ingres_*()	sql_safe_mode restrictions, (!= safe mode)
mysql_*()	sql_safe_mode restrictions, (!= safe mode)
pg_loimport()	Checks whether the files or directories you are about to operate on have the same UID (owner) as the script that is being executed.
posix_mkfifo()	Checks whether the directory in which you are about to operate has the same UID (owner) as the script that is being executed.
putenv()	Obeys the safe_mode_protected_env_vars and safe_mode_allowed_env_vars ini-directives. See also the documentation on **putenv()**
move_uploaded_file()	Checks whether the files or directories you are about to operate on have the same UID (owner) as the script that is being executed.
chdir()	Checks whether the directory in which you are about to operate has the same UID (owner) as the script that is being executed.
dl()	This function is disabled in safe mode.
backtick operator	This function is disabled in safe mode.
shell_exec() (functional equivalent of backticks)	This function is disabled in safe mode.
exec()	You can only execute executables within the safe_mode_exec_dir. For practical reasons it's currently not allowed to have . . components in the path to the executable.
system()	You can only execute executables within the safe_mode_exec_dir. For practical reasons it's currently not allowed to have . . components in the path to the executable.
passthru()	You can only execute executables within the safe_mode_exec_dir. For practical reasons it's currently not allowed to have . . components in the path to the executable.
popen()	You can only execute executables within the safe_mode_exec_dir. For practical reasons it's currently not allowed to have . . components in the path to the executable.
mkdir()	Checks whether the directory in which you are about to operate has the same UID (owner) as the script that is being executed.
rmdir()	Checks whether the files or directories you are about to operate on have the same UID (owner) as the script that is being executed.
rename()	Checks whether the files or directories you are about to operate on have the same UID (owner) as the script that is being executed. Checks whether the directory in which you are about to operate has the same UID (owner) as the script that is being executed.
unlink()	Checks whether the files or directories you are about to operate on have the same UID (owner) as the script that is being executed. Checks whether the directory in which you are about to operate has the same UID (owner) as the script that is being executed.
copy()	Checks whether the files or directories you are about to operate on have the same UID (owner) as the script that is being executed. Checks whether the directory in which you are about to operate has the same UID (owner) as the script that is being executed. (on *source* and *target*)

Function	Limitations
chgrp()	Checks whether the files or directories you are about to operate on have the same UID (owner) as the script that is being executed.
chown()	Checks whether the files or directories you are about to operate on have the same UID (owner) as the script that is being executed.
chmod()	Checks whether the files or directories you are about to operate on have the same UID (owner) as the script that is being executed. In addition, you cannot set the SUID, SGID and sticky bits
touch()	Checks whether the files or directories you are about to operate on have the same UID (owner) as the script that is being executed. Checks whether the directory in which you are about to operate has the same UID (owner) as the script that is being executed.
symlink()	Checks whether the files or directories you are about to operate on have the same UID (owner) as the script that is being executed. Checks whether the directory in which you are about to operate has the same UID (owner) as the script that is being executed. (note: only the target is checked)
link()	Checks whether the files or directories you are about to operate on have the same UID (owner) as the script that is being executed. Checks whether the directory in which you are about to operate has the same UID (owner) as the script that is being executed. (note: only the target is checked)
getallheaders()	In safe mode, headers beginning with 'authorization' (case-insensitive) will not be returned. Warning: this is broken with the aol-server implementation of getallheaders()!
header()	In safe mode, the uid of the script is added to the `realm` part of the `WWW-Authenticate` header if you set this header (used for HTTP Authentication).
PHP_AUTH variables	In safe mode, the variables `PHP_AUTH_USER`, `PHP_AUTH_PW`, and `PHP_AUTH_TYPE` are not available in `$_SERVER`. Regardless, you can still use `REMOTE_USER` for the USER. (note: only affected since PHP 4.3.0)
highlight_file(), show_source()	Checks whether the files or directories you are about to operate on have the same UID (owner) as the script that is being executed. Checks whether the directory in which you are about to operate has the same UID (owner) as the script that is being executed. (note: only affected since PHP 4.2.1)
parse_ini_file()	Checks whether the files or directories you are about to operate on have the same UID (owner) as the script that is being executed. Checks whether the directory in which you are about to operate has the same UID (owner) as the script that is being executed. (note: only affected since PHP 4.2.1)
Any function that uses `php4/main/fopen_wrappers.c`	??

Chapter 23. Using PHP from the command line

As of version 4.3.0, `PHP` supports a new `SAPI` type (Server Application Programming Interface) named `CLI` which means *Command Line Interface*. As the name implies, this `SAPI` type main focus is on developing shell (or desktop as well) applications with `PHP`. There are quite a few differences between the `CLI SAPI` and other `SAPI`s which are explained in this chapter. It's worth mentioning that `CLI` and `CGI` are different SAPI's although they do share many of the same behaviors.

The `CLI SAPI` was released for the first time with `PHP 4.2.0`, but was still experimental and had to be explicitly enabled with `--enable-cli` when running `./configure`. Since `PHP 4.3.0` the `CLI SAPI` is no longer experimental and the option `--enable-cli` is on by default. You may use `--disable-cli` to disable it.

As of PHP 4.3.0, the name, location and existence of the CLI/CGI binaries will differ depending on how PHP is installed on your system. By default when executing `make`, both the CGI and CLI are built and placed as `sapi/cgi/php` and `sapi/cli/php` respectively, in your php source directory. You will note that both are named `php`. What happens during `make install` depends on your configure line. If a module SAPI is chosen during configure, such as apxs, or the `--disable-cgi` option is used, the CLI is copied to `{PREFIX}/bin/php` during `make install` otherwise the CGI is placed there. So, for example, if `--with--apxs` is in your configure line then the CLI is copied to `{PREFIX}/bin/php` during `make install`. If you want to override the installation of the CGI binary, use `make install-cli` after `make install`. Alternatively you can specify `--disable-cgi` in your configure line.

> **Note:** Because both `--enable-cli` and `--enable-cgi` are enabled by default, simply having `--enable-cli` in your configure line does not necessarily mean the CLI will be copied as `{PREFIX}/bin/php` during `make install`.

The windows package distributes the CGI as php.exe and has a folder named cli with the CLI in it, so: `cli/php.exe`.

> **What SAPI do I have?:** From a shell, typing `php -v` will tell you whether `php` is CGI or CLI. See also the function php_sapi_name() and the constant **`PHP_SAPI`**.

Remarkable differences of the CLI SAPI compared to other SAPIs:

- Unlike the CGI SAPI, no headers are written to the output.

  Though the CGI SAPI provides a way to suppress HTTP headers, there's no equivalent switch to enable them in the CLI SAPI.

  CLI is started up in quiet mode by default, though the -q switch is kept for compatibility so that you can use older CGI scripts.

  It does not change the working directory to that of the script. (-C switch kept for compatibility)

  Plain text error messages (no HTML formatting).

- There are certain php.ini directives which are overriden by the CLI SAPI because they do not make sense in shell environments:

  Table 23-1. Overriden php.ini directives

Directive	CLI SAPI default value	Comment
html_errors	FALSE	It can be quite hard to read the error message in your shell when it's cluttered with all those meaningless HTML tags, therefore this directive defaults to FALSE.
implicit_flush	TRUE	It is desired that any output coming from **print()**, **echo()** and friends is immediately written to the output and not cached in any buffer. You still can use output buffering if you want to defer or manipulate standard output.
max_execution_time	0 (unlimited)	Due to endless possibilities of using PHP in shell environments, the maximum execution time has been set to unlimited. Whereas applications written for the web are often executed very quickly, shell application tend to have a much longer execution time.
register_argc_argv	TRUE	Because this setting is TRUE you will always have access to *argc* (number of arguments passed to the application) and *argv* (array of the actual arguments) in the CLI SAPI. As of PHP 4.3.0, the PHP variables $argc and $argv are registered and filled in with the appropriate values when using the CLI SAPI. Prior to this version, the creation of these variables behaved as they do in CGI and MODULE versions which requires the PHP directive register_globals to be *on*. Regardless of version or register_globals setting, you can always go through either $_SERVER or $HTTP_SERVER_VARS. Example: $_SERVER['argv']

  Note: These directives cannot be initialized with another value from the configuration file php.ini or a custom one (if specified). This is a limitation because those default values are applied after all configuration files have been parsed. However, their value can be changed during runtime (which does not make sense for all of those directives, e.g. register_argc_argv).

- To ease working in the shell environment, the following constants are defined:

  Table 23-2. CLI specific Constants

Constant	Description
STDIN	An already opened stream to stdin. This saves opening it with `$stdin = fopen('php://stdin', 'r');`
STDOUT	An already opened stream to stdout. This saves opening it with `$stdout = fopen('php://stdout', 'w');`
STDERR	An already opened stream to stderr. This saves opening it with `$stderr = fopen('php://stderr', 'w');`

  Given the above, you don't need to open e.g. a stream for stderr yourself but simply use the constant instead of the stream resource:

```
php -r 'fwrite(STDERR, "stderr\n");'
```

  You do not need to explicitly close these streams, as they are closed automatically by PHP when your script ends.

- The CLI SAPI does **not** change the current directory to the directory of the executed script!

  Example showing the difference to the CGI SAPI:

```
<?php
```

```
    /* Our simple test application named test.php*/
    echo getcwd(), "\n";
?>
```

When using the CGI version, the output is:

```
$ pwd
/tmp

$ php -q another_directory/test.php
/tmp/another_directory
```

This clearly shows that PHP changes its current directory to the one of the executed script.

Using the CLI SAPI yields:

```
$ pwd
/tmp

$ php -f another_directory/test.php
/tmp
```

This allows greater flexibility when writing shell tools in PHP.

> **Note:** The CGI SAPI supports the CLI SAPI behaviour by means of the -C switch when run from the command line.

The list of command line options provided by the PHP binary can be queried anytime by running PHP with the -h switch:

```
Usage: php [options] [-f] <file> [args...]
       php [options] -r <code> [args...]
       php [options] [-- args...]
  -s               Display colour syntax highlighted source.
  -w               Display source with stripped comments and whitespace.
  -f <file>        Parse <file>.
  -v               Version number
  -c <path>|<file> Look for php.ini file in this directory
  -a               Run interactively
  -d foo[=bar]     Define INI entry foo with value 'bar'
  -e               Generate extended information for debugger/profiler
  -z <file>        Load Zend extension <file>.
  -l               Syntax check only (lint)
  -m               Show compiled in modules
  -i               PHP information
  -r <code>        Run PHP <code> without using script tags <?..?>
  -h               This help

  args...          Arguments passed to script. Use -- args when first argument
                   starts with - or script is read from stdin
```

The CLI SAPI has three different ways of getting the PHP code you want to execute:

1. Telling PHP to execute a certain file.

   ```
   php my_script.php

   php -f my_script.php
   ```

   Both ways (whether using the -f switch or not) execute the file my_script.php. You can choose any file to execute - your PHP scripts do not have to end with the .php extension but can have any name or extension you wish.

2. Pass the PHP code to execute directly on the command line.

   ```
   php -r 'print_r(get_defined_constants());'
   ```

   Special care has to be taken in regards of shell variable substitution and quoting usage.

   > **Note:** Read the example carefully, there are no beginning or ending tags! The -r switch simply does not need them. Using them will lead to a parser error.

3. Provide the PHP code to execute via standard input (stdin).

   This gives the powerful ability to dynamically create PHP code and feed it to the binary, as shown in this (fictional) example:

   ```
   $ some_application | some_filter | php | sort -u >final_output.txt
   ```

You cannot combine any of the three ways to execute code.

Like every shell application, the PHP binary accepts a number of arguments but your PHP script can also receive arguments. The number of arguments which can be passed to your script is not limited by PHP (the shell has a certain size limit in the number of characters which can be passed; usually you won't hit this limit). The arguments passed to your script are available in the global array $argv. The zero index always contains the script name (which is - in case the PHP code is coming from either standard input or from the command line switch -r). The second registered global variable is $argc which contains the number of elements in the $argv array (**not** the number of arguments passed to the script).

As long as the arguments you want to pass to your script do not start with the - character, there's nothing special to watch out for. Passing an argument to your script which starts with a - will cause trouble because PHP itself thinks it has to handle it. To prevent this, use the argument list separator --. After this separator has been parsed by PHP, every argument following it is passed untouched to your script.

```
# This will not execute the given code but will show the PHP usage
$ php -r 'var_dump($argv);' -h
Usage: php [options] [-f] <file> [args...]
[...]

# This will pass the '-h' argument to your script and prevent PHP from showing it's usage
$ php -r 'var_dump($argv);' -- -h
array(2) {
  [0]=>
  string(1) "-"
  [1]=>
  string(2) "-h"
}
```

However, there's another way of using PHP for shell scripting. You can write a script where the first line starts with #!/usr/bin/php. Following this you can place normal PHP code included within the PHP starting and end tags. Once you have set the execution attributes of the file appropriately (e.g. chmod +x test) your script can be executed like a normal shell or perl script:

```
#!/usr/bin/php
<?php
    var_dump($argv);
?>
```

Assuming this file is named test in the current directory, we can now do the following:

```
$ chmod 755 test
$ ./test -h -- foo
array(4) {
  [0]=>
  string(6) "./test"
  [1]=>
  string(2) "-h"
  [2]=>
  string(2) "--"
  [3]=>
  string(3) "foo"
}
```

As you see, in this case no care needs to be taken when passing parameters which start with - to your script.

Table 23-3. Command line options

Option	Description
-s	Display colour syntax highlighted source. This option uses the internal mechanism to parse the file and produces a HTML highlighted version of it and writes it to standard output. Note that all it does it to generate a block of <code> [...] </code> HTML tags, no HTML headers. **Note:** This option does not work together with the -r option.
-w	Display source with stripped comments and whitespace. **Note:** This option does not work together with the -r option.
-f	Parses and executed the given filename to the -f option. This switch is optional and can be left out. Only providing the filename to execute is sufficient.

-v	Writes the PHP, PHP SAPI, and Zend version to standard output, e.g.
	`$ php -v`
	`PHP 4.3.0 (cli), Copyright (c) 1997-2002 The PHP Group`
	`Zend Engine v1.3.0, Copyright (c) 1998-2002 Zend Technologies`

-c	With this option one can either specify a directory where to look for php.ini or you can specify a custom INI file directly (which does not need to be named php.ini), e.g.:
	`$ php -c /custom/directory/ my_script.php`
	
	`$ php -c /custom/directory/custom-file.ini my_script.php`

-a	Runs PHP interactively.

-d	This option allows you to set a custom value for any of the configuration directives allowed in php.ini. The syntax is:
	`-d configuration_directive[=value]`
	
	Examples:
	`# Omitting the value part will set the given configuration directive to "1"`
	`$ php -d max_execution_time -r '$foo = ini_get("max_execution_time"); var_dump($foo);'`

Option	Description

	`string(1) "1"`
	
	`# Passing an empty value part will set the configuration directive to ""`
	`php -d max_execution_time= -r '$foo = ini_get("max_execution_time"); var_dump($foo);'`
	`string(0) ""`
	
	`# The configuration directive will be set to anything passed after the '=' character`
	`$  php -d max_execution_time=20 -r '$foo = ini_get("max_execution_time"); var_dump($foo);'`
	`string(2) "20"`
	`$  php -d max_execution_time=doesntmakesense -r '$foo = ini_get("max_execution_time"); var_dump($foo);'`
	`string(15) "doesntmakesense"`

-e	Generate extended information for debugger/profiler.
-z	Load Zend extension. If only a filename is given, PHP tries to load this extension from the current default library path on your system (usually specified `/etc/ld.so.conf` on Linux systems). Passing a filename with an absolute path information will not use the systems library search path. A relative filename with a directory information will tell `PHP` only to try to load the extension relative to the current directory.
-l	This option provides a convenient way to only perform a syntax check on the given `PHP` code. On succes, the text `No syntax errors detected in <filename>` is written to standard output and the shell return code is `0`. On failure, the text `Errors parsing <filename>` in addition to the internal parser error message is written to standard output and the shell return code is set to `255`. This option won't find fatal errors (like undefined functions). Use `-f` if you would like to test for fatal errors too. **Note:** This option does not work together with the `-r` option.

-m	Using this option, PHP prints out the built in (and loaded) PHP and Zend modules:
	`$ php -m`
	`[PHP Modules]`
	`xml`
	`tokenizer`
	`standard`
	`session`
	`posix`
	`pcre`
	`overload`
	`mysql`
	`mbstring`
	`ctype`
	
	`[Zend Modules]`

-i	This command line option calls **phpinfo()**, and prints out the results. If `PHP` is not working correctly, it is advisable to use `php -i` and see whether any error messages are printed out before or in place of the information tables. Beware that the output is in `HTML` and therefore quite huge.

Option	Description

-r	This option allows execution of PHP right from within the command line. The PHP start and end tags (<?php and ?>) are **not needed** and will cause a parser error if present.
	
	Note: Care has to be taken when using this form of PHP to not collide with command line variable substitution done by the shell.
	
	Example showing a parser error
	`$ php -r "$foo = get_defined_constants();"`
	`Command line code(1) : Parse error - parse error, unexpected '='`
	
	The problem here is that the sh/bash performs variable substitution even when using double quotes ". Since the variable $foo is unlikely to be defined, it expands to nothing which results in the code passed to PHP for execution actually reading:
	`$ php -r " = get_defined_constants();"`
	
	The correct way would be to use single quotes '. Variables in single-quoted strings are not expanded by sh/bash.
	`$ php -r '$foo = get_defined_constants(); var_dump($foo);'`
	`array(370) {`
	`["E_ERROR"]=>`
	`int(1)`
	`["E_WARNING"]=>`
	`int(2)`
	`["E_PARSE"]=>`
	`int(4)`
	`["E_NOTICE"]=>`
	`int(8)`
	`["E_CORE_ERROR"]=>`
	`[...]`
	
	If you are using a shell different from sh/bash, you might experience further issues. Feel free to open a bug report or send a mail to phpdoc@lists.php.net. One can still easily run into troubles when trying to get shell variables into the code or using backslashes for escaping. You've been warned.
	
	Note: -r is available in the *CLI* SAPI and not in the *CGI* SAPI.

-h	With this option, you can get information about the actual list of command line options and some one line descriptions about what they do.

The PHP executable can be used to run PHP scripts absolutely independent from the web server. If you are on a Unix system, you should add a special first line to your PHP script, and make it executable, so the system will know, what program should run the script. On a Windows platform you can associate php.exe with the double click option of the .php files, or you can make a batch file to run the script through PHP. The first line added to the script to work on Unix won't hurt on Windows, so you can write cross platform programs this way. A simple example of writing a command line PHP program can be found below.

Example 23-1. Script intended to be run from command line (script.php)

```
#!/usr/bin/php
<?php

if ($argc != 2 || in_array($argv[1], array('--help', '-help', '-h', '-?'))) {
?>

This is a command line PHP script with one option.

  Usage:
  <?php echo $argv[0]; ?> <option>

  <option> can be some word you would like
  to print out. With the --help, -help, -h,
  or -? options, you can get this help.

<?php
} else {
    echo $argv[1];
}
?>
```

In the script above, we used the special first line to indicate that this file should be run by PHP. We work with a CLI version here, so there will be no HTTP header printouts. There are two variables you can use while writing command line applications with PHP: $argc and $argv. The first is the number of arguments plus one (the name of the script running). The second is an array containing the arguments, starting with the script name as number zero ($argv[0]).

In the program above we checked if there are less or more than one arguments. Also if the argument was --help, -help, -h or -?, we printed out the help message, printing the script name dynamically. If we received some other argument we echoed that out.

If you would like to run the above script on Unix, you need to make it executable, and simply call it as script.php echothis or

`script.php -h`. On Windows, you can make a batch file for this task:

Example 23-2. Batch file to run a command line PHP script (script.bat)

```
@c:\php\cli\php.exe script.php %1 %2 %3 %4
```

Assuming you named the above program `script.php`, and you have your CLI `php.exe` in `c:\php\cli\php.exe` this batch file will run it for you with your added options: `script.bat echothis` or `script.bat -h`.

See also the Readline extension documentation for more functions you can use to enhance your command line applications in PHP.

IV. Function Reference

Table of Contents

I. Apache-specific Functions

Introduction

These functions are only available when running PHP as an Apache 1.x module.

Installation

For PHP installation on Apache 1.x see the Apache section in the installation chapter.

Runtime Configuration

The behaviour of the Apache PHP module is affected by settings in php.ini. Configuration settings from `php.ini` may be overridden by php_flag settings in the server configuration file or local `.htaccess` files.

Example 1. Turning off PHP parsing for a directory using `.htaccess`

```
php_flag engine off
```

Table 1. Apache configuration options

Name	Default	Changeable	Function
engine	On	PHP_INI_ALL	turns PHP parsing on or off
child_terminate	Off	PHP_INI_ALL	specify whether PHP scripts may request child process termination on end of request, see also **apache_child_terminate()**
last_modified	Off	PHP_INI_ALL	send PHP scripts modification date as Last-Modified: header for this request
xbit_hack	Off	PHP_INI_ALL	parse files with executable bit set as PHP regardles of their file ending

Here is a short explanation of the configuration directives.

`engine` **boolean**

This directive is really only useful in the Apache module version of PHP. It is used by sites that would like to turn PHP parsing on and off on a per-directory or per-virtual server basis. By putting **`engine off`** in the appropriate places in the `httpd.conf` file, PHP can be enabled or disabled.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Table of Contents

apache_child_terminate

(PHP 4 >= 4.0.5)

apache_child_terminate -- Terminate apache process after this request

Description

bool **apache_child_terminate** (void)

apache_child_terminate() will register the Apache process executing the current PHP request for termination once execution of PHP code it is completed. It may be used to terminate a process after a script with high memory consumption has been run as memory will usually only be freed internally but not given back to the operating system.

> **Note:** The availability of this feature is controlled by the `php.ini` directive `apache.child_terminate`, which is set to `off` by default.
>
> This feature is also not available on multithreaded versions of apache like the win32 version.

See also **exit()**.

apache_lookup_uri

(PHP 3>= 3.0.4, PHP 4)

apache_lookup_uri -- Perform a partial request for the specified URI and return all info about it

Description

object **apache_lookup_uri** (string filename)

This performs a partial request for a URI. It goes just far enough to obtain all the important information about the given resource and returns this information in a class. The properties of the returned class are:

status
the_request
status_line
method
content_type
handler
uri
filename
path_info
args
boundary
no_cache
no_local_copy
allowed
send_bodyct
bytes_sent
byterange
clength
unparsed_uri
mtime
request_time

> **Note: apache_lookup_uri()** only works when PHP is installed as an Apache module.

apache_note

(PHP 3>= 3.0.2, PHP 4)

apache_note -- Get and set apache request notes

Description

string **apache_note** (string note_name [, string note_value])

apache_note() is an Apache-specific function which gets and sets values in a request's `notes` table. If called with one argument, it returns the current value of note `note_name`. If called with two arguments, it sets the value of note `note_name` to `note_value` and returns the previous value of note `note_name`.

apache_request_headers

(PHP 4 >= 4.3.0)

apache_request_headers -- Fetch all HTTP request headers

Description

array **apache_request_headers** (void)

apache_request_headers() returns an associative array of all the HTTP headers in the current request. This is only supported when PHP runs as an Apache module.

> **Note:** Prior to PHP 4.3.0, **apache_request_headers()** was called **getallheaders()**. After PHP 4.3.0, **getallheaders()** is an alias for **apache_request_headers()**.

Example 1. apache_request_headers() Example

```
<?php
$headers = apache_request_headers();

foreach ($headers as $header => $value) {
    echo "$header: $value <br />\n";
}
?>
```

> **Note:** You can also get at the value of the common CGI variables by reading them from the environment, which works whether or not you are using PHP as an Apache module. Use **phpinfo()** to see a list of all of the available environment variables.

apache_response_headers

(PHP 4 >= 4.3.0)

apache_response_headers -- Fetch all HTTP response headers

Description

array **apache_response_headers** (void)

Returns an array of all Apache response headers. This functionality is only available in PHP version 4.3.0 and greater.

See also **getallheaders()** and **headers_sent()**.

apache_setenv

(PHP 4 >= 4.2.0)

apache_setenv -- Set an Apache subprocess_env variable

Description

int **apache_setenv** (string variable, string value [, bool walk_to_top])

Warning
This function is currently not documented; only the argument list is available.

ascii2ebcdic

(PHP 3>= 3.0.17)

ascii2ebcdic -- Translate string from ASCII to EBCDIC

Description

int **ascii2ebcdic** (string ascii_str)

ascii2ebcdic() is an Apache-specific function which is available only on EBCDIC based operating systems (OS/390, BS2000). It translates the ASCII encoded string *ascii_str* to its equivalent EBCDIC representation (binary safe), and returns the result.

See also the reverse function **ebcdic2ascii()**

ebcdic2ascii

(PHP 3>= 3.0.17)

ebcdic2ascii -- Translate string from EBCDIC to ASCII

Description

int **ebcdic2ascii** (string ebcdic_str)

ebcdic2ascii() is an Apache-specific function which is available only on EBCDIC based operating systems (OS/390, BS2000). It translates the EBCDIC encoded string *ebcdic_str* to its equivalent ASCII representation (binary safe), and returns the result.

See also the reverse function **ascii2ebcdic()**

getallheaders

(PHP 3, PHP 4)

getallheaders -- Fetch all HTTP request headers

Description

array **getallheaders** (void)

getallheaders() is an alias for **apache_request_headers()**. It will return an associative array of all the HTTP headers in the current request. Please read the **apache_request_headers()** documentation for more information on how this function works.

> **Note:** In PHP 4.3.0, **getallheaders()** became an alias for **apache_request_headers()**. Essentially, it was renamed. This is because this function only works when PHP is compiled as an Apache Module.

See also **apache_request_headers()**.

virtual

(PHP 3, PHP 4)

virtual -- Perform an Apache sub-request

Description

int **virtual** (string filename)

virtual() is an Apache-specific function which is equivalent to <!--#include virtual...--> in mod_include. It performs an Apache sub-request. It is useful for including CGI scripts or .shtml files, or anything else that you would parse through Apache. Note that for a CGI script, the script must generate valid CGI headers. At the minimum that means it must generate a Content-type

header. For PHP files, you need to use **include()** or **require()**; **virtual()** cannot be used to include a document which is itself a PHP file.

To run the sub-request, all buffers are terminated and flushed to the browser, pending headers are sent too.

II. Array Functions

Introduction

These functions allow you to interact with and manipulate arrays in various ways. Arrays are essential for storing, managing, and operating on sets of variables.

Simple and multi-dimensional arrays are supported, and may be either user created or created by another function. There are specific database handling functions for populating arrays from database queries, and several functions return arrays.

Please see the Arrays section of the manual for a detailed explanation of how arrays are implemented and used in PHP.

Requirements

No external libraries are needed to build this extension.

Installation

There is no installation needed to use these functions; they are part of the PHP core.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are always available as part of the PHP core.

CASE_LOWER (integer)

> **CASE_LOWER** is used with **array_change_key_case()** and is used to convert array keys to lower case. This is also the default case for **array_change_key_case()**.

CASE_UPPER (integer)

> **CASE_UPPER** is used with **array_change_key_case()** and is used to convert array keys to upper case.

Sorting order flags:

SORT_ASC (integer)

> **SORT_ASC** is used with **array_multisort()** to sort in ascending order.

SORT_DESC (integer)

SORT_DESC is used with **array_multisort()** to sort in descending order.

Sorting type flags: used by various sort functions

SORT_REGULAR (integer)

SORT_REGULAR is used to compare items normally.

SORT_NUMERIC (integer)

SORT_NUMERIC is used to compare items numerically.

SORT_STRING (integer)

SORT_STRING is used to compare items as strings.

See Also

See also **is_array()**, **explode()**, **implode()**, **split()**, **preg_split()**, and **join()**.

Table of Contents

next -- Advance the internal array pointer of an array
pos -- Get the current element from an array
prev -- Rewind the internal array pointer
range -- Create an array containing a range of elements
reset -- Set the internal pointer of an array to its first element
rsort -- Sort an array in reverse order
shuffle -- Shuffle an array
sizeof -- Get the number of elements in variable
sort -- Sort an array
uasort -- Sort an array with a user-defined comparison function and maintain index association
uksort -- Sort an array by keys using a user-defined comparison function
usort -- Sort an array by values using a user-defined comparison function

array_change_key_case

(PHP 4 >= 4.2.0)

array_change_key_case -- Returns an array with all string keys lowercased or uppercased

Description

array **array_change_key_case** (array input [, int case])

array_change_key_case() changes the keys in the *input* array to be all lowercase or uppercase. The change depends on the last optional *case* parameter. You can pass two constants there, **CASE_UPPER** and **CASE_LOWER**. The default is **CASE_LOWER**. The function will leave number indices as is.

Example 1. array_change_key_case() example

```
$input_array = array("FirSt" => 1, "SecOnd" => 4);
print_r(array_change_key_case($input_array, CASE_UPPER));
```

The printout of the above program will be:

```
Array
(
    [FIRST] => 1
    [SECOND] => 2
)
```

array_chunk

(PHP 4 >= 4.2.0)

array_chunk -- Split an array into chunks

Description

array **array_chunk** (array input, int size [, bool preserve_keys])

array_chunk() splits the array into several arrays with *size* values in them. You may also have an array with less values at the end. You get the arrays as members of a multidimensional array indexed with numbers starting from zero.

By setting the optional *preserve_keys* parameter to **TRUE**, you can force PHP to preserve the original keys from the input array. If you specify **FALSE** new number indices will be used in each resulting array with indices starting from zero. The default is **FALSE**.

Example 1. array_chunk() example

```
$input_array = array('a', 'b', 'c', 'd', 'e');
print_r(array_chunk($input_array, 2));
print_r(array_chunk($input_array, 2, TRUE));
```

The printout of the above program will be:

```
Array
(
```

```
    [0] => Array
        (
            [0] => a
            [1] => b
        )

    [1] => Array
        (
            [0] => c
            [1] => d
        )

    [2] => Array
        (
            [0] => e
        )

)
Array
(
    [0] => Array
        (
            [0] => a
            [1] => b
        )

    [1] => Array
        (
            [2] => c
            [3] => d
        )

    [2] => Array
        (
            [4] => e
        )

)
```

array_count_values

(PHP 4)

array_count_values -- Counts all the values of an array

Description

array **array_count_values** (array input)

array_count_values() returns an array using the values of the *input* array as keys and their frequency in *input* as values.

Example 1. array_count_values() example

```
$array = array (1, "hello", 1, "world", "hello");
print_r(array_count_values ($array));
```

The printout of the above program will be:

```
Array
(
    [1] => 2
    [hello] => 2
    [world] => 1
)
```

array_diff_assoc

(PHP 4 >= 4.3.0)

array_diff_assoc -- Computes the difference of arrays with additional index check

Description

array **array_diff_assoc** (array array1, array array2 [, array ...])

array_diff_assoc() returns an **array** containing all the values from *array1* that are not present in any of the other arguments. Note that the keys are used in the comparison unlike **array_diff()**.

Example 1. array_diff_assoc() example

```
<?php
$array1 = array ("a" => "green", "b" => "brown", "c" => "blue", "red");
$array2 = array ("a" => "green", "yellow", "red");
$result = array_diff_assoc ($array1, $array2);

/* The result is:
Array
(
    [b] => brown
    [c] => blue
    [0] => red
)
*/
?>
```

In our example above you see the `"a" => "green"` pair is present in both arrays and thus it is not in the ouput from the function. Unlike this, the pair `0 => "red"` is in the ouput because in the second argument `"red"` has key which is `1`.

Two values from *key* => *value* pairs are considered equal only if `(string) $elem1 === (string) $elem2` . In other words a strict check takes place so the string representations must be the same.

> **Note:** Please note that this function only checks one dimension of a n-dimensional array. Of course you can check deeper dimensions by using, for example, `array_diff_assoc($array1[0], $array2[0]);`.

See also **array_diff()**, **array_intersect()**, and **array_intersect_assoc()**.

array_diff

(PHP 4 >= 4.0.1)

array_diff -- Computes the difference of arrays

Description

array **array_diff** (array array1, array array2 [, array ...])

array_diff() returns an array containing all the values of *array1* that are not present in any of the other arguments. Note that keys are preserved.

Example 1. array_diff() example

```
$array1 = array ("a" => "green", "red", "blue", "red");
$array2 = array ("b" => "green", "yellow", "red");
$result = array_diff ($array1, $array2);
```

This makes `$result` have `array ("blue");`. Multiple occurrences in $array1 are all treated the same way.

> **Note:** Two elements are considered equal if and only if `(string) $elem1 === (string) $elem2`. In words: when the string representation is the same.

> **Note:** Please note that this function only checks one dimension of a n-dimensional array. Of course you can check deeper dimensions by using `array_diff($array1[0], $array2[0]);`.

Warning
This was broken in PHP 4.0.4!

See also **array_diff_assoc()**, **array_intersect()** and **array_intersect_assoc()**.

array_fill

(PHP 4 >= 4.2.0)

array_fill -- Fill an array with values

Description

array **array_fill** (int start_index, int num, mixed value)

array_fill() fills an array with *num* entries of the value of the *value* parameter, keys starting at the *start_index* parameter.

Example 1. array_fill() example

```
$a = array_fill(5, 6, 'banana');
```

$a now has the following entries using **print_r()**:

```
Array
(
    [5]  => banana
    [6]  => banana
    [7]  => banana
    [8]  => banana
    [9]  => banana
    [10] => banana
)
```

array_filter

(PHP 4 >= 4.0.6)

array_filter -- Filters elements of an array using a callback function

Description

array **array_filter** (array input [, callback function])

array_filter() returns an array containing all the elements of *input* filtered according a callback *function*. If the *input* is an associative array the keys are preserved.

Example 1. array_filter() example

```
function odd($var) {
    return ($var % 2 == 1);
}

function even($var) {
    return ($var % 2 == 0);
}

$array1 = array ("a"=>1, "b"=>2, "c"=>3, "d"=>4, "e"=>5);
$array2 = array (6, 7, 8, 9, 10, 11, 12);

echo "Odd :\n";
print_r(array_filter($array1, "odd"));
echo "Even:\n";
print_r(array_filter($array2, "even"));
```

The printout of the program above will be:

```
Odd :
Array
(
    [a] => 1
    [c] => 3
    [e] => 5
)
Even:
Array
(
    [0] => 6
    [2] => 8
    [4] => 10
    [6] => 12
)
```

Users may not change the array itself from the callback function. e.g. Add/delete an element, unset the array that **array_filter()** is applied to. If the array is changed, the behavior of this function is undefined.

See also **array_map()** and **array_reduce()**.

array_flip

(PHP 4)

array_flip -- Exchanges all keys with their associated values in an array

Description

array **array_flip** (array trans)

array_flip() returns an **array** in flip order, i.e. keys from *trans* become values and *trans*'s values become keys.

Note that the values of *trans* need to be valid keys, i.e. they need to be either **integer** or **string**. A warning will be emitted if a value has the wrong type, and the key/value pair in question *will not be flipped*.

If a value has several occurrences, the latest key will be used as its values, and all others will be lost.

array_flip() returns **FALSE** if it fails.

Example 1. array_flip() example

```
$trans = array_flip ($trans);
$original = strtr ($str, $trans);
```

Example 2. array_flip() example : collision

```
$trans = array ("a" => 1, "b" => 1, "c" => 2);
$trans = array_flip ($trans);
print_r($trans);
```

now $trans is :

```
Array
(
    [1] => b
    [2] => c
)
```

array_intersect_assoc

(PHP 4 >= 4.3.0)

array_intersect_assoc -- Computes the intersection of arrays with additional index check

Description

array **array_intersect_assoc** (array array1, array array2 [, array ...])

array_intersect_assoc() returns an array containing all the values of *array1* that are present in all the arguments. Note that the keys are used in the comparison unlike in **array_intersect()**.

Example 1. array_intersect_assoc() example

```
<?php
$array1 = array ("a" => "green", "b" => "brown", "c" => "blue", "red");
$array2 = array ("a" => "green", "yellow", "red");
$result_array = array_intersect_assoc ($array1, $array2);

/* $result_array will look like:

Array
(
    [a] => green
)

*/
```

```
?>
```

In our example you see that only the pair `"a" => "green"` is present in both arrays and thus is returned. The value `"red"` is not returned because in `$array1` it's key is `2` while the key of "red" in `$array2` it is `1`.

The two values from the `key => value` pairs are considered equal only if `(string) $elem1 === (string) $elem2` . In otherwords a strict type check is executed so the string representation must be the same.

See also **array_intersect()**, **array_diff()** and **array_diff_assoc()**.

array_intersect

(PHP 4 >= 4.0.1)

array_intersect -- Computes the intersection of arrays

Description

array **array_intersect** (array array1, array array2 [, array ...])

array_intersect() returns an array containing all the values of *array1* that are present in all the arguments. Note that keys are preserved.

Example 1. array_intersect() example

```
$array1 = array ("a" => "green", "red", "blue");
$array2 = array ("b" => "green", "yellow", "red");
$result = array_intersect ($array1, $array2);
```

This makes `$result` have

```
Array
(
    [a] => green
    [0] => red
)
```

> **Note:** Two elements are considered equal if and only if `(string) $elem1 === (string) $elem2`. In words: when the string representation is the same.

Warning
This was broken in PHP 4.0.4!

See also **array_intersect_assoc()**, **array_diff()**, **array_diff_assoc()**.

array_key_exists

(PHP 4 >= 4.1.0)

array_key_exists -- Checks if the given key or index exists in the array

Description

bool **array_key_exists** (mixed key, array search)

array_key_exists() returns **TRUE** if the given *key* is set in the array. *key* can be any value possible for an array index.

Example 1. array_key_exists() example

```
$search_array = array("first" => 1, "second" => 4);
if (array_key_exists("first", $search_array)) {
    echo "The 'first' element is in the array";
}
```

> **Note:** The name of this function is **key_exists()** in PHP version 4.0.6.

See also **isset()**.

array_keys

(PHP 4)

array_keys -- Return all the keys of an array

Description

array **array_keys** (array input [, mixed search_value])

array_keys() returns the keys, numeric and string, from the *input* array.

If the optional *search_value* is specified, then only the keys for that value are returned. Otherwise, all the keys from the *input* are returned.

Example 1. array_keys() example

```
$array = array (0 => 100, "color" => "red");
print_r(array_keys ($array));

$array = array ("blue", "red", "green", "blue", "blue");
print_r(array_keys ($array, "blue"));

$array = array ("color" => array("blue", "red", "green"), "size" => array("small", "medium", "large"));
print_r(array_keys ($array));
```

The printout of the program above will be:

```
Array
(
    [0] => 0
    [1] => color
)
Array
(
    [0] => 0
    [1] => 3
    [2] => 4
)
Array
(
    [0] => color
    [1] => size
)
```

Note: This function was added to PHP 4, below is an implementation for those still using PHP 3.

Example 2. Implementation of array_keys() for PHP 3 users

```
function array_keys ($arr, $term="") {
    $t = array();
    while (list($k,$v) = each($arr)) {
        if ($term && $v != $term) {
            continue;
        }
        $t[] = $k;
    }
    return $t;
}
```

See also **array_values()**.

array_map

(PHP 4 >= 4.0.6)

array_map -- Applies the callback to the elements of the given arrays

Description

array **array_map** (callback function, array arr1 [, array arr2...])

array_map() returns an array containing all the elements of *arr1* after applying the callback *function* to each one. The number of parameters that the callback *function* accepts should match the number of arrays passed to the **array_map()**

Example 1. array_map() example

```
<?php
function cube($n) {
    return $n*$n*$n;
}

$a = array(1, 2, 3, 4, 5);
$b = array_map("cube", $a);
print_r($b);
?>
```

This makes `$b` have:

```
Array
(
    [0] => 1
    [1] => 8
    [2] => 27
    [3] => 64
    [4] => 125
)
```

Example 2. array_map() - using more arrays

```
<?php
function show_Spanish($n, $m) {
    return "The number $n is called $m in Spanish";
}

function map_Spanish($n, $m) {
    return array ($n => $m);
}

$a = array(1, 2, 3, 4, 5);
$b = array("uno", "dos", "tres", "cuatro", "cinco");

$c = array_map("show_Spanish", $a, $b);
print_r($c);

$d = array_map("map_Spanish", $a , $b);
print_r($d);
?>
```

This results:

```
// printout of $c
Array
(
    [0] => The number 1 is called uno in Spanish
    [1] => The number 2 is called dos in Spanish
    [2] => The number 3 is called tres in Spanish
    [3] => The number 4 is called cuatro in Spanish
    [4] => The number 5 is called cinco in Spanish
)

// printout of $d
Array
(
    [0] => Array
        (
            [1] => uno
        )

    [1] => Array
        (
            [2] => dos
        )

    [2] => Array
        (
            [3] => tres
        )

    [3] => Array
        (
            [4] => cuatro
        )

    [4] => Array
        (
            [5] => cinco
        )

)
```

Usually when using two or more arrays, they should be of equal length because the callback function is applied in parallel to the corresponding elements. If the arrays are of unequal length, the shortest one will be extended with empty elements.

An interesting use of this function is to construct an array of arrays, which can be easily performed by using **NULL** as the name of the callback function

Example 3. Creating an array of arrays

```
<?php
$a = array(1, 2, 3, 4, 5);
$b = array("one", "two", "three", "four", "five");
$c = array("uno", "dos", "tres", "cuatro", "cinco");

$d = array_map(null, $a, $b, $c);
print_r($d);
?>
```

The printout of the program above will be:

```
Array
(
    [0] => Array
        (
            [0] => 1
            [1] => one
```

```
            [2] => uno
        )

    [1] => Array
        (
            [0] => 2
            [1] => two
            [2] => dos
        )

    [2] => Array
        (
            [0] => 3
            [1] => three
            [2] => tres
        )

    [3] => Array
        (
            [0] => 4
            [1] => four
            [2] => cuatro
        )

    [4] => Array
        (
            [0] => 5
            [1] => five
            [2] => cinco
        )

)
```

See also **array_filter()**, **array_reduce()**, and **array_walk()**.

array_merge_recursive

(PHP 4 >= 4.0.1)

array_merge_recursive -- Merge two or more arrays recursively

Description

array **array_merge_recursive** (array array1, array array2 [, array ...])

array_merge_recursive() merges the elements of two or more arrays together so that the values of one are appended to the end of the previous one. It returns the resulting array.

If the input arrays have the same string keys, then the values for these keys are merged together into an array, and this is done recursively, so that if one of the values is an array itself, the function will merge it with a corresponding entry in another array too. If, however, the arrays have the same numeric key, the later value will not overwrite the original value, but will be appended.

Example 1. array_merge_recursive() example

```
$ar1 = array ("color" => array ("favorite" => "red"), 5);
$ar2 = array (10, "color" => array ("favorite" => "green", "blue"));
$result = array_merge_recursive ($ar1, $ar2);
```

The `$result` will be:

```
Array
(
    [color] => Array
        (
            [favorite] => Array
                (
                    [0] => red
                    [1] => green
                )

            [0] => blue
        )

    [0] => 5
    [1] => 10
)
```

See also **array_merge()**.

array_merge

(PHP 4)

array_merge -- Merge two or more arrays

Description

array **array_merge** (array array1, array array2 [, array ...])

array_merge() merges the elements of two or more arrays together so that the values of one are appended to the end of the previous one. It returns the resulting array.

If the input arrays have the same string keys, then the later value for that key will overwrite the previous one. If, however, the arrays contain numeric keys, the later value will **not** overwrite the original value, but will be appended.

Example 1. array_merge() example

```
$array1 = array ("color" => "red", 2, 4);
$array2 = array ("a", "b", "color" => "green", "shape" => "trapezoid", 4);
$result = array_merge ($array1, $array2);
```

The `$result` will be:

```
Array
(
    [color] => green
    [0] => 2
    [1] => 4
    [2] => a
    [3] => b
    [shape] => trapezoid
    [4] => 4
)
```

Example 2. Simple array_merge() example

```
$array1 = array();
$array2 = array(1 => "data");
$result = array_merge($array1, $array2);
```

Don't forget that numeric keys will be renumbered!

```
Array
(
    [0] => data
)
```

If you want to completely preserve the arrays and just want to append them to each other, use the + operator:

```
$array1 = array();
$array2 = array(1 => "data");
$result = $array1 + $array2;
```

The numeric key will be preserved and thus the association remains.

```
Array
(
    [1] => data
)
```

Note: Shared keys will be overwritten on a first-come first-served basis.

See also **array_merge_recursive()**.

array_multisort

(PHP 4)

array_multisort -- Sort multiple or multi-dimensional arrays

Description

bool **array_multisort** (array ar1 [, mixed arg [, mixed ... [, array ...]]])

array_multisort() can be used to sort several arrays at once or a multi-dimensional array according by one of more dimensions. It maintains key association when sorting.

The input arrays are treated as columns of a table to be sorted by rows - this resembles the functionality of SQL ORDER BY clause. The first array is the primary one to sort by. The rows (values) in that array that compare the same are sorted by the next input array, and so on.

The argument structure of this function is a bit unusual, but flexible. The very first argument has to be an array. Subsequently, each argument can be either an array or a sorting flag from the following lists.

Sorting order flags:

- SORT_ASC - sort in ascending order
- SORT_DESC - sort in descending order

Sorting type flags:

- SORT_REGULAR - compare items normally
- SORT_NUMERIC - compare items numerically
- SORT_STRING - compare items as strings

No two sorting flags of the same type can be specified after each array. The sorting flags specified after an array argument apply only to that array - they are reset to default SORT_ASC and SORT_REGULAR before each new array argument.

Returns **TRUE** on success or **FALSE** on failure.

Example 1. Sorting multiple arrays

```
$ar1 = array ("10", 100, 100, "a");
$ar2 = array (1, 3, "2", 1);
array_multisort ($ar1, $ar2);
```

In this example, after sorting, the first array will contain 10, "a", 100, 100. The second array will contain 1, 1, "2", 3. The entries in the second array corresponding to the identical entries in the first array (100 and 100) were sorted as well.

Example 2. Sorting multi-dimensional array

```
$ar = array (array ("10", 100, 100, "a"), array (1, 3, "2", 1));
array_multisort ($ar[0], SORT_ASC, SORT_STRING,
                 $ar[1], SORT_NUMERIC, SORT_DESC);
```

In this example, after sorting, the first array will contain 10, 100, 100, "a" (it was sorted as strings in ascending order), and the second one will contain 1, 3, "2", 1 (sorted as numbers, in descending order).

array_pad

(PHP 4)

array_pad -- Pad array to the specified length with a value

Description

array **array_pad** (array input, int pad_size, mixed pad_value)

array_pad() returns a copy of the *input* padded to size specified by *pad_size* with value *pad_value*. If *pad_size* is positive then the array is padded on the right, if it's negative then on the left. If the absolute value of *pad_size* is less than or equal to the length of the *input* then no padding takes place.

Example 1. array_pad() example

```
$input = array (12, 10, 9);

$result = array_pad ($input, 5, 0);
// result is array (12, 10, 9, 0, 0)

$result = array_pad ($input, -7, -1);
// result is array (-1, -1, -1, -1, 12, 10, 9)

$result = array_pad ($input, 2, "noop");
// not padded
```

array_pop

(PHP 4)

array_pop -- Pop the element off the end of array

Description

mixed **array_pop** (array array)

array_pop() pops and returns the last value of the *array*, shortening the *array* by one element. If *array* is empty (or is not an array), **NULL** will be returned.

Example 1. array_pop() example

```
$stack = array ("orange", "banana", "apple", "raspberry");
$fruit = array_pop ($stack);
```

After this, `$stack` will have only 3 elements:

```
Array
(
    [0] => orange
    [1] => banana
    [2] => apple
)
```

and `rasberry` will be assigned to `$fruit`.

Warning
This function may return Boolean **FALSE**, but may also return a non-Boolean value which evaluates to **FALSE**, such as `0` or `""`. Please read the section on Booleans for more information. Use the === operator for testing the return value of this function.

See also **array_push()**, **array_shift()**, and **array_unshift()**.

array_push

(PHP 4)

array_push -- Push one or more elements onto the end of array

Description

int **array_push** (array array, mixed var [, mixed ...])

array_push() treats *array* as a stack, and pushes the passed variables onto the end of *array*. The length of *array* increases by the number of variables pushed. Has the same effect as:

```
$array[] = $var;
```

repeated for each *var*.

Returns the new number of elements in the array.

Example 1. array_push() example

```
$stack = array ("orange", "banana");
array_push ($stack, "apple", "raspberry");
```

This example would result in `$stack` having the following elements:

```
Array
(
    [0] => orange
    [1] => banana
    [2] => apple
    [3] => raspberry
)
```

See also **array_pop()**, **array_shift()**, and **array_unshift()**.

array_rand

(PHP 4)

array_rand -- Pick one or more random entries out of an array

Description

mixed **array_rand** (array input [, int num_req])

array_rand() is rather useful when you want to pick one or more random entries out of an array. It takes an *input* array and an optional argument *num_req* which specifies how many entries you want to pick - if not specified, it defaults to 1.

If you are picking only one entry, **array_rand()** returns the key for a random entry. Otherwise, it returns an array of keys for the random entries. This is done so that you can pick random keys as well as values out of the array.

Don't forget to call **srand()** to seed the random number generator.

Example 1. array_rand() example

```
srand ((float) microtime() * 10000000);
$input = array ("Neo", "Morpheus", "Trinity", "Cypher", "Tank");
$rand_keys = array_rand ($input, 2);
print $input[$rand_keys[0]]."\n";
print $input[$rand_keys[1]]."\n";
```

array_reduce

(PHP 4 >= 4.0.5)

array_reduce -- Iteratively reduce the array to a single value using a callback function

Description

mixed **array_reduce** (array input, callback function [, int initial])

array_reduce() applies iteratively the *function* function to the elements of the array *input*, so as to reduce the array to a single value. If the optional *initial* is available, it will be used at the beginning of the process, or as a final result in case the array is empty.

Example 1. array_reduce() example

```
function rsum($v, $w) {
    $v += $w;
    return $v;
}

function rmul($v, $w) {
    $v *= $w;
    return $v;
}

$a = array(1, 2, 3, 4, 5);
$x = array();
$b = array_reduce($a, "rsum");
$c = array_reduce($a, "rmul", 10);
$d = array_reduce($x, "rsum", 1);
```

This will result in `$b` containing `15`, `$c` containing `1200` (= 1*2*3*4*5*10), and `$d` containing `1`.

See also **array_filter()** and **array_map()**.

array_reverse

(PHP 4)

array_reverse -- Return an array with elements in reverse order

Description

array **array_reverse** (array array [, bool preserve_keys])

array_reverse() takes input *array* and returns a new array with the order of the elements reversed, preserving the keys if *preserve_keys* is **TRUE**.

Example 1. array_reverse() example

```
$input  = array ("php", 4.0, array ("green", "red"));
$result = array_reverse ($input);
$result_keyed = array_reverse ($input, TRUE);
```

This makes both `$result` and `$result_keyed` have the same elements, but note the difference between the keys. The printout of `$result` and `$result_keyed` will be:

```
Array
(
    [0] => Array
        (
            [0] => green
            [1] => red
        )

    [1] => 4
    [2] => php
)
Array
(
    [2] => Array
        (
            [0] => green
            [1] => red
        )

    [1] => 4
    [0] => php
)
```

Note: The second parameter was added in PHP 4.0.3.

array_search

(PHP 4 >= 4.0.5)

array_search -- Searches the array for a given value and returns the corresponding key if successful

Description

mixed **array_search** (mixed needle, array haystack [, bool strict])

Searches *haystack* for *needle* and returns the key if it is found in the array, **FALSE** otherwise.

Note: Prior to PHP 4.2.0, **array_search()** returns **NULL** on failure instead of **FALSE**.

If the optional third parameter *strict* is set to **TRUE** then the **array_search()** will also check the types of the *needle* in the *haystack*.

Warning

This function may return Boolean **FALSE**, but may also return a non-Boolean value which evaluates to **FALSE**, such as `0` or `""`. Please read the section on Booleans for more information. Use the === operator for testing the return value of this function.

See also **array_keys()** and **in_array()**.

array_shift

(PHP 4)

array_shift -- Shift an element off the beginning of array

Description

mixed **array_shift** (array array)

array_shift() shifts the first value of the *array* off and returns it, shortening the *array* by one element and moving everything down. All numerical array keys will be modified to start counting from zero while literal keys won't be touched. If *array* is empty (or is not an array), **NULL** will be returned.

Example 1. array_shift() example

```
$stack = array ("orange", "banana", "apple", "raspberry");
$fruit = array_shift ($stack);
```

This would result in `$stack` having 3 elements left:

```
Array
(
    [0] => banana
    [1] => apple
    [2] => raspberry
)
```

and `orange` will be assigned to `$fruit`.

See also **array_unshift()**, **array_push()**, and **array_pop()**.

array_slice

(PHP 4)

array_slice -- Extract a slice of the array

Description

array **array_slice** (array array, int offset [, int length])

array_slice() returns the sequence of elements from the array *array* as specified by the *offset* and *length* parameters.

If *offset* is positive, the sequence will start at that offset in the *array*. If *offset* is negative, the sequence will start that far from the end of the *array*.

If *length* is given and is positive, then the sequence will have that many elements in it. If *length* is given and is negative then the sequence will stop that many elements from the end of the array. If it is omitted, then the sequence will have everything from *offset* up until the end of the *array*.

Note that **array_slice()** will ignore array keys, and will calculate offsets and lengths based on the actual positions of elements within the array.

Example 1. array_slice() examples

```
$input = array ("a", "b", "c", "d", "e");

$output = array_slice ($input, 2);      // returns "c", "d", and "e"
$output = array_slice ($input, 2, -1);  // returns "c", "d"
$output = array_slice ($input, -2, 1);  // returns "d"
$output = array_slice ($input, 0, 3);   // returns "a", "b", and "c"
```

See also **array_splice()**.

array_splice

(PHP 4)

array_splice -- Remove a portion of the array and replace it with something else

Description

array **array_splice** (array input, int offset [, int length [, array replacement]])

array_splice() removes the elements designated by *offset* and *length* from the *input* array, and replaces them with the elements of the *replacement* array, if supplied. It returns an array containing the extracted elements.

If *offset* is positive then the start of removed portion is at that offset from the beginning of the *input* array. If *offset* is negative then it starts that far from the end of the *input* array.

If *length* is omitted, removes everything from *offset* to the end of the array. If *length* is specified and is positive, then that many elements will be removed. If *length* is specified and is negative then the end of the removed portion will be that many elements from the end of the array. Tip: to remove everything from *offset* to the end of the array when *replacement* is also specified, use `count($input)` for *length*.

If *replacement* array is specified, then the removed elements are replaced with elements from this array. If *offset* and *length* are such that nothing is removed, then the elements from the *replacement* array are inserted in the place specified by the *offset*. Tip: if the replacement is just one element it is not necessary to put `array()` around it, unless the element is an array itself.

The following equivalences hold:

```
array_push ($input, $x, $y)     array_splice ($input, count ($input), 0,
                                             array ($x, $y))
array_pop ($input)              array_splice ($input, -1)
array_shift ($input)            array_splice ($input, 0, 1)
array_unshift ($input, $x, $y)  array_splice ($input, 0, 0, array ($x, $y))
$input[$x] = $y                 array_splice ($input, $x, 1, $y)
```

Returns the array consisting of removed elements.

Example 1. array_splice() examples

```
$input = array ("red", "green", "blue", "yellow");
array_splice ($input, 2);
// $input is now array ("red", "green")

$input = array ("red", "green", "blue", "yellow");
array_splice ($input, 1, -1);
// $input is now array ("red", "yellow")

$input = array ("red", "green", "blue", "yellow");
array_splice ($input, 1, count($input), "orange");
// $input is now array ("red", "orange")

$input = array ("red", "green", "blue", "yellow");
array_splice ($input, -1, 1, array("black", "maroon"));
// $input is now array ("red", "green",
//          "blue", "black", "maroon")
```

See also **array_slice()**.

array_sum

(PHP 4 >= 4.0.4)

array_sum -- Calculate the sum of values in an array.

Description

mixed **array_sum** (array array)

array_sum() returns the sum of values in an array as an integer or float.

Example 1. array_sum() examples

```
$a = array(2, 4, 6, 8);
echo "sum(a) = ".array_sum($a)."\n";

$b = array("a"=>1.2,"b"=>2.3,"c"=>3.4);
echo "sum(b) = ".array_sum($b)."\n";
```

The printout of the program above will be:

```
sum(a) = 20
sum(b) = 6.9
```

> **Note:** PHP versions prior to 4.0.6 modified the passed array itself and converted strings to numbers (which most of the time converted them to zero, depending on their value).

array_unique

(PHP 4 >= 4.0.1)

array_unique -- Removes duplicate values from an array

Description

array **array_unique** (array array)

array_unique() takes input *array* and returns a new array without duplicate values.

Note that keys are preserved. **array_unique()** sorts the values treated as string at first, then will keep the first key encountered for every value, and ignore all following keys. It does not mean that the key of the first related value from the unsorted *array* will be kept.

> **Note:** Two elements are considered equal if and only if `(string) $elem1 === (string) $elem2`. In words: when the string representation is the same.
>
> The first element will be used.

Warning
This was broken in PHP 4.0.4!

Example 1. array_unique() example

```
$input = array ("a" => "green", "red", "b" => "green", "blue", "red");
$result = array_unique ($input);
print_r($result);
```

This will output:

```
Array
(
    [b] => green
    [1] => blue
    [2] => red
)
```

Example 2. array_unique() and types

```
$input = array (4,"4","3",4,3,"3");
$result = array_unique ($input);
var_dump($result);
```

The printout of the program above will be (PHP 4.0.6):

```
array(2) {
  [3]=>
  int(4)
  [4]=>
  int(3)
}
```

array_unshift

(PHP 4)

array_unshift -- Prepend one or more elements to the beginning of array

Description

int **array_unshift** (array array, mixed var [, mixed ...])

array_unshift() prepends passed elements to the front of the *array*. Note that the list of elements is prepended as a whole, so that the prepended elements stay in the same order. All numerical array keys will be modified to start counting from zero while literal keys won't be touched.

Returns the new number of elements in the *array*.

Example 1. array_unshift() example

```
$queue = array ("orange", "banana");
array_unshift ($queue, "apple", "raspberry");
```

This would result in $queue having the following elements:

```
Array
(
    [0] => apple
    [1] => raspberry
    [2] => orange
    [3] => banana
)
```

See also **array_shift()**, **array_push()**, and **array_pop()**.

array_values

(PHP 4)

array_values -- Return all the values of an array

Description

array **array_values** (array input)

array_values() returns all the values from the *input* array.

Example 1. array_values() example

```
$array = array ("size" => "XL", "color" => "gold");
print_r(array_values ($array));
```

This will output:

```
Array
(
    [0] => XL
    [1] => gold
)
```

Note: This function was added to PHP 4, below is an implementation for those still using PHP 3.

Example 2. Implementation of array_values() for PHP 3 users

```
function array_values ($arr) {
    $t = array();
    while (list($k, $v) = each ($arr)) {
        $t[] = $v;
    }
    return $t;
```

```
}
```

See also **array_keys()**.

array_walk

(PHP 3>= 3.0.3, PHP 4)

array_walk -- Apply a user function to every member of an array

Description

int **array_walk** (array array, callback function [, mixed userdata])

Applies the user-defined function *function* to each element of the *array* array. Typically, *function* takes on two parameters. The *array* parameter's value being the first, and the key/index second. If the optional *userdata* parameter is supplied, it will be passed as the third parameter to the callback *function*.

If function *function* requires more parameters than given to it, an error of level E_WARNING will be generated each time **array_walk()** calls *function*. These warnings may be suppressed by prepending the PHP error operator @ to the **array_walk()** call, or by using **error_reporting()**.

> **Note:** If *function* needs to be working with the actual values of the array, specify the first parameter of *function* as a reference. Then, any changes made to those elements will be made in the original array itself.
>
> **Note:** Passing the key and userdata to *function* was added in 4.0.0

array_walk() is not affected by the internal array pointer of *array*. **array_walk()** will walk through the entire array regardless of pointer position. To reset the pointer, use **reset()**. In PHP 3, **array_walk()** resets the pointer.

Users may not change the array itself from the callback function. e.g. Add/delete elements, unset elements, etc. If the array that **array_walk()** is applied to is changed, the behavior of this function is undefined, and unpredictable.

Example 1. array_walk() example

```
<?php
$fruits = array ("d"=>"lemon", "a"=>"orange", "b"=>"banana", "c"=>"apple");

function test_alter (&$item1, $key, $prefix) {
    $item1 = "$prefix: $item1";
}

function test_print ($item2, $key) {
    echo "$key. $item2<br>\n";
}

echo "Before ...:\n";
array_walk ($fruits, 'test_print');

array_walk ($fruits, 'test_alter', 'fruit');
echo "... and after:\n";

array_walk ($fruits, 'test_print');
?>
```

The printout of the program above will be:

```
Before ...:
d. lemon
a. orange
b. banana
c. apple
... and after:
d. fruit: lemon
a. fruit: orange
b. fruit: banana
c. fruit: apple
```

See also **list()**, foreach, **each()**, and **call_user_func_array()**.

array

(PHP 3, PHP 4)

array -- Create an array

Description

array **array** ([mixed ...])

Returns an array of the parameters. The parameters can be given an index with the => operator.

> **Note: array()** is a language construct used to represent literal arrays, and not a regular function.

Syntax "index => values", separated by commas, define index and values. index may be of type string or numeric. When index is omitted, a integer index is automatically generated, starting at 0. If index is an integer, next generated index will be the biggest integer index + 1. Note that when two identical index are defined, the last overwrite the first.

The following example demonstrates how to create a two-dimensional array, how to specify keys for associative arrays, and how to skip-and-continue numeric indices in normal arrays.

Example 1. array() example

```
$fruits = array (
    "fruits"  => array ("a"=>"orange", "b"=>"banana", "c"=>"apple"),
    "numbers" => array (1, 2, 3, 4, 5, 6),
    "holes"   => array ("first", 5 => "second", "third")
);
```

Example 2. Automatic index with array()

```
$array = array( 1, 1, 1, 1,  1, 8=>1,  4=>1, 19, 3=>13);
print_r($array);
```

will display :

```
Array
(
    [0] => 1
    [1] => 1
    [2] => 1
    [3] => 13
    [4] => 1
    [8] => 1
    [9] => 19
)
```

Note that index '3' is defined twice, and keep its final value of 13. Index 4 is defined after index 8, and next generated index (value 19) is 9, since biggest index was 8.

This example creates a 1-based array.

Example 3. 1-based index with array()

```
$firstquarter  = array(1 => 'January', 'February', 'March');
print_r($firstquarter);
```

will display :

```
Array
(
    [1] => 'January'
    [2] => 'February'
    [3] => 'March'
)
```

See also **array_pad()**, **list()**, and **range()**.

arsort

(PHP 3, PHP 4)

arsort -- Sort an array in reverse order and maintain index association

Description

void **arsort** (array array [, int sort_flags])

This function sorts an array such that array indices maintain their correlation with the array elements they are associated with. This is used mainly when sorting associative arrays where the actual element order is significant.

Example 1. arsort() example

```
$fruits = array ("d"=>"lemon", "a"=>"orange", "b"=>"banana", "c"=>"apple");
arsort ($fruits);
reset ($fruits);
while (list ($key, $val) = each ($fruits)) {
    echo "$key = $val\n";
}
```

This example would display:

```
a = orange
d = lemon
b = banana
c = apple
```

The fruits have been sorted in reverse alphabetical order, and the index associated with each element has been maintained.

You may modify the behavior of the sort using the optional parameter *sort_flags*, for details see **sort()**.

See also **asort()**, **rsort()**, **ksort()**, and **sort()**.

asort

(PHP 3, PHP 4)

asort -- Sort an array and maintain index association

Description

void **asort** (array array [, int sort_flags])

This function sorts an array such that array indices maintain their correlation with the array elements they are associated with. This is used mainly when sorting associative arrays where the actual element order is significant.

Example 1. asort() example

```
$fruits = array ("d"=>"lemon", "a"=>"orange", "b"=>"banana", "c"=>"apple");
asort ($fruits);
reset ($fruits);
while (list ($key, $val) = each ($fruits)) {
    echo "$key = $val\n";
}
```

This example would display:

```
c = apple
b = banana
d = lemon
a = orange
```

The fruits have been sorted in alphabetical order, and the index associated with each element has been maintained.

You may modify the behavior of the sort using the optional parameter *sort_flags*, for details see **sort()**.

See also **arsort()**, **rsort()**, **ksort()**, and **sort()**.

compact

(PHP 4)

compact -- Create array containing variables and their values

Description

array **compact** (mixed varname [, mixed ...])

compact() takes a variable number of parameters. Each parameter can be either a string containing the name of the variable, or an array of variable names. The array can contain other arrays of variable names inside it; **compact()** handles it recursively.

For each of these, **compact()** looks for a variable with that name in the current symbol table and adds it to the output array such that the variable name becomes the key and the contents of the variable become the value for that key. In short, it does the opposite of **extract()**. It returns the output array with all the variables added to it.

Any strings that are not set will simply be skipped.

Example 1. compact() example

```
$city = "San Francisco";
$state = "CA";
$event = "SIGGRAPH";

$location_vars = array ("city", "state");

$result = compact ("event", "nothing_here", $location_vars);
```

After this, `$result` will be:

```
Array
(
    [event] => SIGGRAPH
    [city] => San Francisco
    [state] => CA
)
```

See also **extract()**.

count

(PHP 3, PHP 4)

count -- Count elements in a variable

Description

int **count** (mixed var)

Returns the number of elements in *var*, which is typically an **array** (since anything else will have one element).

If *var* is not an array, `1` will be returned (exception: `count(NULL)` equals `0`).

Warning
count() may return 0 for a variable that isn't set, but it may also return 0 for a variable that has been initialized with an empty array. Use **isset()** to test if a variable is set.

Please see the Arrays section of the manual for a detailed explanation of how arrays are implemented and used in PHP.

Example 1. count() example

```
$a[0] = 1;
$a[1] = 3;
$a[2] = 5;
$result = count ($a);
// $result == 3

$b[0] = 7;
$b[5] = 9;
$b[10] = 11;
$result = count ($b);
// $result == 3;
```

Note: The **sizeof()** function is an alias for **count()**.

See also **is_array()**, **isset()**, and **strlen()**.

current

(PHP 3, PHP 4)

current -- Return the current element in an array

Description

mixed **current** (array array)

Every array has an internal pointer to its "current" element, which is initialized to the first element inserted into the array.

The **current()** function simply returns the array element that's currently being pointed by the internal pointer. It does not move the pointer in any way. If the internal pointer points beyond the end of the elements list, **current()** returns **FALSE**.

Warning
If the array contains empty elements (0 or "", the empty string) then this function will return **FALSE** for these elements as well. This makes it impossible to determine if you are really at the end of the list in such an array using **current()**. To properly traverse an array that may contain empty elements, use the **each()** function.

See also **end()**, **next()**, **prev()**, and **reset()**.

each

(PHP 3, PHP 4)

each -- Return the current key and value pair from an array and advance the array cursor

Description

array **each** (array array)

Returns the current key and value pair from the array *array* and advances the array cursor. This pair is returned in a four-element array, with the keys *0*, *1*, *key*, and *value*. Elements *0* and *key* contain the key name of the array element, and *1* and *value* contain the data.

If the internal pointer for the array points past the end of the array contents, **each()** returns **FALSE**.

Example 1. each() examples

```
$foo = array ("bob", "fred", "jussi", "jouni", "egon", "marliese");
$bar = each ($foo);
```

`$bar` now contains the following key/value pairs:

- 0 => 0
- 1 => 'bob'
- key => 0
- value => 'bob'

```
$foo = array ("Robert" => "Bob", "Seppo" => "Sepi");
$bar = each ($foo);
```

`$bar` now contains the following key/value pairs:

- 0 => 'Robert'
- 1 => 'Bob'
- key => 'Robert'
- value => 'Bob'

each() is typically used in conjunction with **list()** to traverse an array; for instance, `$_POST`:

Example 2. Traversing $_POST with each()

```
echo "Values submitted via POST method:<br />\n";
reset ($_POST);
while (list ($key, $val) = each ($_POST)) {
    echo "$key => $val<br />\n";
}
```

After **each()** has executed, the array cursor will be left on the next element of the array, or on the last element if it hits the end of the array. You have to use **reset()** if you want to traverse the array again using each.

See also **key()**, **list()**, **current()**, **reset()**, **next()**, **prev()**, and foreach.

end

(PHP 3, PHP 4)

end -- Set the internal pointer of an array to its last element

Description

mixed **end** (array array)

end() advances *array*'s internal pointer to the last element, and returns that element.

Example 1. A simple end() example

```
<?php
  $fruits = array('apple','banana','cranberry');
  print end($fruits); // cranberry
?>
```

See also **current()**, **each()**, **next()**, and **reset()**.

extract

(PHP 3>= 3.0.7, PHP 4)

extract -- Import variables into the current symbol table from an array

Description

int **extract** (array var_array [, int extract_type [, string prefix]])

This function is used to import variables from an array into the current symbol table. It takes an associative array *var_array* and treats keys as variable names and values as variable values. For each key/value pair it will create a variable in the current symbol table, subject to *extract_type* and *prefix* parameters.

> **Note:** Beginning with version 4.0.5, this function returns the number of variables extracted.
>
> **Note:** EXTR_IF_EXISTS and EXTR_PREFIX_IF_EXISTS was introduced in version 4.2.0.
>
> **Note:** EXTR_REFS was introduced in version 4.3.0.

extract() checks each key to see whether it has a valid variable name. It also checks for collisions with existing variables in the symbol table. The way invalid/numeric keys and collisions are treated is determined by the *extract_type*. It can be one of the following values:

EXTR_OVERWRITE

> If there is a collision, overwrite the existing variable.

EXTR_SKIP

If there is a collision, don't overwrite the existing variable.

EXTR_PREFIX_SAME

If there is a collision, prefix the variable name with *prefix*.

EXTR_PREFIX_ALL

Prefix all variable names with *prefix*. Beginning with PHP 4.0.5, this includes numeric variables as well.

EXTR_PREFIX_INVALID

Only prefix invalid/numeric variable names with *prefix*. This flag was added in PHP 4.0.5.

EXTR_IF_EXISTS

Only overwrite the variable if it already exists in the current symbol table, otherwise do nothing. This is useful for defining a list of valid variables and then extracting only those variables you have defined out of $_REQUEST, for example. This flag was added in PHP 4.2.0.

EXTR_PREFIX_IF_EXISTS

Only create prefixed variable names if the non-prefixed version of the same variable exists in the current symbol table. This flag was added in PHP 4.2.0.

EXTR_REFS

Extracts variables as references. This effectively means that the values of the imported variables are still referencing the values of the *var_array* parameter. You can use this flag on its own or combine it with any other flag by OR'ing the *extract_type*. This flag was added in PHP 4.3.0.

If *extract_type* is not specified, it is assumed to be EXTR_OVERWRITE.

Note that *prefix* is only required if *extract_type* is EXTR_PREFIX_SAME, EXTR_PREFIX_ALL, EXTR_PREFIX_INVALID or EXTR_PREFIX_IF_EXISTS. If the prefixed result is not a valid variable name, it is not imported into the symbol table.

extract() returns the number of variables successfully imported into the symbol table.

A possible use for **extract()** is to import into the symbol table variables contained in an associative array returned by **wddx_deserialize()**.

Example 1. extract() example

```
<?php

/* Suppose that $var_array is an array returned from
   wddx_deserialize */

$size = "large";
$var_array = array ("color" => "blue",
                    "size"  => "medium",
                    "shape" => "sphere");
extract ($var_array, EXTR_PREFIX_SAME, "wddx");

print "$color, $size, $shape, $wddx_size\n";

?>
```

The above example will produce:

```
blue, large, sphere, medium
```

The `$size` wasn't overwritten, because we specified EXTR_PREFIX_SAME, which resulted in `$wddx_size` being created. If EXTR_SKIP was specified, then $wddx_size wouldn't even have been created. EXTR_OVERWRITE would have caused `$size` to have value "medium", and EXTR_PREFIX_ALL would result in new variables being named `$wddx_color`, `$wddx_size`, and `$wddx_shape`.

You must use an associative array, a numerically indexed array will not produce results unless you use EXTR_PREFIX_ALL or EXTR_PREFIX_INVALID.

See also **compact()**.

in_array

(PHP 4)

in_array -- Return **TRUE** if a value exists in an array

Description

bool **in_array** (mixed needle, array haystack [, bool strict])

Searches *haystack* for *needle* and returns **TRUE** if it is found in the array, **FALSE** otherwise.

If the third parameter *strict* is set to **TRUE** then the **in_array()** function will also check the types of the *needle* in the *haystack*.

> **Note:** If *needle* is a string, the comparison is done in a case-sensitive manner.

> **Note:** In PHP versions before 4.2.0 *needle* was not allowed to be an array.

Example 1. in_array() example

```
$os = array ("Mac", "NT", "Irix", "Linux");
if (in_array ("Irix", $os)) {
    print "Got Irix";
}
if (in_array ("mac", $os)) {
    print "Got mac";
}
```

The second condition fails because **in_array()** is case-sensitive, so the program above will display:

```
Got Irix
```

Example 2. in_array() with strict example

```
<?php
$a = array('1.10', 12.4, 1.13);

if (in_array('12.4', $a, TRUE))
    echo "'12.4' found with strict check\n";
if (in_array(1.13, $a, TRUE))
    echo "1.13 found with strict check\n";
?>
```

This will display:

```
1.13 found with strict check
```

Example 3. in_array() with an array as needle

```
<?php
$a = array(array('p', 'h'), array('p', 'r'), 'o');

if (in_array(array ('p', 'h'), $a))
    echo "'ph' was found\n";
if (in_array(array ('f', 'i'), $a))
    echo "'fi' was found\n";
if (in_array('o', $a))
    echo "'o' was found\n";
?>

// This will output:

'ph' was found
'o' was found
```

See also **array_search()**.

key

(PHP 3, PHP 4)

key -- Fetch a key from an associative array

Description

mixed **key** (array array)

key() returns the index element of the current array position.

See also **current()** and **next()**.

krsort

(PHP 3>= 3.0.13, PHP 4)

krsort -- Sort an array by key in reverse order

Description

int **krsort** (array array [, int sort_flags])

Sorts an array by key in reverse order, maintaining key to data correlations. This is useful mainly for associative arrays.

Example 1. krsort() example

```
$fruits = array ("d"=>"lemon", "a"=>"orange", "b"=>"banana", "c"=>"apple");
krsort ($fruits);
reset ($fruits);
while (list ($key, $val) = each ($fruits)) {
    echo "$key = $val\n";
}
```

This example would display:

```
d = lemon
c = apple
b = banana
a = orange
```

You may modify the behavior of the sort using the optional parameter *sort_flags*, for details see **sort()**.

See also **asort()**, **arsort()**, **ksort()**, **sort()**, **natsort()**, and **rsort()**.

ksort

(PHP 3, PHP 4)

ksort -- Sort an array by key

Description

int **ksort** (array array [, int sort_flags])

Sorts an array by key, maintaining key to data correlations. This is useful mainly for associative arrays.

Example 1. ksort() example

```
$fruits = array ("d"=>"lemon", "a"=>"orange", "b"=>"banana", "c"=>"apple");
ksort ($fruits);
reset ($fruits);
while (list ($key, $val) = each ($fruits)) {
    echo "$key = $val\n";
}
```

This example would display:

```
a = orange
b = banana
c = apple
d = lemon
```

You may modify the behavior of the sort using the optional parameter *sort_flags*, for details see **sort()**.

See also **asort()**, **arsort()**, **krsort()**, **uksort()**, **sort()**, **natsort()**, and **rsort()**.

> **Note:** The second parameter was added in PHP 4.

list

(PHP 3, PHP 4)

list -- Assign variables as if they were an array

Description

void **list** (mixed ...)

Like **array()**, this is not really a function, but a language construct. **list()** is used to assign a list of variables in one operation.

> **Note: list()** only works on numerical arrays and assumes the numerical indices start at 0.

Example 1. list() examples

```
<?php

$info = array('coffee', 'brown', 'caffeine');

// Listing all the variables
list($drink, $color, $power) = $info;
print "$drink is $color and $power makes it special.\n";

// Listing some of them
list($drink, , $power) = $info;
print "$drink has $power.\n";

// Or let's skip to only the third one
list( , , $power) = $info;
print "I need $power!\n";

?>
```

Example 2. An example use of list()

```
<table>
 <tr>
  <th>Employee name</th>
  <th>Salary</th>
 </tr>

<?php

$result = mysql_query ("SELECT id, name, salary FROM employees",$conn);
while (list ($id, $name, $salary) = mysql_fetch_row ($result)) {
    print (" <tr>\n".
           "  <td><a href=\"info.php?id=$id\">$name</a></td>\n".
           "  <td>$salary</td>\n".
           " </tr>\n");
}

?>

</table>
```

Warning
list() assigns the values starting with the right-most parameter. If you are using plain variables, you don't have to worry about this. But if you are using arrays with indices you usually expect the order of the indices in the array the same you wrote in the **list()** from left to right; which it isn't. It's assigned in the reverse order.

Example 3. Using list() with array indices

```
<?php

$info = array('coffee', 'brown', 'caffeine');

list($a[0], $a[1], $a[2]) = $info;

var_dump($a);
```

```
?>
```

Gives the following output (note the order of the elements compared in which order they were written in the **list()** syntax):

```
array(3) {
  [2]=>
  string(8) "caffeine"
  [1]=>
  string(5) "brown"
  [0]=>
  string(6) "coffee"
}
```

See also **each()**, **array()** and **extract()**.

natcasesort

(PHP 4)

natcasesort -- Sort an array using a case insensitive "natural order" algorithm

Description

void **natcasesort** (array array)

This function implements a sort algorithm that orders alphanumeric strings in the way a human being would. This is described as a "natural ordering".

natcasesort() is a case insensitive version of **natsort()**. See **natsort()** for an example of the difference between this algorithm and the regular computer string sorting algorithms.

For more information see: Martin Pool's Natural Order String Comparison page.

See also **sort()**, **natsort()**, **strnatcmp()**, and **strnatcasecmp()**.

natsort

(PHP 4)

natsort -- Sort an array using a "natural order" algorithm

Description

void **natsort** (array array)

This function implements a sort algorithm that orders alphanumeric strings in the way a human being would. This is described as a "natural ordering". An example of the difference between this algorithm and the regular computer string sorting algorithms (used in **sort()**) can be seen below:

Example 1. natsort() example

```
<?php
$array1 = $array2 = array ("img12.png", "img10.png", "img2.png", "img1.png");

sort($array1);
echo "Standard sorting\n";
print_r($array1);

natsort($array2);
echo "\nNatural order sorting\n";
print_r($array2);
?>
```

The code above will generate the following output:

```
Standard sorting
Array
(
    [0] => img1.png
    [1] => img10.png
    [2] => img12.png
```

```
    [3] => img2.png
)

Natural order sorting
Array
(
    [3] => img1.png
    [2] => img2.png
    [1] => img10.png
    [0] => img12.png
)
```

For more information see: Martin Pool's Natural Order String Comparison page.

Note: If you're wanting to maintain index/value associations, consider using `usort($arr, 'strnatcmp')`.

See also **natcasesort()**, **strnatcmp()**, and **strnatcasecmp()**.

next

(PHP 3, PHP 4)

next -- Advance the internal array pointer of an array

Description

mixed **next** (array array)

Returns the array element in the next place that's pointed by the internal array pointer, or **FALSE** if there are no more elements.

next() behaves like **current()**, with one difference. It advances the internal array pointer one place forward before returning the element. That means it returns the next array element and advances the internal array pointer by one. If advancing the internal array pointer results in going beyond the end of the element list, **next()** returns **FALSE**.

Warning
If the array contains empty elements, or elements that have a key value of 0 then this function will return **FALSE** for these elements as well. To properly traverse an array which may contain empty elements or elements with key values of 0 see the **each()** function.

See also **current()**, **end()**, **prev()**, and **reset()**.

pos

(PHP 3, PHP 4)

pos -- Get the current element from an array

Description

mixed **pos** (array array)

This is an alias for **current()**.

See also **end()**, **next()**, **prev()**, and **reset()**.

prev

(PHP 3, PHP 4)

prev -- Rewind the internal array pointer

Description

mixed **prev** (array array)

Returns the array element in the previous place that's pointed by the internal array pointer, or **FALSE** if there are no more

elements.

Warning
If the array contains empty elements then this function will return FALSE for these elements as well. To properly traverse an array which may contain empty elements see the **each()** function.

prev() behaves just like **next()**, except it rewinds the internal array pointer one place instead of advancing it.

See also **current()**, **end()**, **next()**, and **reset()**.

range

(PHP 3>= 3.0.8, PHP 4)

range -- Create an array containing a range of elements

Description

array **range** (mixed low, mixed high [, int step])

range() returns an array of elements from *low* to *high*, inclusive. If low > high, the sequence will be from high to low.

> **New parameter:** The optional *step* parameter was added in 5.0.0.

If a *step* value is given, it will be used as the increment between elements in the sequence. *step* should be given as a positive number. If not specified, *step* will default to 1.

Example 1. range() examples

```
<?php
// array(0,1,2,3,4,5,6,7,8,9)
foreach(range(0, 9) as $number) {
    echo $number;
}

// The step parameter was introduced in 5.0.0
// array(0,10,20,30,40,50,60,70,80,90,100)
foreach(range(0, 100, 10) as $number) {
    echo $number;
}

// Use of characters introduced in 4.1.0
// array('a','b','c','d','e','f','g','h','i');
foreach(range('a', 'i') as $letter) {
    echo $letter;
}
// array('c','b','a');
foreach(range('c', 'a') as $letter) {
    echo $letter;
}
?>
```

> **Note:** Prior to version 4.1.0 the **range()** function only generated incrementing integer arrays. Support for character sequences and decrementing arrays was added in 4.1.0.

See also **shuffle()** and foreach.

reset

(PHP 3, PHP 4)

reset -- Set the internal pointer of an array to its first element

Description

mixed **reset** (array array)

reset() rewinds *array*'s internal pointer to the first element.

reset() returns the value of the first array element.

See also **current()**, **each()**, **next()**, and **prev()**.

rsort

(PHP 3, PHP 4)

rsort -- Sort an array in reverse order

Description

void **rsort** (array array [, int sort_flags])

This function sorts an array in reverse order (highest to lowest).

Example 1. rsort() example

```
$fruits = array ("lemon", "orange", "banana", "apple");
rsort ($fruits);
reset ($fruits);
while (list ($key, $val) = each ($fruits)) {
    echo "$key = $val\n";
}
```

This example would display:

```
0 = orange
1 = lemon
2 = banana
3 = apple
```

The fruits have been sorted in reverse alphabetical order.

You may modify the behavior of the sort using the optional parameter *sort_flags*, for details see **sort()**.

See also **arsort()**, **asort()**, **ksort()**, **sort()**, and **usort()**.

shuffle

(PHP 3>= 3.0.8, PHP 4)

shuffle -- Shuffle an array

Description

void **shuffle** (array array)

This function shuffles (randomizes the order of the elements in) an array. You must use **srand()** to seed this function.

Example 1. shuffle() example

```
$numbers = range (1,20);
srand ((float)microtime()*1000000);
shuffle ($numbers);
while (list (, $number) = each ($numbers)) {
    echo "$number ";
}
```

See also **arsort()**, **asort()**, **ksort()**, **rsort()**, **sort()**, and **usort()**.

sizeof

(PHP 3, PHP 4)

sizeof -- Get the number of elements in variable

Description

int **sizeof** (mixed var)

The **sizeof()** function is an alias for **count()**.

See also **count()**.

sort

(PHP 3, PHP 4)

sort -- Sort an array

Description

void **sort** (array array [, int sort_flags])

This function sorts an array. Elements will be arranged from lowest to highest when this function has completed.

Example 1. sort() example

```
<?php

$fruits = array ("lemon", "orange", "banana", "apple");
sort ($fruits);
reset ($fruits);
while (list ($key, $val) = each ($fruits)) {
    echo "fruits[".$key."] = ".$val."\n";
}

?>
```

This example would display:

```
fruits[0] = apple
fruits[1] = banana
fruits[2] = lemon
fruits[3] = orange
```

The fruits have been sorted in alphabetical order.

The optional second parameter *sort_flags* may be used to modify the sorting behavior using these values:

Sorting type flags:

- SORT_REGULAR - compare items normally
- SORT_NUMERIC - compare items numerically
- SORT_STRING - compare items as strings

See also **arsort()**, **asort()**, **ksort()**, **natsort()**, **natcasesort()**, **rsort()**, **usort()**, **array_multisort()**, and **uksort()**.

> **Note:** The second parameter was added in PHP 4.

uasort

(PHP 3>= 3.0.4, PHP 4)

uasort -- Sort an array with a user-defined comparison function and maintain index association

Description

void **uasort** (array array, callback cmp_function)

This function sorts an array such that array indices maintain their correlation with the array elements they are associated with. This is used mainly when sorting associative arrays where the actual element order is significant. The comparison function is user-defined.

> **Note:** Please see **usort()** and **uksort()** for examples of user-defined comparison functions.

See also usort(), uksort(), sort(), asort(), arsort(), ksort(), and rsort().

uksort

(PHP 3>= 3.0.4, PHP 4)

uksort -- Sort an array by keys using a user-defined comparison function

Description

void **uksort** (array array, callback cmp_function)

This function will sort the keys of an array using a user-supplied comparison function. If the array you wish to sort needs to be sorted by some non-trivial criteria, you should use this function.

Example 1. uksort() example

```
function cmp ($a, $b) {
    if ($a == $b) return 0;
    return ($a > $b) ? -1 : 1;
}

$a = array (4 => "four", 3 => "three", 20 => "twenty", 10 => "ten");

uksort ($a, "cmp");

while (list ($key, $value) = each ($a)) {
    echo "$key: $value\n";
}
```

This example would display:

```
20: twenty
10: ten
4: four
3: three
```

See also usort(), uasort(), sort(), asort(), arsort(), ksort(), natsort(), and rsort().

usort

(PHP 3>= 3.0.3, PHP 4)

usort -- Sort an array by values using a user-defined comparison function

Description

void **usort** (array array, callback cmp_function)

This function will sort an array by its values using a user-supplied comparison function. If the array you wish to sort needs to be sorted by some non-trivial criteria, you should use this function.

The comparison function must return an integer less than, equal to, or greater than zero if the first argument is considered to be respectively less than, equal to, or greater than the second.

> **Note:** If two members compare as equal, their order in the sorted array is undefined. Up to PHP 4.0.6 the user defined functions would keep the original order for those elements, but with the new sort algorithm intruduced with 4.1.0 this is no longer the case as there is no solution to do so in an efficient way.

Example 1. usort() example

```
function cmp ($a, $b) {
    if ($a == $b) return 0;
    return ($a > $b) ? -1 : 1;
}

$a = array (3, 2, 5, 6, 1);

usort ($a, "cmp");
```

```
while (list ($key, $value) = each ($a)) {
    echo "$key: $value\n";
}
```

This example would display:

```
0: 6
1: 5
2: 3
3: 2
4: 1
```

Note: Obviously in this trivial case the rsort() function would be more appropriate.

Example 2. usort() example using multi-dimensional array

```
function cmp ($a, $b) {
    return strcmp($a["fruit"], $b["fruit"]);
}

$fruits[0]["fruit"] = "lemons";
$fruits[1]["fruit"] = "apples";
$fruits[2]["fruit"] = "grapes";

usort($fruits, "cmp");

while (list ($key, $value) = each ($fruits)) {
    echo "\$fruits[$key]: " . $value["fruit"] . "\n";
}
```

When sorting a multi-dimensional array, $a and $b contain references to the first index of the array.

This example would display:

```
$fruits[0]: apples
$fruits[1]: grapes
$fruits[2]: lemons
```

Example 3. usort() example using a member function of an object

```
class TestObj {
    var $name;

    function TestObj($name)
    {
        $this->name = $name;
    }

    /* This is the static comparing function: */
    function cmp_obj($a, $b)
    {
        $al = strtolower($a->name);
        $bl = strtolower($b->name);
        if ($al == $bl) return 0;
        return ($al > $bl) ? +1 : -1;
    }
}

$a[] = new TestObj("c");
$a[] = new TestObj("b");
$a[] = new TestObj("d");

uasort($a, array ("TestObj", "cmp_obj"));

foreach ($a as $item) {
    print $item->name."\n";
}
```

This example would display:

```
b
c
d
```

See also uasort(), uksort(), sort(), asort(), arsort(),ksort(), natsort(), and rsort().

III. Aspell functions [deprecated]

Introduction

The **aspell()** functions allows you to check the spelling on a word and offer suggestions.

Requirements

aspell works only with very old (up to .27.* or so) versions of aspell library. Neither this module, nor those versions of aspell library are supported any longer. If you want to use spell-checking capabilities in PHP, use pspell instead. It uses pspell library and works with newer versions of aspell.

Installation

You need the aspell library, available from: http://aspell.sourceforge.net/.

See Also

See also pspell.

Table of Contents

aspell_check_raw

(PHP 3>= 3.0.7, PHP 4 <= 4.2.3)

aspell_check_raw -- Check a word without changing its case or trying to trim it [deprecated]

Description

bool **aspell_check_raw** (int dictionary_link, string word)

aspell_check_raw() checks the spelling of a word, without changing its case or trying to trim it in any way and returns **TRUE** if the spelling is correct, **FALSE** if not.

Example 1. aspell_check_raw()

```
$aspell_link = aspell_new("english");

if (aspell_check_raw($aspell_link, "test")) {
    echo "This is a valid spelling";
} else {
    echo "Sorry, wrong spelling";
}
```

aspell_check

(PHP 3>= 3.0.7, PHP 4 <= 4.2.3)

aspell_check -- Check a word [deprecated]

Description

bool **aspell_check** (int dictionary_link, string word)

aspell_check() checks the spelling of a word and returns **TRUE** if the spelling is correct, **FALSE** if not.

Example 1. aspell_check()

```
$aspell_link = aspell_new("english");

if (aspell_check($aspell_link, "testt")) {
    echo "This is a valid spelling";
} else {
    echo "Sorry, wrong spelling";
}
```

aspell_new

(PHP 3>= 3.0.7, PHP 4 <= 4.2.3)

aspell_new -- Load a new dictionary [deprecated]

Description

int **aspell_new** (string master [, string personal])

aspell_new() opens up a new dictionary and returns the dictionary link identifier for use in other aspell functions. Returns **FALSE** on error.

Example 1. aspell_new()

```
$aspell_link = aspell_new("english");
```

aspell_suggest

(PHP 3>= 3.0.7, PHP 4 <= 4.2.3)

aspell_suggest -- Suggest spellings of a word [deprecated]

Description

array **aspell_suggest** (int dictionary_link, string word)

aspell_suggest() returns an array of possible spellings for the given word.

Example 1. aspell_suggest()

```
$aspell_link = aspell_new("english");

if (!aspell_check($aspell_link, "test")) {
    $suggestions = aspell_suggest($aspell_link, "test");

    foreach ($suggestions as $suggestion) {
        echo "Possible spelling: $suggestion<br>\n";
    }
}
```

IV. BCMath Arbitrary Precision Mathematics Functions

Introduction

For arbitrary precision mathematics PHP offers the Binary Calculator which supports numbers of any size and precision, represented as strings.

Requirements

Since PHP 4.0.4, libbcmath is bundled with PHP. You don't need any external libraries for this extension.

Installation

In PHP 4, these functions are only available if PHP was configured with `--enable-bcmath`. In PHP 3, these functions are only available if PHP was NOT configured with `--disable-bcmath`.

The windows version of PHP has built in support for this extension. You do not need to load any additional extension in order to use these functions.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. BC math configuration options

Name	Default	Changeable
bcmath.scale	0	PHP_INI_ALL

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Here is a short explanation of the configuration directives.

`bcmath.scale` **integer**

Number of decimal digits for all bcmath functions.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Table of Contents

bcadd

(PHP 3, PHP 4)

bcadd -- Add two arbitrary precision numbers

Description

string **bcadd** (string left_operand, string right_operand [, int scale])

Adds the *left_operand* to the *right_operand* and returns the sum in a string. The optional *scale* parameter is used to set the number of digits after the decimal place in the result.

See also **bcsub()**.

bccomp

(PHP 3, PHP 4)

bccomp -- Compare two arbitrary precision numbers

Description

int **bccomp** (string left_operand, string right_operand [, int scale])

Compares the *left_operand* to the *right_operand* and returns the result as an integer. The optional *scale* parameter is used to set the number of digits after the decimal place which will be used in the comparison. The return value is 0 if the two operands are equal. If the *left_operand* is larger than the *right_operand* the return value is +1 and if the *left_operand* is less than the *right_operand* the return value is -1.

bcdiv

(PHP 3, PHP 4)

bcdiv -- Divide two arbitrary precision numbers

Description

string **bcdiv** (string left_operand, string right_operand [, int scale])

Divides the *left_operand* by the *right_operand* and returns the result. The optional *scale* sets the number of digits after the decimal place in the result.

See also **bcmul()**.

bcmod

(PHP 3, PHP 4)

bcmod -- Get modulus of an arbitrary precision number

Description

string **bcmod** (string left_operand, string modulus)

Get the modulus of the *left_operand* using *modulus*.

See also **bcdiv()**.

bcmul

(PHP 3, PHP 4)

bcmul -- Multiply two arbitrary precision number

Description

string **bcmul** (string left_operand, string right_operand [, int scale])

Multiply the *left_operand* by the *right_operand* and returns the result. The optional *scale* sets the number of digits after the

decimal place in the result.

See also **bcdiv()**.

bcpow

(PHP 3, PHP 4)

bcpow -- Raise an arbitrary precision number to another

Description

string **bcpow** (string x, string y [, int scale])

Raise *x* to the power *y*. The optional *scale* can be used to set the number of digits after the decimal place in the result.

See also **bcpowmod()**, and **bcsqrt()**.

bcpowmod

(PHP 5 CVS only)

bcpowmod -- Raise an arbitrary precision number to another, reduced by a specified modulus.

Description

string **bcpowmod** (string x, string y, string modulus [, int scale])

Use the fast-exponentiation method to raise *x* to the power *y* with respect to the modulus *modulus*. The optional *scale* can be used to set the number of digits after the decimal place in the result.

The following two statements are functionally identical. The **bcpowmod()** version however, executes in less time and can accept larger parameters.

```
<?php
$a = bcpowmod($x,$y,$mod);

$b = bcmod(bcpow($x,$y),$mod);

/* $a and $b are equal to each other. */
?>
```

> **Note:** Because this method uses the modulus operation, non-natural numbers may give unexpected results. A natural number is any positive non-zero integer.

See also **bcpow()**, and **bcmod()**.

bcscale

(PHP 3, PHP 4)

bcscale -- Set default scale parameter for all bc math functions

Description

string **bcscale** (int scale)

This function sets the default scale parameter for all subsequent bc math functions that do not explicitly specify a scale parameter.

bcsqrt

(PHP 3, PHP 4)

bcsqrt -- Get the square root of an arbitrary precision number

Description

string **bcsqrt** (string operand [, int scale])

Return the square root of the *operand*. The optional *scale* parameter sets the number of digits after the decimal place in the result.

See also **bcpow()**.

bcsub

(PHP 3, PHP 4)

bcsub -- Subtract one arbitrary precision number from another

Description

string **bcsub** (string left_operand, string right_operand [, int scale])

Subtracts the *right_operand* from the *left_operand* and returns the result in a string. The optional *scale* parameter is used to set the number of digits after the decimal place in the result.

See also **bcadd()**.

V. Bzip2 Compression Functions

Introduction

The bzip2 functions are used to transparently read and write bzip2 (.bz2) compressed files.

Requirements

This module uses the functions of the bzip2 library by Julian Seward. This module requires bzip2/libbzip2 version >= 1.0.x.

Installation

Bzip2 support in PHP is not enabled by default. You will need to use the `--with-bz2[=DIR]` configuration option when compiling PHP to enable bzip2 support.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension defines one resource type: a file pointer identifying the bz2-file to work on.

Predefined Constants

This extension has no constants defined.

Examples

This example opens a temporary file and writes a test string to it, then prints out the contents of the file.

Example 1. Small bzip2 Example

```
<?php

$filename = "/tmp/testfile.bz2";
$str = "This is a test string.\n";

// open file for writing
$bz = bzopen($filename, "w");

// write string to file
bzwrite($bz, $str);

// close file
bzclose($bz);

// open file for reading
$bz = bzopen($filename, "r");

// read 10 characters
print bzread($bz, 10);

// output until end of the file (or the next 1024 char) and close it.
print bzread($bz);

bzclose($bz);

?>
```

Table of Contents

bzclose

(PHP 4 >= 4.0.4)

bzclose -- Close a bzip2 file pointer

Description

int **bzclose** (resource bz)

Closes the bzip2 file referenced by the pointer *bz*.

Returns **TRUE** on success or **FALSE** on failure.

The file pointer must be valid, and must point to a file successfully opened by **bzopen()**.

See also **bzopen()**.

bzcompress

(PHP 4 >= 4.0.4)

bzcompress -- Compress a string into bzip2 encoded data

Description

string **bzcompress** (string source [, int blocksize [, int workfactor]])

bzcompress() compresses the *source* string and returns it as bzip2 encoded data.

The optional parameter *blocksize* specifies the blocksize used during compression and should be a number from 1 to 9 with 9 giving the best compression, but using more resources to do so. *blocksize* defaults to 4.

The optional parameter *workfactor* controls how the compression phase behaves when presented with worst case, highly repetitive, input data. The value can be between 0 and 250 with 0 being a special case and 30 being the default value. Regardless of the *workfactor*, the generated output is the same.

Example 1. bzcompress() Example

```
<?php
$str = "sample data";
$bzstr = bzcompress($str, 9);
print( $bzstr );
?>
```

See also **bzdecompress()**.

bzdecompress

(PHP 4 >= 4.0.4)

bzdecompress -- Decompresses bzip2 encoded data

Description

string **bzdecompress** (string source [, int small])

bzdecompress() decompresses the *source* string containing bzip2 encoded data and returns it. If the optional parameter *small* is **TRUE**, an alternative decompression algorithm will be used which uses less memory (the maximum memory requirement drops to around 2300K) but works at roughly half the speed. See the bzip2 documentation for more information about this feature.

Example 1. bzdecompress()

```
<?php
$start_str = "This is not an honest face?";
$bzstr = bzcompress($start_str);

print( "Compressed String: " );
print( $bzstr );
print( "\n<br>\n" );

$str = bzdecompress($bzstr);
print( "Decompressed String: " );
print( $str );
print( "\n<br>\n" );
?>
```

See also **bzcompress()**.

bzerrno

(PHP 4 >= 4.0.4)

bzerrno -- Returns a bzip2 error number

Description

int **bzerrno** (resource bz)

Returns the error number of any bzip2 error returned by the file pointer *bz*.

See also **bzerror()** and **bzerrstr()**.

bzerror

(PHP 4 >= 4.0.4)

bzerror -- Returns the bzip2 error number and error string in an array

Description

array **bzerror** (resource bz)

Returns the error number and error string, in an associative array, of any bzip2 error returned by the file pointer *bz*.

Example 1. bzerror() Example

```
<?php
$error = bzerror($bz);

echo $error["errno"];
echo $error["errstr"];
?>
```

See also **bzerrno()** and **bzerrstr()**.

bzerrstr

(PHP 4 >= 4.0.4)

bzerrstr -- Returns a bzip2 error string

Description

string **bzerrstr** (resource bz)

Returns the error string of any bzip2 error returned by the file pointer *bz*.

See also **bzerrno()** and **bzerror()**.

bzflush

(PHP 4 >= 4.0.4)

bzflush -- Force a write of all buffered data

Description

int **bzflush** (resource bz)

Forces a write of all buffered bzip2 data for the file pointer *bz*.

Returns **TRUE** on success or **FALSE** on failure.

See also **bzread()** and **bzwrite()**.

bzopen

(PHP 4 >= 4.0.4)

bzopen -- Open a bzip2 compressed file

Description

resource **bzopen** (string filename, string mode)

Opens a bzip2 (.bz2) file for reading or writing. *filename* is the name of the file to open. *mode* is similar to the **fopen()** function (`r' for read, `w' for write, etc.).

If the open fails, the function returns **FALSE**, otherwise it returns a pointer to the newly opened file.

Example 1. bzopen() Example

```
<?php
$bz = bzopen("/tmp/foo.bz2", "r");
$decompressed_file = bzread($bz, filesize("/tmp/foo.bz2"));
bzclose($bz);

print( "The contents of /tmp/foo.bz2 are: " );
print( "\n<br>\n" );
print( $decompressed_file );
?>
```

See also **bzclose()**.

bzread

(PHP 4 >= 4.0.4)

bzread -- Binary safe bzip2 file read

Description

string **bzread** (resource bz [, int length])

bzread() reads up to *length* bytes from the bzip2 file pointer referenced by *bz*. Reading stops when *length* (uncompressed) bytes have been read or EOF is reached, whichever comes first. If the optional parameter *length* is not specified, **bzread()** will read 1024 (uncompressed) bytes at a time.

Example 1. bzread() Example

```
<?php
$bz = bzopen("/tmp/foo.bz2", "r");
$str = bzread($bz, 2048);
print( $str );
?>
```

See also **bzwrite()** and **bzopen()**.

bzwrite

(PHP 4 >= 4.0.4)

bzwrite -- Binary safe bzip2 file write

Description

int **bzwrite** (resource bz, string data [, int length])

bzwrite() writes the contents of the string *data* to the bzip2 file stream pointed to by *bz*. If the optional *length* argument is given, writing will stop after length (uncompressed) bytes have been written or the end of string is reached, whichever comes first.

Example 1. bzwrite() Example

```
<?php
$str = "uncompressed data";
$bz = bzopen("/tmp/foo.bz2", "w");
bzwrite($bz, $str, strlen($str));
bzclose($bz);
?>
```

See also **bzread()** and **bzopen()**.

VI. Calendar functions

Introduction

The calendar extension presents a series of functions to simplify converting between different calendar formats. The intermediary or standard it is based on is the Julian Day Count. The Julian Day Count is a count of days starting from January 1st, 4713 B.C. To convert between calendar systems, you must first convert to Julian Day Count, then to the calendar system of your choice. Julian Day Count is very different from the Julian Calendar! For more information on Julian Day Count, visit http://serendipity.magnet.ch/hermetic/cal_stud/jdn.htm. For more information on calendar systems visit http://genealogy.org/~scottlee/cal-overview.html. Excerpts from this page are included in these instructions, and are in quotes.

Installation

To get these functions to work, you have to compile PHP with `--enable-calendar`.

The windows version of PHP has built in support for this extension. You do not need to load any additional extension in order to use these functions.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

CAL_GREGORIAN (integer)

CAL_JULIAN (integer)

CAL_JEWISH (integer)

CAL_FRENCH (integer)

CAL_NUM_CALS (integer)

CAL_DOW_DAYNO (integer)

CAL_DOW_SHORT (integer)

CAL_DOW_LONG (integer)

CAL_MONTH_GREGORIAN_SHORT (integer)

CAL_MONTH_GREGORIAN_LONG (integer)

CAL_MONTH_JULIAN_SHORT (integer)

CAL_MONTH_JULIAN_LONG (integer)

CAL_MONTH_JEWISH (integer)

CAL_MONTH_FRENCH (integer)

The following constants are available since PHP 4.3.0 :

CAL_EASTER_DEFAULT (integer)

CAL_EASTER_ROMAN (integer)

CAL_EASTER_ALWAYS_GREGORIAN (integer)

CAL_EASTER_ALWAYS_JULIAN (integer)

Table of Contents

cal_days_in_month

(PHP 4 >= 4.1.0)

cal_days_in_month -- Return the number of days in a month for a given year and calendar

Description

int **cal_days_in_month** (int calendar, int month, int year)

This function will return the number of days in the *month* of *year* for the specified *calendar*.

See also **jdtounix()**.

cal_from_jd

(PHP 4 >= 4.1.0)

cal_from_jd -- Converts from Julian Day Count to a supported calendar and return extended information

Description

array **cal_from_jd** (int jd, int calendar)

Warning
This function is currently not documented; only the argument list is available.

cal_info

(PHP 4 >= 4.1.0)

cal_info -- Returns information about a particular calendar

Description

array **cal_info** (int calendar)

Warning
This function is currently not documented; only the argument list is available.

cal_to_jd

(PHP 4 >= 4.1.0)

cal_to_jd -- Converts from a supported calendar to Julian Day Count

Description

int **cal_to_jd** (int calendar, int month, int day, int year)

Warning
This function is currently not documented; only the argument list is available.

easter_date

(PHP 3>= 3.0.9, PHP 4)

easter_date -- Get UNIX timestamp for midnight on Easter of a given year

Description

int **easter_date** ([int year])

Returns the UNIX timestamp corresponding to midnight on Easter of the given year.

Since PHP 4.3.0, the *year* parameter is optional and defaults to the current year according to the localtime if ommited.

Warning: This function will generate a warning if the year is outside of the range for UNIX timestamps (i.e. before 1970 or after 2037).

Example 1. easter_date() example

```
echo date ("M-d-Y", easter_date(1999));        /* "Apr-04-1999" */
echo date ("M-d-Y", easter_date(2000));        /* "Apr-23-2000" */
echo date ("M-d-Y", easter_date(2001));        /* "Apr-15-2001" */
```

The date of Easter Day was defined by the Council of Nicaea in AD325 as the Sunday after the first full moon which falls on or after the Spring Equinox. The Equinox is assumed to always fall on 21st March, so the calculation reduces to determining the date of the full moon and the date of the following Sunday. The algorithm used here was introduced around the year 532 by Dionysius Exiguus. Under the Julian Calendar (for years before 1753) a simple 19-year cycle is used to track the phases of the Moon. Under the Gregorian Calendar (for years after 1753 - devised by Clavius and Lilius, and introduced by Pope Gregory XIII in October 1582, and into Britain and its then colonies in September 1752) two correction factors are added to make the cycle more accurate.

(The code is based on a C program by Simon Kershaw, <webmaster@ely.anglican.org>)

See **easter_days()** for calculating Easter before 1970 or after 2037.

easter_days

(PHP 3>= 3.0.9, PHP 4)

easter_days -- Get number of days after March 21 on which Easter falls for a given year

Description

int **easter_days** ([int year [, int method]])

Returns the number of days after March 21 on which Easter falls for a given year. If no year is specified, the current year is assumed.

Since PHP 4.3.0, the *year* parameter is optional and defaults to the current year according to the localtime if ommited.

The *method* parameter was also introduced in PHP 4.3.0 and allows to calculate easter dates based on the Gregorian calendar during the years 1582 - 1752 when set to **CAL_EASTER_ROMAN**. See the calendar constants for more valid constants.

This function can be used instead of **easter_date()** to calculate Easter for years which fall outside the range of UNIX timestamps (i.e. before 1970 or after 2037).

Example 1. easter_days() example

```
echo easter_days (1999);        /* 14, i.e. April 4   */
echo easter_days (1492);        /* 32, i.e. April 22  */
echo easter_days (1913);        /*  2, i.e. March 23  */
```

The date of Easter Day was defined by the Council of Nicaea in AD325 as the Sunday after the first full moon which falls on or after the Spring Equinox. The Equinox is assumed to always fall on 21st March, so the calculation reduces to determining the date of the full moon and the date of the following Sunday. The algorithm used here was introduced around the year 532 by Dionysius Exiguus. Under the Julian Calendar (for years before 1753) a simple 19-year cycle is used to track the phases of the Moon. Under the Gregorian Calendar (for years after 1753 - devised by Clavius and Lilius, and introduced by Pope Gregory XIII in October 1582, and into Britain and its then colonies in September 1752) two correction factors are added to make the cycle more accurate.

(The code is based on a C program by Simon Kershaw, <webmaster@ely.anglican.org>)

See also **easter_date()**.

FrenchToJD

(PHP 3, PHP 4)

FrenchToJD -- Converts a date from the French Republican Calendar to a Julian Day Count

Description

int **frenchtojd** (int month, int day, int year)

Converts a date from the French Republican Calendar to a Julian Day Count.

These routines only convert dates in years 1 through 14 (Gregorian dates 22 September 1792 through 22 September 1806). This more than covers the period when the calendar was in use.

GregorianToJD

(PHP 3, PHP 4)

GregorianToJD -- Converts a Gregorian date to Julian Day Count

Description

int **gregoriantojd** (int month, int day, int year)

Valid Range for Gregorian Calendar 4714 B.C. to 9999 A.D.

Although this function can handle dates all the way back to 4714 B.C., such use may not be meaningful. The Gregorian calendar was not instituted until October 15, 1582 (or October 5, 1582 in the Julian calendar). Some countries did not accept it until much later. For example, Britain converted in 1752, The USSR in 1918 and Greece in 1923. Most European countries used the Julian calendar prior to the Gregorian.

Example 1. Calendar functions

```
<?php
$jd = GregorianToJD (10,11,1970);
echo "$jd\n";
$gregorian = JDToGregorian ($jd);
echo "$gregorian\n";
?>
```

JDDayOfWeek

(PHP 3, PHP 4)

JDDayOfWeek -- Returns the day of the week

Description

mixed **jddayofweek** (int julianday, int mode)

Returns the day of the week. Can return a string or an integer depending on the mode.

Table 1. Calendar week modes

Mode	Meaning
0	Returns the day number as an int (0=sunday, 1=monday, etc)
1	Returns string containing the day of week (english-gregorian)
2	Returns a string containing the abbreviated day of week (english-gregorian)

JDMonthName

(PHP 3, PHP 4)

JDMonthName -- Returns a month name

Description

string **jdmonthname** (int julianday, int mode)

Returns a string containing a month name. *mode* tells this function which calendar to convert the Julian Day Count to, and what type of month names are to be returned.

Table 1. Calendar modes

Mode	Meaning	Values
0	Gregorian - abbreviated	Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec
1	Gregorian	January, February, March, April, May, June, July, August, September, October, November, December
2	Julian - abbreviated	Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec
3	Julian	January, February, March, April, May, June, July, August, September, October, November, December
4	Jewish	Tishri, Heshvan, Kislev, Tevet, Shevat, AdarI, AdarII, Nisan, Iyyar, Sivan, Tammuz, Av, Elul
5	French Republican	Vendemiaire, Brumaire, Frimaire, Nivose, Pluviose, Ventose, Germinal, Floreal, Prairial, Messidor, Thermidor, Fructidor, Extra

JDToFrench

(PHP 3, PHP 4)

JDToFrench -- Converts a Julian Day Count to the French Republican Calendar

Description

string **jdtofrench** (int juliandaycount)

Converts a Julian Day Count to the French Republican Calendar.

JDToGregorian

(PHP 3, PHP 4)

JDToGregorian -- Converts Julian Day Count to Gregorian date

Description

string **jdtogregorian** (int julianday)

Converts Julian Day Count to a string containing the Gregorian date in the format of "month/day/year".

JDToJewish

(PHP 3, PHP 4)

JDToJewish -- Converts a Julian Day Count to the Jewish Calendar

Description

string **jdtojewish** (int julianday)

Converts a Julian Day Count the the Jewish Calendar.

JDToJulian

(PHP 3, PHP 4)

JDToJulian -- Converts a Julian Day Count to a Julian Calendar Date

Description

string **jdtojulian** (int julianday)

Converts Julian Day Count to a string containing the Julian Calendar Date in the format of "month/day/year".

jdtounix

(PHP 4)

jdtounix -- Convert Julian Day to UNIX timestamp

Description

int **jdtounix** (int jday)

This function will return a UNIX timestamp corresponding to the Julian Day given in *jday* or **FALSE** if *jday* is not inside the UNIX epoch (Gregorian years between 1970 and 2037 or 2440588 <= *jday* <= 2465342). The time returned is localtime (and not GMT).

See also **unixtojd()**.

JewishToJD

(PHP 3, PHP 4)

JewishToJD -- Converts a date in the Jewish Calendar to Julian Day Count

Description

int **jewishtojd** (int month, int day, int year)

Although this function can handle dates all the way back to the year 1 (3761 B.C.), such use may not be meaningful. The Jewish calendar has been in use for several thousand years, but in the early days there was no formula to determine the start of a month. A new month was started when the new moon was first observed.

JulianToJD

(PHP 3, PHP 4)

JulianToJD -- Converts a Julian Calendar date to Julian Day Count

Description

int **juliantojd** (int month, int day, int year)

Valid Range for Julian Calendar 4713 B.C. to 9999 A.D.

Although this function can handle dates all the way back to 4713 B.C., such use may not be meaningful. The calendar was created in 46 B.C., but the details did not stabilize until at least 8 A.D., and perhaps as late at the 4th century. Also, the beginning of a year varied from one culture to another - not all accepted January as the first month.

Caution
Remember, the current calendar system being used worldwide is the Gregorian calendar. **gregoriantojd()** can be used to convert such dates to their Julian Day count.

unixtojd

(PHP 4)

unixtojd -- Convert UNIX timestamp to Julian Day

Description

int **unixtojd** ([int timestamp])

Return the Julian Day for a UNIX *timestamp* (seconds since 1.1.1970), or for the current day if no *timestamp* is given.

See also **jdtounix()**.

VII. CCVS API Functions

Introduction

These functions interface the CCVS API, allowing you to work directly with CCVS from your PHP scripts. CCVS is RedHat's solution to the "middle-man" in credit card processing. It lets you directly address the credit card clearing houses via your *nix box and a modem. Using the CCVS module for PHP, you can process credit cards directly through CCVS via your PHP Scripts. The

following references will outline the process.

> **Note:** CCVS has been discontinued by Red Hat and there are no plans to issue further keys or support contracts. Those looking for a replacement can consider MCVE by Main Street Softworks as a potential replacement. It is similar in design and has documented PHP support!

Installation

To enable CCVS Support in PHP, first verify your CCVS installation directory. You will then need to configure PHP with the `--with-ccvs` option. If you use this option without specifying the path to your CCVS installation, PHP will attempt to look in the default CCVS Install location (/usr/local/ccvs). If CCVS is in a non-standard location, run configure with: `--with-ccvs=$ccvs_path`, where $ccvs_path is the path to your CCVS installation. Please note that CCVS support requires that $ccvs_path/lib and $ccvs_path/include exist, and include cv_api.h under the include directory and libccvs.a under the lib directory.

Additionally, a ccvsd process will need to be running for the configurations you intend to use in your PHP scripts. You will also need to make sure the PHP Processes are running under the same user as your CCVS was installed as (e.g. if you installed CCVS as user 'ccvs', your PHP processes must run as 'ccvs' as well.)

See Also

Additional information about CCVS can be found at http://www.redhat.com/products/ccvs. RedHat maintains slightly outdated but still useful documentation at http://www.redhat.com/products/ccvs/support/CCVS3.3docs/ProgPHP.html.

Table of Contents

ccvs_add

(4.0.2 - 4.2.3 only)

ccvs_add -- Add data to a transaction

Description

string **ccvs_add** (string session, string invoice, string argtype, string argval)

Warning
This function is currently not documented; only the argument list is available.

ccvs_auth

(4.0.2 - 4.2.3 only)

ccvs_auth -- Perform credit authorization test on a transaction

Description

string **ccvs_auth** (string session, string invoice)

Warning
This function is currently not documented; only the argument list is available.

ccvs_command

(4.0.2 - 4.2.3 only)

ccvs_command -- Performs a command which is peculiar to a single protocol, and thus is not available in the general CCVS API

Description

string **ccvs_command** (string session, string type, string argval)

Warning
This function is currently not documented; only the argument list is available.

ccvs_count

(4.0.2 - 4.2.3 only)

ccvs_count -- Find out how many transactions of a given type are stored in the system

Description

int **ccvs_count** (string session, string type)

Warning
This function is currently not documented; only the argument list is available.

ccvs_delete

(4.0.2 - 4.2.3 only)

ccvs_delete -- Delete a transaction

Description

string **ccvs_delete** (string session, string invoice)

Warning
This function is currently not documented; only the argument list is available.

ccvs_done

(4.0.2 - 4.2.3 only)

ccvs_done -- Terminate CCVS engine and do cleanup work

Description

string **ccvs_done** (string sess)

Warning
This function is currently not documented; only the argument list is available.

ccvs_init

(4.0.2 - 4.2.3 only)

ccvs_init -- Initialize CCVS for use

Description

string **ccvs_init** (string name)

Warning
This function is currently not documented; only the argument list is available.

ccvs_lookup

(4.0.2 - 4.2.3 only)

ccvs_lookup -- Look up an item of a particular type in the database #

Description

string **ccvs_lookup** (string session, string invoice, int inum)

Warning
This function is currently not documented; only the argument list is available.

ccvs_new

(4.0.2 - 4.2.3 only)

ccvs_new -- Create a new, blank transaction

Description

string **ccvs_new** (string session, string invoice)

Warning
This function is currently not documented; only the argument list is available.

ccvs_report

(4.0.2 - 4.2.3 only)

ccvs_report -- Return the status of the background communication process

Description

string **ccvs_report** (string session, string type)

Warning
This function is currently not documented; only the argument list is available.

ccvs_return

(4.0.2 - 4.2.3 only)

ccvs_return -- Transfer funds from the merchant to the credit card holder

Description

string **ccvs_return** (string session, string invoice)

Warning
This function is currently not documented; only the argument list is available.

ccvs_reverse

(4.0.2 - 4.2.3 only)

ccvs_reverse -- Perform a full reversal on an already-processed authorization

Description

string **ccvs_reverse** (string session, string invoice)

Warning
This function is currently not documented; only the argument list is available.

ccvs_sale

(4.0.2 - 4.2.3 only)

ccvs_sale -- Transfer funds from the credit card holder to the merchant

Description

string **ccvs_sale** (string session, string invoice)

Warning
This function is currently not documented; only the argument list is available.

ccvs_status

(4.0.2 - 4.2.3 only)

ccvs_status -- Check the status of an invoice

Description

string **ccvs_status** (string session, string invoice)

Warning
This function is currently not documented; only the argument list is available.

ccvs_textvalue

(4.0.2 - 4.2.3 only)

ccvs_textvalue -- Get text return value for previous function call

Description

string **ccvs_textvalue** (string session)

Warning
This function is currently not documented; only the argument list is available.

ccvs_void

(4.0.2 - 4.2.3 only)

ccvs_void -- Perform a full reversal on a completed transaction

Description

string **ccvs_void** (string session, string invoice)

Warning
This function is currently not documented; only the argument list is available.

VIII. COM support functions for Windows

Introduction

COM is a technology which allows the reuse of code written in any language (by any language) using a standard calling convention and hiding behind APIs the implementation details such as what machine the Component is stored on and the executable which houses it. It can be thought of as a super Remote Procedure Call (RPC) mechanism with some basic object roots. It separates implementation from interface.

COM encourages versioning, separation of implementation from interface and hiding the implementation details such as executable location and the language it was written in.

Requirements

COM functions are only available on the Windows version of PHP.

Installation

There is no installation needed to use these functions; they are part of the PHP core.

The windows version of PHP has built in support for this extension. You do not need to load any additional extension in order to use these functions.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. Com configuration options

Name	Default	Changeable
com.allow_dcom	"0"	PHP_INI_SYSTEM
com.autoregister_typelib	"0"	PHP_INI_SYSTEM
com.autoregister_verbose	"0"	PHP_INI_SYSTEM
com.autoregister_casesensitive	"1"	PHP_INI_SYSTEM
com.typelib_file	""	PHP_INI_SYSTEM

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

CLSCTX_INPROC_SERVER (integer)

CLSCTX_INPROC_HANDLER (integer)

CLSCTX_LOCAL_SERVER (integer)

CLSCTX_REMOTE_SERVER (integer)

CLSCTX_SERVER (integer)

CLSCTX_ALL (integer)

VT_NULL (integer)

VT_EMPTY (integer)

VT_UI1 (integer)

VT_I2 (integer)

VT_I4 (integer)

VT_R4 (integer)

VT_R8 (integer)

VT_BOOL (integer)

VT_ERROR (integer)

VT_CY (integer)

VT_DATE (integer)

VT_BSTR (integer)

VT_DECIMAL (integer)

VT_UNKNOWN (integer)

VT_DISPATCH (integer)

VT_VARIANT (integer)

VT_I1 (integer)

VT_UI2 (integer)

VT_UI4 (integer)

VT_INT (integer)

VT_UINT (integer)

VT_ARRAY (integer)

VT_BYREF (integer)

CP_ACP (integer)

CP_MACCP (integer)

CP_OEMCP (integer)

CP_UTF7 (integer)

CP_UTF8 (integer)

CP_SYMBOL (integer)

CP_THREAD_ACP (integer)

See Also

For further information on COM read the COM specification or perhaps take a look at Don Box's Yet Another COM Library (YACL)

Table of Contents

COM

(no version information, might be only in CVS)

COM -- COM class

Synopsis

```
$obj = new COM("server.object")
```

Description

The COM class provides a framework to integrate (D)COM components into your php scripts.

Methods

string **COM::COM** (string module_name [, string server_name [, int codepage]])

COM class constructor. Parameters:

module_name

name or class-id of the requested component.

server_name

name of the DCOM server from which the component should be fetched. If **NULL**, `localhost` is assumed. To allow DCOM **com.allow_dcom** has to be set to **TRUE** in `php.ini`.

codepage

specifies the codepage that is used to convert php-strings to unicode-strings and vice versa. Possible values are **CP_ACP**, **CP_MACCP**, **CP_OEMCP**, **CP_SYMBOL**, **CP_THREAD_ACP**, **CP_UTF7** and **CP_UTF8**.

Example 1. COM example (1)

```
// starting word
$word = new COM("word.application") or die("Unable to instanciate Word");
print "Loaded Word, version {$word->Version}\n";

//bring it to front
$word->Visible = 1;

//open an empty document
$word->Documents->Add();

//do some weird stuff
$word->Selection->TypeText("This is a test...");
$word->Documents[1]->SaveAs("Useless test.doc");

//closing word
$word->Quit();

//free the object
$word->Release();
$word = null;
```

Example 2. COM example (2)

```
$conn = new COM("ADODB.Connection") or die("Cannot start ADO");
$conn->Open("Provider=SQLOLEDB; Data Source=localhost;
Initial Catalog=database; User ID=user; Password=password");

$rs = $conn->Execute("SELECT * FROM sometable");    // Recordset

$num_columns = $rs->Fields->Count();
echo $num_columns . "\n";

for ($i=0; $i < $num_columns; $i++)
{
    $fld[$i] = $rs->Fields($i);
}

$rowcount = 0;
while (!$rs->EOF)
{
    for ($i=0; $i < $num_columns; $i++)
    {
        echo $fld[$i]->value . "\t";
    }
    echo "\n";
    $rowcount++;            // increments rowcount
    $rs->MoveNext();
}

$rs->Close();
$conn->Close();

$rs->Release();
$conn->Release();

$rs = null;
$conn = null;
```

VARIANT

(no version information, might be only in CVS)

VARIANT -- VARIANT class

Synopsis

```
$vVar = new VARIANT($var)
```

Description

A simple container to wrap variables into VARIANT structures.

Methods

string **VARIANT::VARIANT** ([mixed value [, int type [, int codepage]]])

VARIANT class constructor. Parameters:

value

initial value. if omitted an VT_EMPTY object is created.

type

specifies the content type of the VARIANT object. Possible values are **VT_UI1**, **VT_UI2**, **VT_UI4**, **VT_I1**, **VT_I2**, **VT_I4**, **VT_R4**, **VT_R8**, **VT_INT**, **VT_UINT**, **VT_BOOL**, **VT_ERROR**, **VT_CY**, **VT_DATE**, **VT_BSTR**, **VT_DECIMAL**, **VT_UNKNOWN**, **VT_DISPATCH** and **VT_VARIANT**. These values are mutual exclusive, but they can be combined with **VT_BYREF** to specify being a value. If omitted, the type of *value* is used. Consult the msdn library for additional information.

codepage

specifies the codepage that is used to convert php-strings to unicode-strings and vice versa. Possible values are **CP_ACP**, **CP_MACCP**, **CP_OEMCP**, **CP_SYMBOL**, **CP_THREAD_ACP**, **CP_UTF7** and **CP_UTF8**.

com_addref

(4.1.0 - 4.3.0 only)

com_addref -- Increases the components reference counter.

Description

void **com_addref** (void)

Increases the components reference counter.

com_get

(PHP 3>= 3.0.3, 4.0.5 - 4.3.0 only)

com_get -- Gets the value of a COM Component's property

Description

mixed **com_get** (resource com_object, string property)

Returns the value of the *property* of the COM component referenced by *com_object*. Returns **FALSE** on error.

com_invoke

(PHP 3>= 3.0.3)

com_invoke -- Calls a COM component's method.

Description

mixed **com_invoke** (resource com_object, string function_name [, mixed function parameters, ...])

com_invoke() invokes a method of the COM component referenced by *com_object*. Returns **FALSE** on error, returns the *function_name*'s return value on success.

com_isenum

(4.1.0 - 4.3.0 only)

com_isenum -- Grabs an IEnumVariant

Description

void **com_isenum** (object com_module)

Warning
This function is currently not documented; only the argument list is available.

com_load_typelib

(4.1.0 - 4.3.0 only)

com_load_typelib -- Loads a Typelib

Description

void **com_load_typelib** (string typelib_name [, int case_insensitive])

Warning
This function is currently not documented; only the argument list is available.

com_load

(PHP 3>= 3.0.3)

com_load -- Creates a new reference to a COM component

Description

string **com_load** (string module name [, string server name [, int codepage]])

com_load() creates a new COM component and returns a reference to it. Returns **FALSE** on failure. Possible values for *codepage* are **CP_ACP**, **CP_MACCP**, **CP_OEMCP**, **CP_SYMBOL**, **CP_THREAD_ACP**, **CP_UTF7** and **CP_UTF8**.

com_propget

(PHP 3>= 3.0.3, 4.0.5 - 4.3.0 only)

com_propget -- Gets the value of a COM Component's property

Description

mixed **com_propget** (resource com_object, string property)

This function is an alias for **com_get()**.

com_propput

(PHP 3>= 3.0.3, 4.0.5 - 4.3.0 only)

com_propput -- Assigns a value to a COM component's property

Description

void **com_propput** (resource com_object, string property, mixed value)

This function is an alias for **com_set()**.

com_propset

(PHP 3>= 3.0.3, 4.0.5 - 4.3.0 only)

com_propset -- Assigns a value to a COM component's property

Description

void **com_propset** (resource com_object, string property, mixed value)

This function is an alias for **com_set()**.

com_release

(4.1.0 - 4.3.0 only)

com_release -- Decreases the components reference counter.

Description

void **com_release** (void)

Decreases the components reference counter.

com_set

(PHP 3>= 3.0.3, 4.0.5 - 4.3.0 only)

com_set -- Assigns a value to a COM component's property

Description

void **com_set** (resource com_object, string property, mixed value)

Sets the value of the *property* of the COM component referenced by *com_object*. Returns the newly set value if succeeded, **FALSE** on error.

IX. Class/Object Functions

Introduction

These functions allow you to obtain information about classes and instance objects. You can obtain the name of the class to which a object belongs, as well as its member properties and methods. Using these functions, you can find out not only the class membership of an object, but also its parentage (i.e. what class is the object class extending).

Requirements

No external libraries are needed to build this extension.

Installation

There is no installation needed to use these functions; they are part of the PHP core.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Examples

In this example, we first define a base class and an extension of the class. The base class describes a general vegetable, whether it is edible or not and what is its color. The subclass `Spinach` adds a method to cook it and another to find out if it is cooked.

Example 1. classes.inc

```
<?php

// base class with member properties and methods
class Vegetable {

    var $edible;
    var $color;

    function Vegetable( $edible, $color="green" ) {
        $this->edible = $edible;
        $this->color = $color;
    }

    function is_edible() {
        return $this->edible;
    }

    function what_color() {
        return $this->color;
    }

} // end of class Vegetable

// extends the base class
class Spinach extends Vegetable {

    var $cooked = false;

    function Spinach() {
        $this->Vegetable( true, "green" );
    }

    function cook_it() {
        $this->cooked = true;
    }

    function is_cooked() {
        return $this->cooked;
    }

} // end of class Spinach

?>
```

We then instantiate 2 objects from these classes and print out information about them, including their class parentage. We also define some utility functions, mainly to have a nice printout of the variables.

Example 2. test_script.php

```
<pre>
<?php

include "classes.inc";

// utility functions

function print_vars($obj) {
    $arr = get_object_vars($obj);
    while (list($prop, $val) = each($arr))
        echo "\t$prop = $val\n";
}

function print_methods($obj) {
    $arr = get_class_methods(get_class($obj));
    foreach ($arr as $method)
        echo "\tfunction $method()\n";
}

function class_parentage($obj, $class) {
    global $$obj;
    if (is_subclass_of($$obj, $class)) {
        echo "Object $obj belongs to class ".get_class($$obj);
        echo " a subclass of $class\n";
    } else {
        echo "Object $obj does not belong to a subclass of $class\n";
    }
}

// instantiate 2 objects

$veggie = new Vegetable(true,"blue");
$leafy = new Spinach();

// print out information about objects
echo "veggie: CLASS ".get_class($veggie)."\n";
echo "leafy: CLASS ".get_class($leafy);
echo ", PARENT ".get_parent_class($leafy)."\n";

// show veggie properties
echo "\nveggie: Properties\n";
print_vars($veggie);

// and leafy methods
echo "\nleafy: Methods\n";
print_methods($leafy);

echo "\nParentage:\n";
class_parentage("leafy", "Spinach");
class_parentage("leafy", "Vegetable");
?>
</pre>
```

One important thing to note in the example above is that the object `$leafy` is an instance of the class **Spinach** which is a subclass of **Vegetable**, therefore the last part of the script above will output:

```
[...]
Parentage:
Object leafy does not belong to a subclass of Spinach
Object leafy belongs to class spinach a subclass of Vegetable
```

Table of Contents

call_user_method_array

(PHP 4 >= 4.0.5)

call_user_method_array -- Call a user method given with an array of parameters [deprecated]

Description

mixed **call_user_method_array** (string method_name, object obj [, array paramarr])

Warning
The **call_user_method_array()** function is deprecated as of PHP 4.1.0, use the **call_user_func_array()** variety with the `array(&$obj, "method_name")` syntax instead.

Calls the method referred by *`method_name`* from the user defined *`obj`* object, using the parameters in *`paramarr`*.

See also: **call_user_func_array()**, **call_user_func()**, **call_user_method()**.

> **Note:** This function was added to the CVS code after release of PHP 4.0.4pl1

call_user_method

(PHP 3>= 3.0.3, PHP 4)

call_user_method -- Call a user method on an specific object [deprecated]

Description

mixed **call_user_method** (string method_name, object obj [, mixed parameter [, mixed ...]])

Warning
The **call_user_method()** function is deprecated as of PHP 4.1.0, use the **call_user_func()** variety with the `array(&$obj, "method_name")` syntax instead.

Calls the method referred by *`method_name`* from the user defined *`obj`* object. An example of usage is below, where we define a class, instantiate an object and use **call_user_method()** to call indirectly its `print_info` method.

```
<?php
class Country {
    var $NAME;
    var $TLD;

    function Country($name, $tld) {
        $this->NAME = $name;
        $this->TLD = $tld;
    }

    function print_info($prestr="") {
        echo $prestr."Country: ".$this->NAME."\n";
        echo $prestr."Top Level Domain: ".$this->TLD."\n";
    }
}

$cntry = new Country("Peru","pe");

echo "* Calling the object method directly\n";
$cntry->print_info();

echo "\n* Calling the same method indirectly\n";
call_user_method ("print_info", $cntry, "\t");
?>
```

See also **call_user_func_array()**, **call_user_func()**, and **call_user_method_array()**.

class_exists

(PHP 4)

class_exists -- Checks if the class has been defined

Description

bool **class_exists** (string class_name)

This function returns **TRUE** if the class given by *`class_name`* has been defined, **FALSE** otherwise.

See also **get_declared_classes()**.

get_class_methods

(PHP 4)

get_class_methods -- Returns an array of class methods' names

Description

array **get_class_methods** (mixed class_name)

This function returns an array of method names defined for the class specified by *class_name*.

> **Note:** As of PHP 4.0.6, you can specify the object itself instead of *class_name*. For example:

```
$class_methods = get_class_methods($my_class); // see below the full example
```

Example 1. get_class_methods() example

```
<?php

class myclass {
    // constructor
    function myclass() {
        return(TRUE);
    }

    // method 1
    function myfunc1() {
        return(TRUE);
    }

    // method 2
    function myfunc2() {
        return(TRUE);
    }
}

$my_object = new myclass();

$class_methods = get_class_methods(get_class($my_object));

foreach ($class_methods as $method_name) {
    echo "$method_name\n";
}

?>
```

Will produce:

```
myclass
myfunc1
myfunc2
```

See also **get_class_vars()** and **get_object_vars()**.

get_class_vars

(PHP 4)

get_class_vars -- Returns an array of default properties of the class

Description

array **get_class_vars** (string class_name)

This function will return an associative array of default properties of the class. The resulting array elements are in the form of *varname => value*.

> **Note:** Prior to PHP 4.2.0, Uninitialized class variables will not be reported by **get_class_vars()**.

Example 1. get_class_vars() example

```
<?php

class myclass {

    var $var1; // this has no default value...
    var $var2 = "xyz";
    var $var3 = 100;

    // constructor
    function myclass() {
        return(TRUE);
    }

}

$my_class = new myclass();

$class_vars = get_class_vars(get_class($my_class));

foreach ($class_vars as $name => $value) {
    echo "$name : $value\n";
}

?>
```

Will produce:

```
// Before PHP 4.2.0
var2 : xyz
var3 : 100

// As of PHP 4.2.0
var1 :
var2 : xyz
var3 : 100
```

See also **get_class_methods()**, **get_object_vars()**

get_class

(PHP 4)

get_class -- Returns the name of the class of an object

Description

string **get_class** (object obj)

This function returns the name of the class of which the object *obj* is an instance. Returns **FALSE** if *obj* is not an object.

> **Note: get_class()** returns a user defined class name in lowercase. A class defined in a PHP extension is returned in its original notation.

See also **get_parent_class()**, **gettype()**, and **is_subclass_of()**.

get_declared_classes

(PHP 4)

get_declared_classes -- Returns an array with the name of the defined classes

Description

array **get_declared_classes** (void)

This function returns an array of the names of the declared classes in the current script.

> **Note:** In PHP 4.0.1pl2, three extra classes are returned at the beginning of the array: **stdClass** (defined in `Zend/zend.c`), **OverloadedTestClass** (defined in `ext/standard/basic_functions.c`) and **Directory** (defined in `ext/standard/dir.c`).
>
> Also note that depending on what libraries you have compiled into PHP, additional classes could be present. This

means that you will not be able to define your own classes using these names. There is a list of predefined classes in the Predefined Classes section of the appendices.

See also **class_exists()**.

get_object_vars

(PHP 4)

get_object_vars -- Returns an associative array of object properties

Description

array **get_object_vars** (object obj)

This function returns an associative array of defined object properties for the specified object *obj*.

Note: In versions prior to PHP 4.2.0, if the variables declared in the class of which the *obj* is an instance, have not been assigned a value, those will not be returned in the array. In versions after PHP 4.2.0, the key will be assigned with a **NULL** value.

Example 1. Use of get_object_vars()

```
<?php
class Point2D {
    var $x, $y;
    var $label;

    function Point2D($x, $y) {
        $this->x = $x;
        $this->y = $y;
    }

    function setLabel($label) {
        $this->label = $label;
    }

    function getPoint() {
        return array("x" => $this->x,
                     "y" => $this->y,
                     "label" => $this->label);
    }
}

// "$label" is declared but not defined
$p1 = new Point2D(1.233, 3.445);
print_r(get_object_vars($p1));

$p1->setLabel("point #1");
print_r(get_object_vars($p1));

?>
```

The printout of the above program will be:

```
Array
 (
     [x] => 1.233
     [y] => 3.445
     [label] =>
 )

 Array
 (
     [x] => 1.233
     [y] => 3.445
     [label] => point #1
 )
```

See also **get_class_methods()** and **get_class_vars()**!

get_parent_class

(PHP 4)

get_parent_class -- Retrieves the parent class name for object or class

Description

string **get_parent_class** (mixed obj)

If *obj* is an object, returns the name of the parent class of the class of which *obj* is an instance.

If *obj* is a string, returns the name of the parent class of the class with that name. This functionality was added in PHP 4.0.5.

See also **get_class()** and **is_subclass_of()**

is_a

(PHP 4 >= 4.2.0)

is_a -- Returns **TRUE** if the object is of this class or has this class as one of its parents

Description

bool **is_a** (object object, string class_name)

This function returns **TRUE** if the object is of this class or has this class as one of its parents, **FALSE** otherwise.

See also **get_class()**, **get_parent_class()**, and **is_subclass_of()**.

is_subclass_of

(PHP 4)

is_subclass_of -- Returns **TRUE** if the object has this class as one of its parents

Description

bool **is_subclass_of** (object object, string class_name)

This function returns **TRUE** if the object *object*, belongs to a class which is a subclass of *class_name*, **FALSE** otherwise.

See also **get_class()**, **get_parent_class()** and **is_a()**.

method_exists

(PHP 4)

method_exists -- Checks if the class method exists

Description

bool **method_exists** (object object, string method_name)

This function returns **TRUE** if the method given by *method_name* has been defined for the given *object*, **FALSE** otherwise.

X. ClibPDF functions

Introduction

ClibPDF lets you create PDF documents with PHP. ClibPDF functionality and API are similar to PDFlib. This documentation should be read alongside the ClibPDF manual since it explains the library in much greater detail.

Many functions in the native ClibPDF and the PHP module, as well as in PDFlib, have the same name. All functions except for **cpdf_open()** take the handle for the document as their first parameter.

Currently this handle is not used internally since ClibPDF does not support the creation of several PDF documents at the same time. Actually, you should not even try it, the results are unpredictable. I can't oversee what the consequences in a multi threaded environment are. According to the author of ClibPDF this will change in one of the next releases (current version when this was written is 1.10). If you need this functionality use the pdflib module.

A nice feature of ClibPDF (and PDFlib) is the ability to create the pdf document completely in memory without using temporary files. It also provides the ability to pass coordinates in a predefined unit length. (This feature can also be simulated by **pdf_translate()** when using the PDFlib functions.)

Another nice feature of ClibPDF is the fact that any page can be modified at any time even if a new page has been already opened. The function **cpdf_set_current_page()** allows to leave the current page and presume modifying an other page.

Most of the functions are fairly easy to use. The most difficult part is probably creating a very simple PDF document at all. The following example should help you to get started. It creates a document with one page. The page contains the text "Times-Roman" in an outlined 30pt font. The text is underlined.

Requirements

In order to use the ClibPDF functions you need to install the ClibPDF package. It is available for download from FastIO, but requires that you purchase a license for commercial use. PHP requires that you use cpdflib >= 2.

Installation

To get these functions to work, you have to compile PHP with `--with-cpdflib[=DIR]`. DIR is the cpdflib install directory, defaults to `/usr`. In addition you can specify the jpeg library and the tiff library for ClibPDF to use. To do so add to your configure line the options `--with-jpeg-dir[=DIR] --with-tiff-dir[=DIR]`.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

CPDF_PM_NONE (integer)

CPDF_PM_OUTLINES (integer)

CPDF_PM_THUMBS (integer)

CPDF_PM_FULLSCREEN (integer)

CPDF_PL_SINGLE (integer)

CPDF_PL_1COLUMN (integer)

CPDF_PL_2LCOLUMN (integer)

CPDF_PL_2RCOLUMN (integer)

Examples

Example 1. Simple ClibPDF Example

```
<?php
$cpdf = cpdf_open(0);
cpdf_page_init($cpdf, 1, 0, 595, 842, 1.0);
cpdf_add_outline($cpdf, 0, 0, 0, 1, "Page 1");
cpdf_begin_text($cpdf);
cpdf_set_font($cpdf, "Times-Roman", 30, "WinAnsiEncoding");
cpdf_set_text_rendering($cpdf, 1);
cpdf_text($cpdf, "Times Roman outlined", 50, 750);
cpdf_end_text($cpdf);
cpdf_moveto($cpdf, 50, 740);
cpdf_lineto($cpdf, 330, 740);
cpdf_stroke($cpdf);
cpdf_finalize($cpdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($cpdf);
cpdf_close($cpdf);
?>
```

The pdflib distribution contains a more complex example which creates a series of pages with an analog clock. Here is that example converted into PHP using the ClibPDF extension:

Example 2. pdfclock example from pdflib 2.0 distribution

```
<?php
$radius = 200;
$margin = 20;
$pagecount = 40;

$pdf = cpdf_open(0);
cpdf_set_creator($pdf, "pdf_clock.php3");
cpdf_set_title($pdf, "Analog Clock");

while($pagecount-- > 0) {
  cpdf_page_init($pdf, $pagecount+1, 0, 2 * ($radius + $margin), 2 * ($radius + $margin), 1.0);

  cpdf_set_page_animation($pdf, 4, 0.5, 0, 0, 0);  /* wipe */

  cpdf_translate($pdf, $radius + $margin, $radius + $margin);
  cpdf_save($pdf);
  cpdf_setrgbcolor($pdf, 0.0, 0.0, 1.0);

  /* minute strokes */
  cpdf_setlinewidth($pdf, 2.0);
  for ($alpha = 0; $alpha < 360; $alpha += 6)
    {
    cpdf_rotate($pdf, 6.0);
    cpdf_moveto($pdf, $radius, 0.0);
    cpdf_lineto($pdf, $radius-$margin/3, 0.0);
    cpdf_stroke($pdf);
    }

  cpdf_restore($pdf);
  cpdf_save($pdf);

  /* 5 minute strokes */
  cpdf_setlinewidth($pdf, 3.0);
  for ($alpha = 0; $alpha < 360; $alpha += 30)
  {
    cpdf_rotate($pdf, 30.0);
    cpdf_moveto($pdf, $radius, 0.0);
    cpdf_lineto($pdf, $radius-$margin, 0.0);
    cpdf_stroke($pdf);
  }

  $ltime = getdate();

  /* draw hour hand */
  cpdf_save($pdf);
  cpdf_rotate($pdf, -(($ltime['minutes']/60.0) + $ltime['hours'] - 3.0) * 30.0);
  cpdf_moveto($pdf, -$radius/10, -$radius/20);
  cpdf_lineto($pdf, $radius/2, 0.0);
  cpdf_lineto($pdf, -$radius/10, $radius/20);
  cpdf_closepath($pdf);
  cpdf_fill($pdf);
  cpdf_restore($pdf);

  /* draw minute hand */
  cpdf_save($pdf);
  cpdf_rotate($pdf, -(($ltime['seconds']/60.0) + $ltime['minutes'] - 15.0) * 6.0);
  cpdf_moveto($pdf, -$radius/10, -$radius/20);
  cpdf_lineto($pdf, $radius * 0.8, 0.0);
  cpdf_lineto($pdf, -$radius/10, $radius/20);
  cpdf_closepath($pdf);
  cpdf_fill($pdf);
  cpdf_restore($pdf);

  /* draw second hand */
  cpdf_setrgbcolor($pdf, 1.0, 0.0, 0.0);
  cpdf_setlinewidth($pdf, 2);
  cpdf_save($pdf);
  cpdf_rotate($pdf, -(($ltime['seconds'] - 15.0) * 6.0));
  cpdf_moveto($pdf, -$radius/5, 0.0);
  cpdf_lineto($pdf, $radius, 0.0);
```

```
  cpdf_stroke($pdf);
  cpdf_restore($pdf);

  /* draw little circle at center */
  cpdf_circle($pdf, 0, 0, $radius/30);
  cpdf_fill($pdf);

  cpdf_restore($pdf);

  cpdf_finalize_page($pdf, $pagecount+1);
}

cpdf_finalize($pdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($pdf);
cpdf_close($pdf);
?>
```

See Also

See also the PDFlib extension documentation.

Table of Contents

cpdf_set_text_rendering -- Determines how text is rendered
cpdf_set_text_rise -- Sets the text rise
cpdf_set_title -- Sets the title field of the pdf document
cpdf_set_viewer_preferences -- How to show the document in the viewer
cpdf_set_word_spacing -- Sets spacing between words
cpdf_setdash -- Sets dash pattern
cpdf_setflat -- Sets flatness
cpdf_setgray_fill -- Sets filling color to gray value
cpdf_setgray_stroke -- Sets drawing color to gray value
cpdf_setgray -- Sets drawing and filling color to gray value
cpdf_setlinecap -- Sets linecap parameter
cpdf_setlinejoin -- Sets linejoin parameter
cpdf_setlinewidth -- Sets line width
cpdf_setmiterlimit -- Sets miter limit
cpdf_setrgbcolor_fill -- Sets filling color to rgb color value
cpdf_setrgbcolor_stroke -- Sets drawing color to rgb color value
cpdf_setrgbcolor -- Sets drawing and filling color to rgb color value
cpdf_show_xy -- Output text at position
cpdf_show -- Output text at current position
cpdf_stringwidth -- Returns width of text in current font
cpdf_stroke -- Draw line along path
cpdf_text -- Output text with parameters
cpdf_translate -- Sets origin of coordinate system

cpdf_add_annotation

(PHP 3>= 3.0.12, PHP 4)

cpdf_add_annotation -- Adds annotation

Description

void **cpdf_add_annotation** (int pdf document, float llx, float lly, float urx, float ury, string title, string content [, int mode])

The **cpdf_add_annotation()** adds a note with the lower left corner at (*llx*, *lly*) and the upper right corner at (*urx*, *ury*).

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

cpdf_add_outline

(PHP 3>= 3.0.9, PHP 4)

cpdf_add_outline -- Adds bookmark for current page

Description

void **cpdf_add_outline** (int pdf document, string text)

The **cpdf_add_outline()** function adds a bookmark with text *text* that points to the current page.

Example 1. Adding a page outline

```
<?php
$cpdf = cpdf_open(0);
cpdf_page_init($cpdf, 1, 0, 595, 842);
cpdf_add_outline($cpdf, 0, 0, 0, 1, "Page 1");
// ...
// some drawing
// ...
cpdf_finalize($cpdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($cpdf);
cpdf_close($cpdf);
?>
```

cpdf_arc

(PHP 3>= 3.0.8, PHP 4)

cpdf_arc -- Draws an arc

Description

void **cpdf_arc** (int pdf document, float x-coor, float y-coor, float radius, float start, float end [, int mode])

The **cpdf_arc()** function draws an arc with center at point (*x-coor*, *y-coor*) and radius *radius*, starting at angle *start* and ending at angle *end*.

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

See also **cpdf_circle()**.

cpdf_begin_text

(PHP 3>= 3.0.8, PHP 4)

cpdf_begin_text -- Starts text section

Description

void **cpdf_begin_text** (int pdf document)

The **cpdf_begin_text()** function starts a text section. It must be ended with **cpdf_end_text()**.

Example 1. Text output

```
<?php
cpdf_begin_text($pdf);
cpdf_set_font($pdf, 16, "Helvetica", "WinAnsiEncoding");
cpdf_text($pdf, 100, 100, "Some text");
cpdf_end_text($pdf)
?>
```

See also **cpdf_end_text()**.

cpdf_circle

(PHP 3>= 3.0.8, PHP 4)

cpdf_circle -- Draw a circle

Description

void **cpdf_circle** (int pdf document, float x-coor, float y-coor, float radius [, int mode])

The **cpdf_circle()** function draws a circle with center at point (*x-coor*, *y-coor*) and radius *radius*.

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

See also **cpdf_arc()**.

cpdf_clip

(PHP 3>= 3.0.8, PHP 4)

cpdf_clip -- Clips to current path

Description

void **cpdf_clip** (int pdf document)

The **cpdf_clip()** function clips all drawing to the current path.

cpdf_close

(PHP 3>= 3.0.8, PHP 4)

cpdf_close -- Closes the pdf document

Description

void **cpdf_close** (int pdf document)

The **cpdf_close()** function closes the pdf document. This should be the last function even after **cpdf_finalize()**, **cpdf_output_buffer()** and **cpdf_save_to_file()**.

See also **cpdf_open()**.

cpdf_closepath_fill_stroke

(PHP 3>= 3.0.8, PHP 4)

cpdf_closepath_fill_stroke -- Close, fill and stroke current path

Description

void **cpdf_closepath_fill_stroke** (int pdf document)

The **cpdf_closepath_fill_stroke()** function closes, fills the interior of the current path with the current fill color and draws current path.

See also **cpdf_closepath()**, **cpdf_stroke()**, **cpdf_fill()**, **cpdf_setgray_fill()**, **cpdf_setgray()**, **cpdf_setrgbcolor_fill()**, **cpdf_setrgbcolor()**.

cpdf_closepath_stroke

(PHP 3>= 3.0.8, PHP 4)

cpdf_closepath_stroke -- Close path and draw line along path

Description

void **cpdf_closepath_stroke** (int pdf document)

The **cpdf_closepath_stroke()** function is a combination of **cpdf_closepath()** and **cpdf_stroke()**. Then clears the path.

See also **cpdf_closepath()**, **cpdf_stroke()**.

cpdf_closepath

(PHP 3>= 3.0.8, PHP 4)

cpdf_closepath -- Close path

Description

void **cpdf_closepath** (int pdf document)

The **cpdf_closepath()** function closes the current path.

cpdf_continue_text

(PHP 3>= 3.0.8, PHP 4)

cpdf_continue_text -- Output text in next line

Description

void **cpdf_continue_text** (int pdf document, string text)

The **cpdf_continue_text()** function outputs the string in *text* in the next line.

See also **cpdf_show_xy()**, **cpdf_text()**, **cpdf_set_leading()**, **cpdf_set_text_pos()**.

cpdf_curveto

(PHP 3>= 3.0.8, PHP 4)

cpdf_curveto -- Draws a curve

Description

void **cpdf_curveto** (int pdf document, float x1, float y1, float x2, float y2, float x3, float y3 [, int mode])

The **cpdf_curveto()** function draws a Bezier curve from the current point to the point (*x3*, *y3*) using (*x1*, *y1*) and (*x2*, *y2*) as control points.

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

See also **cpdf_moveto()**, **cpdf_rmoveto()**, **cpdf_rlineto()**, **cpdf_lineto()**.

cpdf_end_text

(PHP 3>= 3.0.8, PHP 4)

cpdf_end_text -- Ends text section

Description

void **cpdf_end_text** (int pdf document)

The **cpdf_end_text()** function ends a text section which was started with **cpdf_begin_text()**.

Example 1. Text output

```
<?php
cpdf_begin_text($pdf);
cpdf_set_font($pdf, 16, "Helvetica", "WinAnsiEncoding");
cpdf_text($pdf, 100, 100, "Some text");
cpdf_end_text($pdf)
?>
```

See also **cpdf_begin_text()**.

cpdf_fill_stroke

(PHP 3>= 3.0.8, PHP 4)

cpdf_fill_stroke -- Fill and stroke current path

Description

void **cpdf_fill_stroke** (int pdf document)

The **cpdf_fill_stroke()** function fills the interior of the current path with the current fill color and draws current path.

See also **cpdf_closepath()**, **cpdf_stroke()**, **cpdf_fill()**, **cpdf_setgray_fill()**, **cpdf_setgray()**, **cpdf_setrgbcolor_fill()**, **cpdf_setrgbcolor()**.

cpdf_fill

(PHP 3>= 3.0.8, PHP 4)

cpdf_fill -- Fill current path

Description

void **cpdf_fill** (int pdf document)

The **cpdf_fill()** function fills the interior of the current path with the current fill color.

See also **cpdf_closepath()**, **cpdf_stroke()**, **cpdf_setgray_fill()**, **cpdf_setgray()**, **cpdf_setrgbcolor_fill()**, **cpdf_setrgbcolor()**.

cpdf_finalize_page

(PHP 3>= 3.0.10, PHP 4)

cpdf_finalize_page -- Ends page

Description

void **cpdf_finalize_page** (int pdf document, int page number)

The **cpdf_finalize_page()** function ends the page with page number *page number*.

This function is only for saving memory. A finalized page takes less memory but cannot be modified anymore.

See also **cpdf_page_init()**.

cpdf_finalize

(PHP 3>= 3.0.8, PHP 4)

cpdf_finalize -- Ends document

Description

void **cpdf_finalize** (int pdf document)

The **cpdf_finalize()** function ends the document. You still have to call **cpdf_close()**

See also **cpdf_close()**.

cpdf_global_set_document_limits

(PHP 4)

cpdf_global_set_document_limits -- Sets document limits for any pdf document

Description

void **cpdf_global_set_document_limits** (int maxpages, int maxfonts, int maximages, int maxannotations, int maxobjects)

The **cpdf_global_set_document_limits()** function sets several document limits. This function has to be called before **cpdf_open()** to take effect. It sets the limits for any document open afterwards.

See also **cpdf_open()**.

cpdf_import_jpeg

(PHP 3>= 3.0.9, PHP 4)

cpdf_import_jpeg -- Opens a JPEG image

Description

int **cpdf_import_jpeg** (int pdf document, string file name, float x-coor, float y-coor, float angle, float width, float height, float x-scale, float y-scale [, int mode])

The **cpdf_import_jpeg()** function opens an image stored in the file with the name *file name*. The format of the image has to be jpeg. The image is placed on the current page at position (*x-coor*, *y-coor*). The image is rotated by *angle* degrees.

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

See also **cpdf_place_inline_image()**.

cpdf_lineto

(PHP 3>= 3.0.8, PHP 4)

cpdf_lineto -- Draws a line

Description

void **cpdf_lineto** (int pdf document, float x-coor, float y-coor [, int mode])

The **cpdf_lineto()** function draws a line from the current point to the point with coordinates (*x-coor*, *y-coor*).

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

See also **cpdf_moveto()**, **cpdf_rmoveto()**, **cpdf_curveto()**.

cpdf_moveto

(PHP 3>= 3.0.8, PHP 4)

cpdf_moveto -- Sets current point

Description

void **cpdf_moveto** (int pdf document, float x-coor, float y-coor [, int mode])

The **cpdf_moveto()** function set the current point to the coordinates *x-coor* and *y-coor*.

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

cpdf_newpath

(PHP 3>= 3.0.9, PHP 4)

cpdf_newpath -- Starts a new path

Description

void **cpdf_newpath** (int pdf document)

The **cpdf_newpath()** starts a new path on the document given by the *pdf document* parameter.

cpdf_open

(PHP 3>= 3.0.8, PHP 4)

cpdf_open -- Opens a new pdf document

Description

int **cpdf_open** (int compression [, string filename])

The **cpdf_open()** function opens a new pdf document. The first parameter turns document compression on if it is unequal to 0. The second optional parameter sets the file in which the document is written. If it is omitted the document is created in memory and can either be written into a file with the **cpdf_save_to_file()** or written to standard output with **cpdf_output_buffer()**.

> **Note:** The return value will be needed in further versions of ClibPDF as the first parameter in all other functions which are writing to the pdf document.
>
> The ClibPDF library takes the filename "-" as a synonym for stdout. If PHP is compiled as an apache module this will not work because the way ClibPDF outputs to stdout does not work with apache. You can solve this problem by skipping the filename and using **cpdf_output_buffer()** to output the pdf document.

See also **cpdf_close()**, **cpdf_output_buffer()**.

cpdf_output_buffer

(PHP 3>= 3.0.9, PHP 4)

cpdf_output_buffer -- Outputs the pdf document in memory buffer

Description

void **cpdf_output_buffer** (int pdf document)

The **cpdf_output_buffer()** function outputs the pdf document to stdout. The document has to be created in memory which is the case if **cpdf_open()** has been called with no filename parameter.

See also **cpdf_open()**.

cpdf_page_init

(PHP 3>= 3.0.8, PHP 4)

cpdf_page_init -- Starts new page

Description

void **cpdf_page_init** (int pdf document, int page number, int orientation, float height, float width [, float unit])

The **cpdf_page_init()** function starts a new page with height *height* and width *width*. The page has number *page number* and orientation *orientation*. *orientation* can be 0 for portrait and 1 for landscape. The last optional parameter *unit* sets the unit for the coordinate system. The value should be the number of postscript points per unit. Since one inch is equal to 72 points, a value of 72 would set the unit to one inch. The default is also 72.

See also **cpdf_set_current_page()**.

cpdf_place_inline_image

(PHP 3>= 3.0.9, PHP 4)

cpdf_place_inline_image -- Places an image on the page

Description

void **cpdf_place_inline_image** (int pdf document, int image, float x-coor, float y-coor, float angle, float width, float height [, int mode])

The **cpdf_place_inline_image()** function places an image created with the php image functions on the page at position (*x-coor*, *y-coor*). The image can be scaled at the same time.

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

See also **cpdf_import_jpeg()**.

cpdf_rect

(PHP 3>= 3.0.8, PHP 4)

cpdf_rect -- Draw a rectangle

Description

void **cpdf_rect** (int pdf document, float x-coor, float y-coor, float width, float height [, int mode])

The **cpdf_rect()** function draws a rectangle with its lower left corner at point (*x-coor*, *y-coor*). This width is set to *width*. This height is set to *height*.

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

cpdf_restore

(PHP 3>= 3.0.8, PHP 4)

cpdf_restore -- Restores formerly saved environment

Description

void **cpdf_restore** (int pdf document)

The **cpdf_restore()** function restores the environment saved with **cpdf_save()**. It works like the postscript command grestore. Very useful if you want to translate or rotate an object without effecting other objects.

Example 1. Save/Restore

```
<?php
cpdf_save($pdf);
// do all kinds of rotations, transformations, ...
cpdf_restore($pdf)
?>
```

See also **cpdf_save()**.

cpdf_rlineto

(PHP 3>= 3.0.9, PHP 4)

cpdf_rlineto -- Draws a line

Description

void **cpdf_rlineto** (int pdf document, float x-coor, float y-coor [, int mode])

The **cpdf_rlineto()** function draws a line from the current point to the relative point with coordinates (*x-coor*, *y-coor*).

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

See also **cpdf_moveto()**, **cpdf_rmoveto()**, **cpdf_curveto()**.

cpdf_rmoveto

(PHP 3>= 3.0.9, PHP 4)

cpdf_rmoveto -- Sets current point

Description

void **cpdf_rmoveto** (int pdf document, float x-coor, float y-coor [, int mode])

The **cpdf_rmoveto()** function set the current point relative to the coordinates *x-coor* and *y-coor*.

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

See also **cpdf_moveto()**.

cpdf_rotate_text

(PHP 3>= 3.0.9, PHP 4)

cpdf_rotate_text -- Sets text rotation angle

Description

void **cpdf_rotate_text** (int pdfdoc, float angle)

Warning
This function is currently not documented; only the argument list is available.

cpdf_rotate

(PHP 3>= 3.0.8, PHP 4)

cpdf_rotate -- Sets rotation

Description

void **cpdf_rotate** (int pdf document, float angle)

The **cpdf_rotate()** function set the rotation in degrees to *angle*.

cpdf_save_to_file

(PHP 3>= 3.0.8, PHP 4)

cpdf_save_to_file -- Writes the pdf document into a file

Description

void **cpdf_save_to_file** (int pdf document, string filename)

The **cpdf_save_to_file()** function outputs the pdf document into a file if it has been created in memory.

This function is not needed if the pdf document has been open by specifying a filename as a parameter of **cpdf_open()**.

See also **cpdf_output_buffer()**, **cpdf_open()**.

cpdf_save

(PHP 3>= 3.0.8, PHP 4)

cpdf_save -- Saves current environment

Description

void **cpdf_save** (int pdf document)

The **cpdf_save()** function saves the current environment. It works like the postscript command gsave. Very useful if you want to translate or rotate an object without effecting other objects.

See also **cpdf_restore()**.

cpdf_scale

(PHP 3>= 3.0.8, PHP 4)

cpdf_scale -- Sets scaling

Description

void **cpdf_scale** (int pdf document, float x-scale, float y-scale)

The **cpdf_scale()** function set the scaling factor in both directions.

cpdf_set_action_url

(PHP 3>= 3.0.9, PHP 4)

cpdf_set_action_url -- Sets hyperlink

Description

void **cpdf_set_action_url** (int pdfdoc, float xll, float yll, float xur, float xur, string url [, int mode])

Warning
This function is currently not documented; only the argument list is available.

cpdf_set_char_spacing

(PHP 3>= 3.0.8, PHP 4)

cpdf_set_char_spacing -- Sets character spacing

Description

void **cpdf_set_char_spacing** (int pdf document, float space)

The **cpdf_set_char_spacing()** function sets the spacing between characters.

See also **cpdf_set_word_spacing()**, **cpdf_set_leading()**.

cpdf_set_creator

(PHP 3>= 3.0.8, PHP 4)

cpdf_set_creator -- Sets the creator field in the pdf document

Description

void **cpdf_set_creator** (string creator)

The **cpdf_set_creator()** function sets the creator of a pdf document.

See also **cpdf_set_subject()**, **cpdf_set_title()**, **cpdf_set_keywords()**.

cpdf_set_current_page

(PHP 3>= 3.0.9, PHP 4)

cpdf_set_current_page -- Sets current page

Description

void **cpdf_set_current_page** (int pdf document, int page number)

The **cpdf_set_current_page()** function set the page on which all operations are performed. One can switch between pages until a page is finished with **cpdf_finalize_page()**.

See also **cpdf_finalize_page()**.

cpdf_set_font_directories

(PHP 4 >= 4.0.6)

cpdf_set_font_directories -- Sets directories to search when using external fonts

Description

void **cpdf_set_font_directories** (int pdfdoc, string pfmdir, string pfbdir)

Warning
This function is currently not documented; only the argument list is available.

cpdf_set_font_map_file

(PHP 4 >= 4.0.6)

cpdf_set_font_map_file -- Sets fontname to filename translation map when using external fonts

Description

void **cpdf_set_font_map_file** (int pdfdoc, string filename)

Warning

This function is currently not documented; only the argument list is available.

cpdf_set_font

(PHP 3>= 3.0.8, PHP 4)

cpdf_set_font -- Select the current font face and size

Description

void **cpdf_set_font** (int pdf document, string font name, float size, string encoding)

The **cpdf_set_font()** function sets the current font face, font size and encoding. Currently only the standard postscript fonts are supported.

The last parameter *encoding* can take the following values: "MacRomanEncoding", "MacExpertEncoding", "WinAnsiEncoding", and "**NULL**". "**NULL**" stands for the font's built-in encoding.

See the ClibPDF Manual for more information, especially how to support asian fonts.

cpdf_set_horiz_scaling

(PHP 3>= 3.0.8, PHP 4)

cpdf_set_horiz_scaling -- Sets horizontal scaling of text

Description

void **cpdf_set_horiz_scaling** (int pdf document, float scale)

The **cpdf_set_horiz_scaling()** function sets the horizontal scaling to *scale* percent.

cpdf_set_keywords

(PHP 3>= 3.0.8, PHP 4)

cpdf_set_keywords -- Sets the keywords field of the pdf document

Description

void **cpdf_set_keywords** (string keywords)

The **cpdf_set_keywords()** function sets the keywords of a pdf document.

See also **cpdf_set_title()**, **cpdf_set_creator()**, **cpdf_set_subject()**.

cpdf_set_leading

(PHP 3>= 3.0.8, PHP 4)

cpdf_set_leading -- Sets distance between text lines

Description

void **cpdf_set_leading** (int pdf document, float distance)

The **cpdf_set_leading()** function sets the distance between text lines. This will be used if text is output by **cpdf_continue_text()**.

See also **cpdf_continue_text()**.

cpdf_set_page_animation

(PHP 3>= 3.0.9, PHP 4)

cpdf_set_page_animation -- Sets duration between pages

Description

void **cpdf_set_page_animation** (int pdf document, int transition, float duration)

The **cpdf_set_page_animation()** function set the transition between following pages.

The value of *transition* can be

0 for none,
1 for two lines sweeping across the screen reveal the page,
2 for multiple lines sweeping across the screen reveal the page,
3 for a box reveals the page,
4 for a single line sweeping across the screen reveals the page,
5 for the old page dissolves to reveal the page,
6 for the dissolve effect moves from one screen edge to another,
7 for the old page is simply replaced by the new page (default)

The value of *duration* is the number of seconds between page flipping.

cpdf_set_subject

(PHP 3>= 3.0.8, PHP 4)

cpdf_set_subject -- Sets the subject field of the pdf document

Description

void **cpdf_set_subject** (string subject)

The **cpdf_set_subject()** function sets the subject of a pdf document.

See also **cpdf_set_title()**, **cpdf_set_creator()**, **cpdf_set_keywords()**.

cpdf_set_text_matrix

(PHP 3>= 3.0.8, PHP 4)

cpdf_set_text_matrix -- Sets the text matrix

Description

void **cpdf_set_text_matrix** (int pdf document, array matrix)

The **cpdf_set_text_matrix()** function sets a matrix which describes a transformation applied on the current text font.

cpdf_set_text_pos

(PHP 3>= 3.0.8, PHP 4)

cpdf_set_text_pos -- Sets text position

Description

void **cpdf_set_text_pos** (int pdf document, float x-coor, float y-coor [, int mode])

The **cpdf_set_text_pos()** function sets the position of text for the next **cpdf_show()** function call.

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

See also **cpdf_show()**, **cpdf_text()**.

cpdf_set_text_rendering

(PHP 3>= 3.0.8, PHP 4)

cpdf_set_text_rendering -- Determines how text is rendered

Description

void **cpdf_set_text_rendering** (int pdf document, int mode)

The **cpdf_set_text_rendering()** function determines how text is rendered.

The possible values for *mode* are 0=fill text, 1=stroke text, 2=fill and stroke text, 3=invisible, 4=fill text and add it to clipping path, 5=stroke text and add it to clipping path, 6=fill and stroke text and add it to clipping path, 7=add it to clipping path.

cpdf_set_text_rise

(PHP 3>= 3.0.8, PHP 4)

cpdf_set_text_rise -- Sets the text rise

Description

void **cpdf_set_text_rise** (int pdf document, float value)

The **cpdf_set_text_rise()** function sets the text rising to *value* units.

cpdf_set_title

(PHP 3>= 3.0.8, PHP 4)

cpdf_set_title -- Sets the title field of the pdf document

Description

void **cpdf_set_title** (string title)

The **cpdf_set_title()** function sets the title of a pdf document.

See also **cpdf_set_subject()**, **cpdf_set_creator()**, **cpdf_set_keywords()**.

cpdf_set_viewer_preferences

(PHP 3>= 3.0.9, PHP 4)

cpdf_set_viewer_preferences -- How to show the document in the viewer

Description

void **cpdf_set_viewer_preferences** (int pdfdoc, array preferences)

Warning
This function is currently not documented; only the argument list is available.

cpdf_set_word_spacing

(PHP 3>= 3.0.8, PHP 4)

cpdf_set_word_spacing -- Sets spacing between words

Description

void **cpdf_set_word_spacing** (int pdf document, float space)

The **cpdf_set_word_spacing()** function sets the spacing between words.

See also **cpdf_set_char_spacing()**, **cpdf_set_leading()**.

cpdf_setdash

(PHP 3>= 3.0.8, PHP 4)

cpdf_setdash -- Sets dash pattern

Description

void **cpdf_setdash** (int pdf document, float white, float black)

The **cpdf_setdash()** function set the dash pattern *white* white units and *black* black units. If both are 0 a solid line is set.

cpdf_setflat

(PHP 3>= 3.0.8, PHP 4)

cpdf_setflat -- Sets flatness

Description

void **cpdf_setflat** (int pdf document, float value)

The **cpdf_setflat()** function set the flatness to a value between 0 and 100.

cpdf_setgray_fill

(PHP 3>= 3.0.8, PHP 4)

cpdf_setgray_fill -- Sets filling color to gray value

Description

void **cpdf_setgray_fill** (int pdf document, float value)

The **cpdf_setgray_fill()** function sets the current gray value to fill a path.

See also **cpdf_setrgbcolor_fill()**.

cpdf_setgray_stroke

(PHP 3>= 3.0.8, PHP 4)

cpdf_setgray_stroke -- Sets drawing color to gray value

Description

void **cpdf_setgray_stroke** (int pdf document, float gray value)

The **cpdf_setgray_stroke()** function sets the current drawing color to the given gray value.

See also **cpdf_setrgbcolor_stroke()**.

cpdf_setgray

(PHP 3>= 3.0.8, PHP 4)

cpdf_setgray -- Sets drawing and filling color to gray value

Description

void **cpdf_setgray** (int pdf document, float gray value)

The **cpdf_setgray()** function sets the current drawing and filling color to the given gray value.

See also **cpdf_setrgbcolor_stroke()**, **cpdf_setrgbcolor_fill()**.

cpdf_setlinecap

(PHP 3>= 3.0.8, PHP 4)

cpdf_setlinecap -- Sets linecap parameter

Description

void **cpdf_setlinecap** (int pdf document, int value)

The **cpdf_setlinecap()** function set the linecap parameter between a value of 0 and 2. 0 = butt end, 1 = round, 2 = projecting square.

cpdf_setlinejoin

(PHP 3>= 3.0.8, PHP 4)

cpdf_setlinejoin -- Sets linejoin parameter

Description

void **cpdf_setlinejoin** (int pdf document, long value)

The **cpdf_setlinejoin()** function set the linejoin parameter between a value of 0 and 2. 0 = miter, 1 = round, 2 = bevel.

cpdf_setlinewidth

(PHP 3>= 3.0.8, PHP 4)

cpdf_setlinewidth -- Sets line width

Description

void **cpdf_setlinewidth** (int pdf document, float width)

The **cpdf_setlinewidth()** function set the line width to *width*.

cpdf_setmiterlimit

(PHP 3>= 3.0.8, PHP 4)

cpdf_setmiterlimit -- Sets miter limit

Description

void **cpdf_setmiterlimit** (int pdf document, float value)

The **cpdf_setmiterlimit()** function set the miter limit to a value greater or equal than 1.

cpdf_setrgbcolor_fill

(PHP 3>= 3.0.8, PHP 4)

cpdf_setrgbcolor_fill -- Sets filling color to rgb color value

Description

void **cpdf_setrgbcolor_fill** (int pdf document, float red value, float green value, float blue value)

The **cpdf_setrgbcolor_fill()** function sets the current rgb color value to fill a path.

See also **cpdf_setrgbcolor_stroke()**, **cpdf_setrgbcolor()**.

cpdf_setrgbcolor_stroke

(PHP 3>= 3.0.8, PHP 4)

cpdf_setrgbcolor_stroke -- Sets drawing color to rgb color value

Description

void **cpdf_setrgbcolor_stroke** (int pdf document, float red value, float green value, float blue value)

The **cpdf_setrgbcolor_stroke()** function sets the current drawing color to the given rgb color value.

See also **cpdf_setrgbcolor_fill()**, **cpdf_setrgbcolor()**.

cpdf_setrgbcolor

(PHP 3>= 3.0.8, PHP 4)

cpdf_setrgbcolor -- Sets drawing and filling color to rgb color value

Description

void **cpdf_setrgbcolor** (int pdf document, float red value, float green value, float blue value)

The **cpdf_setrgbcolor()** function sets the current drawing and filling color to the given rgb color value.

See also **cpdf_setrgbcolor_stroke()**, **cpdf_setrgbcolor_fill()**.

cpdf_show_xy

(PHP 3>= 3.0.8, PHP 4)

cpdf_show_xy -- Output text at position

Description

void **cpdf_show_xy** (int pdf document, string text, float x-coor, float y-coor [, int mode])

The **cpdf_show_xy()** function outputs the string *text* at position with coordinates (*x-coor*, *y-coor*).

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

Note: The function **cpdf_show_xy()** is identical to **cpdf_text()** without the optional parameters.

See also **cpdf_text()**.

cpdf_show

(PHP 3>= 3.0.8, PHP 4)

cpdf_show -- Output text at current position

Description

void **cpdf_show** (int pdf document, string text)

The **cpdf_show()** function outputs the string in *text* at the current position.

See also **cpdf_text()**, **cpdf_begin_text()**, **cpdf_end_text()**.

cpdf_stringwidth

(PHP 3>= 3.0.8, PHP 4)

cpdf_stringwidth -- Returns width of text in current font

Description

float **cpdf_stringwidth** (int pdf document, string text)

The **cpdf_stringwidth()** function returns the width of the string in *text*. It requires a font to be set before.

See also **cpdf_set_font()**.

cpdf_stroke

(PHP 3>= 3.0.8, PHP 4)

cpdf_stroke -- Draw line along path

Description

void **cpdf_stroke** (int pdf document)

The **cpdf_stroke()** function draws a line along current path.

See also **cpdf_closepath()**, **cpdf_closepath_stroke()**.

cpdf_text

(PHP 3>= 3.0.8, PHP 4)

cpdf_text -- Output text with parameters

Description

void **cpdf_text** (int pdf document, string text, float x-coor, float y-coor [, int mode [, float orientation [, int alignmode]]])

The **cpdf_text()** function outputs the string *text* at position with coordinates (*x-coor*, *y-coor*).

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit. The optional parameter *orientation* is the rotation of the text in degree. The optional parameter *alignmode* determines how the text is aligned.

See the ClibPDF documentation for possible values.

See also **cpdf_show_xy()**.

cpdf_translate

(PHP 3>= 3.0.8, PHP 4)

cpdf_translate -- Sets origin of coordinate system

Description

void **cpdf_translate** (int pdf document, float x-coor, float y-coor [, int mode])

The **cpdf_translate()** function set the origin of coordinate system to the point (*x-coor*, *y-coor*).

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

XI. Crack functions

Introduction

These functions allow you to use the CrackLib library to test the 'strength' of a password. The 'strength' of a password is tested by that checks length, use of upper and lower case and checked against the specified CrackLib dictionary. CrackLib will also give helpful diagnostic messages that will help 'strengthen' the password.

Requirements

More information regarding CrackLib along with the library can be found at http://www.users.dircon.co.uk/~crypto/.

Installation

In order to use these functions, you must compile PHP with Crack support by using the `--with-crack[=DIR]` option.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. Crack configuration options

Name	Default	Changeable
crack.default_dictionary	NULL	PHP_INI_SYSTEM

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Examples

This example shows how to open a CrackLib dictionary, test a given password, retrieve any diagnostic messages, and close the dictionary.

Example 1. CrackLib example

```
<?php
// Open CrackLib Dictionary
$dictionary = crack_opendict('/usr/local/lib/pw_dict')
     or die('Unable to open CrackLib dictionary');

// Perform password check
$check = crack_check($dictionary, 'gx9A2s0x');

// Retrieve messages
$diag = crack_getlastmessage();
echo $diag; // 'strong password'

// Close dictionary
crack_closedict($dictionary);
?>
```

Note: If **crack_check()** returns **TRUE**, **crack_getlastmessage()** will return 'strong password'.

Table of Contents

crack_check

(PHP 4 >= 4.0.5)

crack_check -- Performs an obscure check with the given password

Description

bool **crack_check** ([resource dictionary, string password])

Returns **TRUE** if *password* is strong, or **FALSE** otherwise.

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

crack_check() performs an obscure check with the given *password* on the specified *dictionary* . If *dictionary* is not specified, the last opened dictionary is used.

crack_closedict

(PHP 4 >= 4.0.5)

crack_closedict -- Closes an open CrackLib dictionary

Description

bool **crack_closedict** ([resource dictionary])

Returns **TRUE** on success or **FALSE** on failure.

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

crack_closedict() closes the specified *dictionary* identifier. If *dictionary* is not specified, the current dictionary is closed.

crack_getlastmessage

(PHP 4 >= 4.0.5)

crack_getlastmessage -- Returns the message from the last obscure check

Description

string **crack_getlastmessage** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

crack_getlastmessage() returns the message from the last obscure check.

crack_opendict

(PHP 4 >= 4.0.5)

crack_opendict -- Opens a new CrackLib dictionary

Description

resource **crack_opendict** (string dictionary)

Returns a dictionary resource identifier on success, or **FALSE** on failure.

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

crack_opendict() opens the specified CrackLib *dictionary* for use with **crack_check()**.

> **Note:** Only one dictionary may be open at a time.

See also: **crack_check()**, and **crack_closedict()**.

XII. CURL, Client URL Library Functions

Introduction

PHP supports libcurl, a library created by Daniel Stenberg, that allows you to connect and communicate to many different types of servers with many different types of protocols. libcurl currently supports the http, https, ftp, gopher, telnet, dict, file, and ldap protocols. libcurl also supports HTTPS certificates, HTTP POST, HTTP PUT, FTP uploading (this can also be done with PHP's ftp extension), HTTP form based upload, proxies, cookies, and user+password authentication.

These functions have been added in PHP 4.0.2.

Requirements

In order to use the CURL functions you need to install the CURL package. PHP requires that you use CURL 7.0.2-beta or higher. PHP will not work with any version of CURL below version 7.0.2-beta. From PHP version 4.2.3 you will atleast need CURL version 7.9.0 or higher.

Installation

To use PHP's CURL support you must also compile PHP `--with-curl[=DIR]` where DIR is the location of the directory containing the lib and include directories. In the "include" directory there should be a folder named "curl" which should contain the `easy.h` and `curl.h` files. There should be a file named `libcurl.a` located in the "lib" directory. Beginning with PHP 4.3.0 you can configure PHP to use CURL for url streams `--with-curlwrappers`.

> **Note to Win32 Users:** In order to enable this module on a Windows environment, you must copy *libeay32.dll* and *ssleay32.dll* from the DLL folder of the PHP/Win32 binary package to the SYSTEM32 folder of your windows machine. (Ex: C:\WINNT\SYSTEM32 or C:\WINDOWS\SYSTEM32)

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

CURLOPT_PORT (integer)

CURLOPT_FILE (integer)

CURLOPT_INFILE (integer)

CURLOPT_INFILESIZE (integer)

CURLOPT_URL (integer)

CURLOPT_PROXY (integer)

CURLOPT_VERBOSE (integer)

CURLOPT_HEADER (integer)

CURLOPT_HTTPHEADER (integer)

CURLOPT_NOPROGRESS (integer)

CURLOPT_NOBODY (integer)

CURLOPT_FAILONERROR (integer)

CURLOPT_UPLOAD (integer)

CURLOPT_POST (integer)

CURLOPT_FTPLISTONLY (integer)

CURLOPT_FTPAPPEND (integer)

CURLOPT_NETRC (integer)

CURLOPT_FOLLOWLOCATION (integer)

CURLOPT_FTPASCII (integer)

CURLOPT_PUT (integer)

CURLOPT_MUTE (integer)

CURLOPT_USERPWD (integer)

CURLOPT_PROXYUSERPWD (integer)

CURLOPT_RANGE (integer)

CURLOPT_TIMEOUT (integer)

CURLOPT_POSTFIELDS (integer)

CURLOPT_REFERER (integer)

CURLOPT_USERAGENT (integer)

CURLOPT_FTPPORT (integer)

CURLOPT_LOW_SPEED_LIMIT (integer)

CURLOPT_LOW_SPEED_TIME (integer)

CURLOPT_RESUME_FROM (integer)

CURLOPT_COOKIE (integer)

CURLOPT_SSLCERT (integer)

CURLOPT_SSLCERTPASSWD (integer)

CURLOPT_WRITEHEADER (integer)

CURLOPT_SSL_VERIFYHOST (integer)

CURLOPT_COOKIEFILE (integer)

CURLOPT_SSLVERSION (integer)

CURLOPT_TIMECONDITION (integer)

CURLOPT_TIMEVALUE (integer)

CURLOPT_CUSTOMREQUEST (integer)

CURLOPT_STDERR (integer)

CURLOPT_TRANSFERTEXT (integer)

CURLOPT_RETURNTRANSFER (integer)

CURLOPT_QUOTE (integer)

CURLOPT_POSTQUOTE (integer)

CURLOPT_INTERFACE (integer)

CURLOPT_KRB4LEVEL (integer)

CURLOPT_HTTPPROXYTUNNEL (integer)

CURLOPT_FILETIME (integer)

CURLOPT_WRITEFUNCTION (integer)

CURLOPT_READFUNCTION (integer)

CURLOPT_PASSWDFUNCTION (integer)

CURLOPT_HEADERFUNCTION (integer)

CURLOPT_MAXREDIRS (integer)

CURLOPT_MAXCONNECTS (integer)

CURLOPT_CLOSEPOLICY (integer)

CURLOPT_FRESH_CONNECT (integer)

CURLOPT_FORBID_REUSE (integer)

CURLOPT_RANDOM_FILE (integer)

CURLOPT_EGDSOCKET (integer)

CURLOPT_CONNECTTIMEOUT (integer)

CURLOPT_SSL_VERIFYPEER (integer)

CURLOPT_CAINFO (integer)

CURLOPT_COOKIEJAR (integer)

CURLOPT_SSL_CIPHER_LIST (integer)

CURLOPT_BINARYTRANSFER (integer)

CURLCLOSEPOLICY_LEAST_RECENTLY_USED (integer)

CURLCLOSEPOLICY_LEAST_TRAFFIC (integer)

CURLCLOSEPOLICY_SLOWEST (integer)

CURLCLOSEPOLICY_CALLBACK (integer)

CURLCLOSEPOLICY_OLDEST (integer)

CURLINFO_EFFECTIVE_URL (integer)

CURLINFO_HTTP_CODE (integer)

CURLINFO_HEADER_SIZE (integer)

CURLINFO_REQUEST_SIZE (integer)

CURLINFO_TOTAL_TIME (integer)

CURLINFO_NAMELOOKUP_TIME (integer)

CURLINFO_CONNECT_TIME (integer)

CURLINFO_PRETRANSFER_TIME (integer)

CURLINFO_SIZE_UPLOAD (integer)

CURLINFO_SIZE_DOWNLOAD (integer)

CURLINFO_SPEED_DOWNLOAD (integer)

CURLINFO_SPEED_UPLOAD (integer)

CURLINFO_FILETIME (integer)

CURLINFO_SSL_VERIFYRESULT (integer)

CURLINFO_CONTENT_LENGTH_DOWNLOAD (integer)

CURLINFO_CONTENT_LENGTH_UPLOAD (integer)

CURLE_OK (integer)

CURLE_UNSUPPORTED_PROTOCOL (integer)

CURLE_FAILED_INIT (integer)

CURLE_URL_MALFORMAT (integer)

CURLE_URL_MALFORMAT_USER (integer)

CURLE_COULDNT_RESOLVE_PROXY (integer)

CURLE_COULDNT_RESOLVE_HOST (integer)

CURLE_COULDNT_CONNECT (integer)

CURLE_FTP_WEIRD_SERVER_REPLY (integer)

CURLE_FTP_ACCESS_DENIED (integer)

CURLE_FTP_USER_PASSWORD_INCORRECT (integer)

CURLE_FTP_WEIRD_PASS_REPLY (integer)

CURLE_FTP_WEIRD_USER_REPLY (integer)

CURLE_FTP_WEIRD_PASV_REPLY (integer)

CURLE_FTP_WEIRD_227_FORMAT (integer)

CURLE_FTP_CANT_GET_HOST (integer)

CURLE_FTP_CANT_RECONNECT (integer)

CURLE_FTP_COULDNT_SET_BINARY (integer)

CURLE_PARTIAL_FILE (integer)

CURLE_FTP_COULDNT_RETR_FILE (integer)

CURLE_FTP_WRITE_ERROR (integer)

CURLE_FTP_QUOTE_ERROR (integer)

CURLE_HTTP_NOT_FOUND (integer)

CURLE_WRITE_ERROR (integer)

CURLE_MALFORMAT_USER (integer)

CURLE_FTP_COULDNT_STOR_FILE (integer)

CURLE_READ_ERROR (integer)

CURLE_OUT_OF_MEMORY (integer)

CURLE_OPERATION_TIMEOUTED (integer)

CURLE_FTP_COULDNT_SET_ASCII (integer)

CURLE_FTP_PORT_FAILED (integer)

CURLE_FTP_COULDNT_USE_REST (integer)

CURLE_FTP_COULDNT_GET_SIZE (integer)

CURLE_HTTP_RANGE_ERROR (integer)

CURLE_HTTP_POST_ERROR (integer)

CURLE_SSL_CONNECT_ERROR (integer)

CURLE_FTP_BAD_DOWNLOAD_RESUME (integer)

CURLE_FILE_COULDNT_READ_FILE (integer)

CURLE_LDAP_CANNOT_BIND (integer)

CURLE_LDAP_SEARCH_FAILED (integer)

CURLE_LIBRARY_NOT_FOUND (integer)

CURLE_FUNCTION_NOT_FOUND (integer)

CURLE_ABORTED_BY_CALLBACK (integer)

CURLE_BAD_FUNCTION_ARGUMENT (integer)

CURLE_BAD_CALLING_ORDER (integer)

CURLE_HTTP_PORT_FAILED (integer)

CURLE_BAD_PASSWORD_ENTERED (integer)

CURLE_TOO_MANY_REDIRECTS (integer)

CURLE_UNKNOWN_TELNET_OPTION (integer)

CURLE_TELNET_OPTION_SYNTAX (integer)

CURLE_OBSOLETE (integer)

CURLE_SSL_PEER_CERTIFICATE (integer)

Examples

Once you've compiled PHP with CURL support, you can begin using the CURL functions. The basic idea behind the CURL functions is that you initialize a CURL session using the **curl_init()**, then you can set all your options for the transfer via the **curl_setopt()**, then you can execute the session with the **curl_exec()** and then you finish off your session using the **curl_close()**. Here is an example that uses the CURL functions to fetch the example.com homepage into a file:

Example 1. Using PHP's CURL module to fetch the example.com homepage

```
<?php

$ch = curl_init ("http://www.example.com/");
$fp = fopen ("example_homepage.txt", "w");

curl_setopt ($ch, CURLOPT_FILE, $fp);
curl_setopt ($ch, CURLOPT_HEADER, 0);

curl_exec ($ch);
curl_close ($ch);
fclose ($fp);
?>
```

Table of Contents

curl_close

(PHP 4 >= 4.0.2)

curl_close -- Close a CURL session

Description

void **curl_close** (resource ch)

This function closes a CURL session and frees all resources. The CURL handle, *ch*, is also deleted.

curl_errno

(PHP 4 >= 4.0.3)

curl_errno -- Return an integer containing the last error number

Description

int **curl_errno** (resource ch)

Warning
This function is currently not documented; only the argument list is available.

curl_error

(PHP 4 >= 4.0.3)

curl_error -- Return a string containing the last error for the current session

Description

string **curl_error** (resource ch)

Warning
This function is currently not documented; only the argument list is available.

curl_exec

(PHP 4 >= 4.0.2)

curl_exec -- Perform a CURL session

Description

bool **curl_exec** (resource ch)

This function should be called after you initialize a CURL session and all the options for the session are set. Its purpose is simply to execute the predefined CURL session (given by the *ch*).

> **Tip:** As with anything that outputs its result directly to the browser, you can use the output-control functions to capture the output of this function, and save it in a **string** (for example).

curl_getinfo

(PHP 4 >= 4.0.4)

curl_getinfo -- Get information regarding a specific transfer

Description

string **curl_getinfo** (resource ch [, int opt])

Returns information about the last transfer, *opt* may be one of the following:

- "CURLINFO_EFFECTIVE_URL" - Last effective URL
- "CURLINFO_HTTP_CODE" - Last received HTTP code
- "CURLINFO_FILETIME" - Remote time of the retrieved document, if -1 is returned the time of the document is unknown
- "CURLINFO_TOTAL_TIME" - Total transaction time in seconds for last transfer
- "CURLINFO_NAMELOOKUP_TIME" - Time in seconds until name resolving was complete
- "CURLINFO_CONNECT_TIME" - Time in seconds it took to establish the connection
- "CURLINFO_PRETRANSFER_TIME" - Time in seconds from start until just before file transfer begins

- "CURLINFO_STARTTRANSFER_TIME" - Time in seconds until the first byte is about to be transfered
- "CURLINFO_REDIRECT_TIME" - Time in seconds of all redirection steps before final transaction was started
- "CURLINFO_SIZE_UPLOAD" - Total number of bytes uploaded
- "CURLINFO_SIZE_DOWNLOAD" - Total number of bytes downloaded
- "CURLINFO_SPEED_DOWNLOAD" - Average download speed
- "CURLINFO_SPEED_UPLOAD" - Average upload speed
- "CURLINFO_HEADER_SIZE" - Total size of all headers received
- "CURLINFO_REQUEST_SIZE" - Total size of issued requests, currently only for HTTP requests
- "CURLINFO_SSL_VERIFYRESULT" - Result of SSL certification verification requested by setting CURLOPT_SSL_VERIFYPEER
- "CURLINFO_CONTENT_LENGTH_DOWNLOAD" - content-length of download, read from Content-Length: field
- "CURLINFO_CONTENT_LENGTH_UPLOAD" - Specified size of upload
- "CURLINFO_CONTENT_TYPE" - Content-type of downloaded object, NULL indicates server did not send valid Content-Type: header

If called without the optional parameter opt an assoctive array is returned with the following array elements which correspond to *opt* options:

- "url"
- "content_type"
- "http_encode"
- "header_size"
- "request_size"
- "filetime"
- "ssl_verify_result"
- "redirect_count"
- "total_time"
- "namelookup_time"
- "connect_time"
- "pretransfer_time"
- "size_upload"
- "size_download"
- "speed_download"
- "speed_upload"
- "download_content_length"
- "upload_content_length"
- "starttransfer_time"
- "redirect_time"

curl_init

(PHP 4 >= 4.0.2)

curl_init -- Initialize a CURL session

Description

resource **curl_init** ([string url])

The **curl_init()** will initialize a new session and return a CURL handle for use with the **curl_setopt()**, **curl_exec()**, and **curl_close()** functions. If the optional *url* parameter is supplied then the CURLOPT_URL option will be set to the value of the parameter. You can manually set this using the **curl_setopt()** function.

Example 1. Initializing a new CURL session and fetching a webpage

```
<?php
$ch = curl_init();

curl_setopt ($ch, CURLOPT_URL, "http://www.example.com/");
curl_setopt ($ch, CURLOPT_HEADER, 0);

curl_exec ($ch);

curl_close ($ch);
?>
```

See also: **curl_close()**, **curl_setopt()**

curl_setopt

(PHP 4 >= 4.0.2)

curl_setopt -- Set an option for a CURL transfer

Description

bool **curl_setopt** (resource ch, string option, mixed value)

The **curl_setopt()** function will set options for a CURL session identified by the *ch* parameter. The *option* parameter is the option you want to set, and the *value* is the value of the option given by the *option*.

The *value* should be a long for the following options (specified in the *option* parameter):

- *CURLOPT_INFILESIZE*: When you are uploading a file to a remote site, this option should be used to tell PHP what the expected size of the infile will be.
- *CURLOPT_VERBOSE*: Set this option to a non-zero value if you want CURL to report everything that is happening.
- *CURLOPT_HEADER*: Set this option to a non-zero value if you want the header to be included in the output.
- *CURLOPT_NOPROGRESS*: Set this option to a non-zero value if you don't want PHP to display a progress meter for CURL transfers.

  Note: PHP automatically sets this option to a non-zero parameter, this should only be changed for debugging purposes.
- *CURLOPT_NOBODY*: Set this option to a non-zero value if you don't want the body included with the output.
- *CURLOPT_FAILONERROR*: Set this option to a non-zero value if you want PHP to fail silently if the HTTP code returned is greater than 300. The default behavior is to return the page normally, ignoring the code.
- *CURLOPT_UPLOAD*: Set this option to a non-zero value if you want PHP to prepare for an upload.
- *CURLOPT_POST*: Set this option to a non-zero value if you want PHP to do a regular HTTP POST. This POST is a normal `application/x-www-form-urlencoded` kind, most commonly used by HTML forms.
- *CURLOPT_FTPLISTONLY*: Set this option to a non-zero value and PHP will just list the names of an FTP directory.
- *CURLOPT_FTPAPPEND*: Set this option to a non-zero value and PHP will append to the remote file instead of overwriting it.
- *CURLOPT_NETRC*: Set this option to a non-zero value and PHP will scan your `~./netrc` file to find your username and password for the remote site that you're establishing a connection with.
- *CURLOPT_FOLLOWLOCATION*: Set this option to a non-zero value to follow any "Location: " header that the server sends as a part of the HTTP header (note this is recursive, PHP will follow as many "Location: " headers that it is sent.)

- *CURLOPT_PUT*: Set this option to a non-zero value to HTTP PUT a file. The file to PUT must be set with the CURLOPT_INFILE and CURLOPT_INFILESIZE.
- *CURLOPT_MUTE*: Set this option to a non-zero value and PHP will be completely silent with regards to the CURL functions.
- *CURLOPT_TIMEOUT*: Pass a long as a parameter that contains the maximum time, in seconds, that you'll allow the CURL functions to take.
- *CURLOPT_LOW_SPEED_LIMIT*: Pass a long as a parameter that contains the transfer speed in bytes per second that the transfer should be below during CURLOPT_LOW_SPEED_TIME seconds for PHP to consider it too slow and abort.
- *CURLOPT_LOW_SPEED_TIME*: Pass a long as a parameter that contains the time in seconds that the transfer should be below the CURLOPT_LOW_SPEED_LIMIT for PHP to consider it too slow and abort.
- *CURLOPT_RESUME_FROM*: Pass a long as a parameter that contains the offset, in bytes, that you want the transfer to start from.
- *CURLOPT_SSLVERSION*: Pass a long as a parameter that contains the SSL version (2 or 3) to use. By default PHP will try and determine this by itself, although, in some cases you must set this manually.
- *CURLOPT_SSL_VERIFYHOST*: Pass a long if CURL should verify the Common name of the peer certificate in the SSL handshake. A value of 1 denotes that we should check for the existence of the common name, a value of 2 denotes that we should make sure it matches the provided hostname.
- *CURLOPT_TIMECONDITION*: Pass a long as a parameter that defines how the CURLOPT_TIMEVALUE is treated. You can set this parameter to TIMECOND_IFMODSINCE or TIMECOND_ISUNMODSINCE. This is a HTTP-only feature.
- *CURLOPT_TIMEVALUE*: Pass a long as a parameter that is the time in seconds since January 1st, 1970. The time will be used as specified by the CURLOPT_TIMEVALUE option, or by default the TIMECOND_IFMODSINCE will be used.
- *CURLOPT_RETURNTRANSFER*: Pass a non-zero value if you want CURL to directly return the transfer instead of printing it out directly.

The *value* parameter should be a string for the following values of the *option* parameter:

- *CURLOPT_URL*: This is the URL that you want PHP to fetch. You can also set this option when initializing a session with the **curl_init()** function.
- *CURLOPT_USERPWD*: Pass a string formatted in the [username]:[password] manner, for PHP to use for the connection.
- *CURLOPT_PROXYUSERPWD*: Pass a string formatted in the [username]:[password] format for connection to the HTTP proxy.
- *CURLOPT_RANGE*: Pass the specified range you want. It should be in the "X-Y" format, where X or Y may be left out. The HTTP transfers also support several intervals, separated with commas as in X-Y,N-M.
- *CURLOPT_POSTFIELDS*: Pass a string containing the full data to post in an HTTP "POST" operation.
- *CURLOPT_REFERER*: Pass a string containing the "referer" header to be used in an HTTP request.
- *CURLOPT_USERAGENT*: Pass a string containing the "user-agent" header to be used in an HTTP request.
- *CURLOPT_FTPPORT*: Pass a string containing the value which will be used to get the IP address to use for the ftp "POST" instruction. The POST instruction tells the remote server to connect to our specified IP address. The string may be a plain IP address, a hostname, a network interface name (under UNIX), or just a plain '-' to use the systems default IP address.
- *CURLOPT_COOKIE*: Pass a string containing the content of the cookie to be set in the HTTP header.
- *CURLOPT_SSLCERT*: Pass a string containing the filename of PEM formatted certificate.
- *CURLOPT_SSLCERTPASSWD*: Pass a string containing the password required to use the CURLOPT_SSLCERT certificate.
- *CURLOPT_COOKIEFILE*: Pass a string containing the name of the file containing the cookie data. The cookie file can be in Netscape format, or just plain HTTP-style headers dumped into a file.
- *CURLOPT_CUSTOMREQUEST*: Pass a string to be used instead of GET or HEAD when doing an HTTP request. This is useful for doing DELETE or other, more obscure, HTTP requests. Valid values are things like GET, POST, and so on; i.e. do not enter a whole HTTP request line here. For instance, entering 'GET /index.html HTTP/1.0\r\n\r\n' would be incorrect.

  Note: Don't do this without making sure your server supports the command first.
- *CURLOPT_PROXY*: Give the name of the HTTP proxy to tunnel requests through.
- *CURLOPT_INTERFACE*: Pass the name of the outgoing network interface to use. This can be an interface name, an IP address or a host name.
- *CURLOPT_KRB4LEVEL*: Pass the KRB4 (Kerberos 4) security level. Anyone of the following strings (in order from least powerful,

to most powerful): 'clear', 'safe', 'confidential', 'private'. If the string does not match one of these, then 'private' is used. If you set this to NULL, this disables KRB4 security. KRB4 security only works with FTP transactions currently.

- *CURLOPT_HTTPHEADER*: Pass an array of HTTP header fields to set.
- *CURLOPT_QUOTE*: Pass an array of FTP commands to perform on the server prior to the FTP request.
- *CURLOPT_POSTQUOTE*: Pass an array of FTP commands to execute on the server, after the FTP request has been performed.

The following options expect a file descriptor that is obtained by using the **fopen()** function:

- *CURLOPT_FILE*: The file where the output of your transfer should be placed, the default is STDOUT.
- *CURLOPT_INFILE*: The file where the input of your transfer comes from.
- *CURLOPT_WRITEHEADER*: The file to write the header part of the output into.
- *CURLOPT_STDERR*: The file to write errors to instead of stderr.

curl_version

(PHP 4 >= 4.0.2)

curl_version -- Return the current CURL version

Description

string **curl_version** (void)

The **curl_version()** function returns a string containing the current CURL version.

XIII. Cybercash payment functions

Installation

These functions are only available if the interpreter has been compiled with the `--with-cybercash=[DIR]`.

This extension has been removed from PHP as of PHP 4.3.0, if you have questions as to why then you will find the following CyberCash Faq helpful. In short, CyberCash was bought out by VeriSign and although the CyberCash service continues to exist, VeriSign encourages users to switch. See the above faq for details.

Table of Contents

cybercash_base64_decode

(PHP 4 <= 4.2.3)

cybercash_base64_decode -- base64 decode data for Cybercash

Description

string **cybercash_base64_decode** (string inbuff)

cybercash_base64_encode

(PHP 4 <= 4.2.3)

cybercash_base64_encode -- base64 encode data for Cybercash

Description

string **cybercash_base64_encode** (string inbuff)

cybercash_decr

(PHP 4 <= 4.2.3)

cybercash_decr -- Cybercash decrypt

Description

array **cybercash_decr** (string wmk, string sk, string inbuff)

The function returns an associative array with the elements "errcode" and, if "errcode" is **FALSE**, "outbuff" (string), "outLth" (long) and "macbuff" (string).

cybercash_encr

(PHP 4 <= 4.2.3)

cybercash_encr -- Cybercash encrypt

Description

array **cybercash_encr** (string wmk, string sk, string inbuff)

The function returns an associative array with the elements "errcode" and, if "errcode" is **FALSE**, "outbuff" (string), "outLth" (long) and "macbuff" (string).

XIV. Crédit Mutuel CyberMUT functions

Introduction

This extension allows you to process credit cards transactions using Crédit Mutuel CyberMUT system (http://www.creditmutuel.fr/centre_commercial/vendez_sur_internet.html).

CyberMUT is a popular Web Payment Service in France, provided by the Crédit Mutuel bank. If you are foreign in France, these functions will not be useful for you.

The use of these functions is almost identical to the original SDK functions, except for the parameters of return for **cybermut_creerformulairecm()** and **cybermut_creerreponsecm()**, which are returned directly by functions PHP, whereas they had passed in reference in the original functions.

These functions have been added in PHP 4.0.6.

> **Note:** These functions only provide a link to CyberMUT SDK. Be sure to read the CyberMUT Developers Guide for full details of the required parameters.

Installation

These functions are only available if PHP has been compiled with the `--with-cybermut[=DIR]` option, where DIR is the location of `libcm-mac.a` and `cm-mac.h` . You will require the appropriate SDK for your platform, which may be sent to you after your CyberMUT's subscription (contact them via Web, or go to the nearest Crédit Mutuel).

> **Note:** This extension is not available on Windows platforms.

Table of Contents

cybermut_creerformulairecm

(4.0.5 - 4.2.3 only)

cybermut_creerformulairecm -- Generate HTML form of request for payment

Description

string **cybermut_creerformulairecm** (string url_CM, string version, string TPE, string montant, string ref_commande, string texte_libre, string url_retour, string url_retour_ok, string url_retour_err, string langue, string code_societe, string texte_bouton)

cybermut_creerformulairecm() is used to generate the HTML form of request for payment.

Example 1. First step of payment (equiv cgi1.c)

```
<?php
// Directory where the keys are located
putenv("CMKEYDIR=/var/creditmut/cles");

// Version number
$VERSION="1.2";

  $retour =  cybermut_creerformulairecm(
  "https://www.creditmutuel.fr/test/telepaiement/paiement.cgi",
  $VERSION,
  "1234567890",
  "300FRF",
  $REFERENCE,
  $TEXTE_LIBRE,
  $URL_RETOUR,
  $URL_RETOUR_OK,
  $URL_RETOUR_ERR,
  "francais",
  "company",
  "Paiement par carte bancaire");

  echo $retour;
?>
```

See also **cybermut_testmac()** and **cybermut_creerreponsecm()**.

cybermut_creerreponsecm

(4.0.5 - 4.2.3 only)

cybermut_creerreponsecm -- Generate the acknowledgement of delivery of the confirmation of payment

Description

string **cybermut_creerreponsecm** (string phrase)

cybermut_creerreponsecm() returns a string containing delivery acknowledgement message.

The parameter is "OK" if the message of confirmation of the payment was correctly identified by **cybermut_testmac()**. Any other chain is regarded as an error message.

See also **cybermut_creerformulairecm()** and **cybermut_testmac()**.

cybermut_testmac

(4.0.5 - 4.2.3 only)

cybermut_testmac -- Make sure that there no was data diddling contained in the received message of confirmation

Description

bool **cybermut_testmac** (string code_MAC, string version, string TPE, string cdate, string montant, string ref_commande, string texte_libre, string code-retour)

cybermut_testmac() is used to make sure that there was not data diddling contained in the received message of confirmation. Pay attention to parameters *code-retour* and *texte-libre*, which cannot be evaluated as is, because of the dash. You must retrieve them by using:

```
<?php
  $code_retour = $_GET["code-retour"];
  $texte_libre = $_GET["texte-libre"];
?>
```

Example 1. Last step of payment (equiv cgi2.c)

```
<?php
// Make sure that Enable Track Vars is ON.
// Directory where are located the keys
putenv("CMKEYDIR=/var/creditmut/cles");

// Version number
$VERSION="1.2";

$texte_libre = $_GET["texte-libre"];
$code_retour = $_GET["code-retour"];

$mac_ok = cybermut_testmac($MAC,$VERSION,$TPE,$date,$montant,$reference,$texte_libre,$code_retour);

if ($mac_ok) {

  //
  // insert data processing here
  //
  //

  $result=cybermut_creerreponsecm("OK");
} else {
  $result=cybermut_creerreponsecm("Document Falsifie");
}

?>
```

See also **cybermut_creerformulairecm()** and **cybermut_creerreponsecm()**.

XV. Cyrus IMAP administration functions

Introduction

Warning
This function is currently not documented; only the argument list is available.

Note: This extension is not available on Windows platforms.

Installation

To enable Cyrus IMAP support and to use these functions you have to compile PHP with the `--with-cyrus` option.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

CYRUS_CONN_NONSYNCLITERAL (integer)

CYRUS_CONN_INITIALRESPONSE (integer)

CYRUS_CALLBACK_NUMBERED (integer)

CYRUS_CALLBACK_NOLITERAL (integer)

Table of Contents

cyrus_authenticate

(PHP 4 >= 4.1.0)

cyrus_authenticate -- Authenticate against a Cyrus IMAP server

Description

bool **cyrus_authenticate** (resource connection [, string mechlist [, string service [, string user [, int minssf [, int maxssf]]]]])

Warning
This function is currently not documented; only the argument list is available.

cyrus_bind

(PHP 4 >= 4.1.0)

cyrus_bind -- Bind callbacks to a Cyrus IMAP connection

Description

bool **cyrus_bind** (resource connection, array callbacks)

Warning
This function is currently not documented; only the argument list is available.

cyrus_close

(PHP 4 >= 4.1.0)

cyrus_close -- Close connection to a Cyrus IMAP server

Description

bool **cyrus_close** (resource connection)

Warning
This function is currently not documented; only the argument list is available.

cyrus_connect

(PHP 4 >= 4.1.0)

cyrus_connect -- Connect to a Cyrus IMAP server

Description

resource **cyrus_connect** ([string host [, string port [, int flags]]])

Warning
This function is currently not documented; only the argument list is available.

cyrus_query

(PHP 4 >= 4.1.0)

cyrus_query -- Send a query to a Cyrus IMAP server

Description

bool **cyrus_query** (resource connection, string query)

Warning
This function is currently not documented; only the argument list is available.

cyrus_unbind

(PHP 4 >= 4.1.0)

cyrus_unbind -- Unbind ...

Description

bool **cyrus_unbind** (resource connection, string trigger_name)

Warning
This function is currently not documented; only the argument list is available.

XVI. Character type functions

Introduction

The functions provided by this extension check whether a character or string falls into a certain character class according to the current locale (see also **setlocale()**).

When called with an integer argument these functions behave exactly like their C counterparts from `ctype.h`.

When called with a string argument they will check every character in the string and will only return **TRUE** if every character in the string matches the requested criteria. When called with an empty string the result will always be **TRUE**.

Passing anything else but a string or integer will return **FALSE** immediately.

Requirements

None besides functions from the standard C library which are always available.

Installation

Beginning with PHP 4.2.0 these functions are enabled by default. For older versions you have to configure and compile PHP with `--enable-ctype`. You can disable ctype support with `--disable-ctype`.

The windows version of PHP has built in support for this extension. You do not need to load any additional extension in order to use these functions.

Note: Builtin support for ctype is available with PHP 4.3.0.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Table of Contents

ctype_alnum

(PHP 4 >= 4.0.4)

ctype_alnum -- Check for alphanumeric character(s)

Description

bool **ctype_alnum** (string text)

Returns **TRUE** if every character in *text* is either a letter or a digit, **FALSE** otherwise. In the standard `C` locale letters are just `[A-Za-z]`. The function is equivalent to `(ctype_alpha($text) || ctype_digit($text))`.

See also **ctype_alpha()**, **ctype_digit()**, and **setlocale()**.

ctype_alpha

(PHP 4 >= 4.0.4)

ctype_alpha -- Check for alphabetic character(s)

Description

bool **ctype_alpha** (string text)

Returns **TRUE** if every character in *text* is a letter from the current locale, **FALSE** otherwise. In the standard `C` locale letters are just `[A-Za-z]` and **ctype_alpha()** is equivalent to `(ctype_upper($text) || ctype_lower($text))`, but other languages have letters that are considered neither upper nor lower case.

See also **ctype_upper()**, **ctype_lower()**, and **setlocale()**.

ctype_cntrl

(PHP 4 >= 4.0.4)

ctype_cntrl -- Check for control character(s)

Description

bool **ctype_cntrl** (string text)

Returns **TRUE** if every character in *text* has a special control function, **FALSE** otherwise. Control characters are e.g. line feed, tab, esc.

ctype_digit

(PHP 4 >= 4.0.4)

ctype_digit -- Check for numeric character(s)

Description

bool **ctype_digit** (string text)

Returns **TRUE** if every character in *text* is a decimal digit, **FALSE** otherwise.

See also **ctype_alnum()** and **ctype_xdigit()**.

ctype_graph

(PHP 4 >= 4.0.4)

ctype_graph -- Check for any printable character(s) except space

Description

bool **ctype_graph** (string text)

Returns **TRUE** if every character in *text* is printable and actually creates visible output (no white space), **FALSE** otherwise.

See also **ctype_alnum()**, **ctype_print()**, and **ctype_punct()**.

ctype_lower

(PHP 4 >= 4.0.4)

ctype_lower -- Check for lowercase character(s)

Description

bool **ctype_lower** (string text)

Returns **TRUE** if every character in *text* is a lowercase letter in the current locale.

See also **ctype_alpha()** and **ctype_upper()**.

ctype_print

(PHP 4 >= 4.0.4)

ctype_print -- Check for printable character(s)

Description

bool **ctype_print** (string text)

Returns **TRUE** if every character in *text* will actually create output (including blanks). Returns **FALSE** if *text* contains control characters or characters that do not have any output or control function at all.

See also **ctype_cntrl()**, **ctype_graph()**, and **ctype_punct()**.

ctype_punct

(PHP 4 >= 4.0.4)

ctype_punct -- Check for any printable character which is not whitespace or an alphanumeric character

Description

bool **ctype_punct** (string text)

Returns **TRUE** if every character in *text* is printable, but neither letter, digit or blank, **FALSE** otherwise.

See also **ctype_cntrl()**, **ctype_graph()**, and **ctype_punct()**.

ctype_space

(PHP 4 >= 4.0.4)

ctype_space -- Check for whitespace character(s)

Description

bool **ctype_space** (string text)

Returns **TRUE** if every character in *text* creates some sort of white space, **FALSE** otherwise. Besides the blank character this also includes tab, vertical tab, line feed, carriage return and form feed characters.

ctype_upper

(PHP 4 >= 4.0.4)

ctype_upper -- Check for uppercase character(s)

Description

bool **ctype_upper** (string text)

Returns **TRUE** if every character in *text* is a uppercase letter in the current locale.

See also **ctype_alpha()** and **ctype_lower()**.

ctype_xdigit

(PHP 4 >= 4.0.4)

ctype_xdigit -- Check for character(s) representing a hexadecimal digit

Description

bool **ctype_xdigit** (string text)

Returns **TRUE** if every character in *text* is a hexadecimal 'digit', that is a decimal digit or a character from `[A-Fa-f]` , **FALSE** otherwise.

See also **ctype_digit()**.

XVII. Database (dbm-style) abstraction layer functions

Introduction

These functions build the foundation for accessing Berkeley DB style databases.

This is a general abstraction layer for several file-based databases. As such, functionality is limited to a common subset of features supported by modern databases such as Sleepycat Software's DB2. (This is not to be confused with IBM's DB2 software, which is supported through the ODBC functions.)

Requirements

The behaviour of various aspects depends on the implementation of the underlying database. Functions such as **dba_optimize()** and **dba_sync()** will do what they promise for one database and will do nothing for others. You have to download and install supported dba-Handlers.

Table 1. List of DBA handlers

Handler	Notes
dbm	Dbm is the oldest (original) type of Berkeley DB style databases. You should avoid it, if possible. We do not support the compatibility functions built into DB2 and gdbm, because they are only compatible on the source code level, but cannot handle the original dbm format.
ndbm	Ndbm is a newer type and more flexible than dbm. It still has most of the arbitrary limits of dbm (therefore it is deprecated).
gdbm	Gdbm is the GNU database manager.
db2	DB2 is Sleepycat Software's DB2. It is described as "a programmatic toolkit that provides high-performance built-in database support for both standalone and client/server applications.
db3	DB3 is Sleepycat Software's DB3.
db4	DB4 is Sleepycat Software's DB4. This is available since PHP 5.0.0.
cdb	Cdb is "a fast, reliable, lightweight package for creating and reading constant databases." It is from the author of qmail and can be found here. Since it is constant, we support only reading operations. And since PHP 4.3.0 we support writing (not updating) through the internal cdb library.
cdb_make	Since PHP 4.3.0 we support creation (not updating) of cdb files when the bundeled cdb library is used.
flatfile	This is available since PHP 4.3.0 for compatibility with the deprecated dbm extension only and should be avoided. However you may use this where files were created in this format. That happens when configure could not find any external library.

When invoking the **dba_open()** or **dba_popen()** functions, one of the handler names must be supplied as an argument. The actually available list of handlers is displayed by invoking **phpinfo()** or **dba_handlers()**.

Installation

By using the `--enable-dba=shared` configuration option you can build a dynamic loadable modul to enable PHP for basic support of dbm-style databases. You also have to add support for at least one of the following handlers by specifying the `--with-XXXX` configure switch to your PHP configure line.

Table 2. Supported DBA handlers

Handler	Configure Switch
dbm	To enable support for dbm add `--with-dbm[=DIR]`.
ndbm	To enable support for ndbm add `--with-ndbm[=DIR]`.
gdbm	To enable support for gdbm add `--with-gdbm[=DIR]`.
db2	To enable support for db2 add `--with-db2[=DIR]`. **Note:** db2 conflicts with db3 and db4.
db3	To enable support for db3 add `--with-db3[=DIR]`. **Note:** db3 conflicts with db2 and db4.
db4	To enable support for db4 add `--with-db4[=DIR]`. **Note:** db4 conflicts with db2 and db3. **Note:** This was added in PHP 5.0.0. In earlier version you need to use `--with-db3=DIR` with DIR being the path to db4 librarie. It is not possible to use db versions starting from 4.1 with PHP prior to version 4.3.0.
cdb	To enable support for cdb add `--with-cdb[=DIR]`. **Note:** Since PHP 4.3.0 you can omit DIR to use the bundeled cdb library that adds the cdb_make handler which allows creation of cdb files and allows to access cdb files on the network using php's streams.
flatfile	To enable support for flatfile add `--with-flatfile`. **Note:** This was added in PHP 4.3.0 to add compatibility with deprecated dbm extension. Uee this handler only when you cannot install one of the libraries required by the other handlers and when you cannot use bundeled cdb handler.

Note: Up to PHP 4.3.0 you are able to add both db2 and db3 handler but only one of them can be used internally. That means that you cannot have both file formats. Starting with PHP 5.0.0 there is a configuration check avoid such missconfigurations.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

The functions **dba_open()** and **dba_popen()** return a handle to the specified database file to access which is used by all other dba-function calls.

Predefined Constants

This extension has no constants defined.

Examples

Example 1. DBA example

```
<?php
$id = dba_open ("/tmp/test.db", "n", "db2");
```

```
if (!$id) {
    echo "dba_open failed\n";
    exit;
}

dba_replace ("key", "This is an example!", $id);

if (dba_exists ("key", $id)) {
    echo dba_fetch ("key", $id);
    dba_delete ("key", $id);
}

dba_close ($id);
?>
```

DBA is binary safe and does not have any arbitrary limits. However, it inherits all limits set by the underlying database implementation.

All file-based databases must provide a way of setting the file mode of a new created database, if that is possible at all. The file mode is commonly passed as the fourth argument to **dba_open()** or **dba_popen()**.

You can access all entries of a database in a linear way by using the **dba_firstkey()** and **dba_nextkey()** functions. You may not change the database while traversing it.

Example 2. Traversing a database

```
<?php

// ...open database...

$key = dba_firstkey ($id);

while ($key != false) {
    if (...) {          // remember the key to perform some action later
        $handle_later[] = $key;
    }
    $key = dba_nextkey ($id);
}

for ($i = 0; $i < count($handle_later); $i++)
    dba_delete ($handle_later[$i], $id);

?>
```

Table of Contents

dba_close

(PHP 3>= 3.0.8, PHP 4)

dba_close -- Close database

Description

void **dba_close** (resource handle)

dba_close() closes the established database and frees all resources specified by *handle*.

handle is a database handle returned by **dba_open()**.

dba_close() does not return any value.

See also: dba_open() and dba_popen()

dba_delete

(PHP 3>= 3.0.8, PHP 4)

dba_delete -- Delete entry specified by key

Description

bool **dba_delete** (string key, resource handle)

dba_delete() deletes the entry specified by *key* from the database specified with *handle*.

key is the key of the entry which is deleted.

handle is a database handle returned by dba_open().

dba_delete() returns **TRUE** or **FALSE**, if the entry is deleted or not deleted, respectively.

See also: dba_exists(), dba_fetch(), dba_insert(), and dba_replace().

dba_exists

(PHP 3>= 3.0.8, PHP 4)

dba_exists -- Check whether key exists

Description

bool **dba_exists** (string key, resource handle)

dba_exists() checks whether the specified *key* exists in the database specified by *handle*.

Key is the key the check is performed for.

Handle is a database handle returned by dba_open().

dba_exists() returns **TRUE** or **FALSE**, if the key is found or not found, respectively.

See also: dba_fetch(), dba_delete(), dba_insert(), and dba_replace().

dba_fetch

(PHP 3>= 3.0.8, PHP 4)

dba_fetch -- Fetch data specified by key

Description

string **dba_fetch** (string key [, int skip, resource handle])

dba_fetch() fetches the data specified by *key* from the database specified with *handle*.

Key is the key the data is specified by.

Skip is the number of key-value pairs to ignore when using cdb databases. This value is ignored for all other databases which do not support multiple keys with the same name.

Handle is a database handle returned by dba_open().

dba_fetch() returns the associated string or **FALSE**, if the key/data pair is found or not found, respectively.

See also: dba_exists(), dba_delete(), dba_insert(), and dba_replace().

Note: The skip parameter is available since PHP 4.3 to support cdb's capability of multiple keys having the same name.

dba_firstkey

(PHP 3>= 3.0.8, PHP 4)

dba_firstkey -- Fetch first key

Description

string **dba_firstkey** (resource handle)

dba_firstkey() returns the first key of the database specified by *handle* and resets the internal key pointer. This permits a linear search through the whole database.

Handle is a database handle returned by **dba_open()**.

dba_firstkey() returns the key or **FALSE** depending on whether it succeeds or fails, respectively.

See also: **dba_nextkey()** and example 2 in the DBA examples

dba_handlers

(PHP 4 >= 4.3.0)

dba_handlers -- List handlers available

Description

array **dba_handlers** (void)

dba_handlers() returns an array with all handlers suppoerted by this extension.

When the internal cdb library is used you will see 'cdb' and 'cdb_make'.

dba_insert

(PHP 3>= 3.0.8, PHP 4)

dba_insert -- Insert entry

Description

bool **dba_insert** (string key, string value, resource handle)

dba_insert() inserts the entry described with *key* and *value* into the database specified by *handle*. It fails, if an entry with the same *key* already exists.

key is the key of the entry to be inserted.

value is the value to be inserted.

handle is a database handle returned by **dba_open()**.

dba_insert() returns **TRUE** or **FALSE**, depending on whether it succeeds of fails, respectively.

See also: **dba_exists()** **dba_delete()** **dba_fetch()** **dba_replace()**

dba_list

(PHP 4 >= 4.3.0)

dba_list -- List all open database files

Description

array **dba_list** (void)

dba_list() returns an associative array with all open database files. This array is in the form: `resourceid=>filename`.

dba_nextkey

(PHP 3>= 3.0.8, PHP 4)

dba_nextkey -- Fetch next key

Description

string **dba_nextkey** (resource handle)

dba_nextkey() returns the next key of the database specified by *handle* and advances the internal key pointer.

handle is a database handle returned by **dba_open()**.

dba_nextkey() returns the key or **FALSE** depending on whether it succeeds or fails, respectively.

See also: **dba_firstkey()** and example 2 in the DBA examples

dba_open

(PHP 3>= 3.0.8, PHP 4)

dba_open -- Open database

Description

resource **dba_open** (string path, string mode, string handler [, ...])

dba_open() establishes a database instance for *path* with *mode* using *handler*.

path is commonly a regular path in your filesystem.

mode is "r" for read access, "w" for read/write access to an already existing database, "c" for read/write access and database creation if it doesn't currently exist, and "n" for create, truncate and read/write access. Additional you can set the databse lock method with the next char. Use "l" to lock the database with an .lck file or "d" to lock the databasefile itself. It is important that all of your applications do this consistently. If you want to test the access and do not want to wait for the lock you can add "t" as third character. When you are absolutly sure that you do not require database locking you can do so by using "-" instead of "l" or "d". When none of "d", "l" or "-" is used dba will lock on the database file as it would with "d".

handler is the name of the handler which shall be used for accessing *path*. It is passed all optional parameters given to **dba_open()** and can act on behalf of them.

dba_open() returns a positive handle or **FALSE**, in the case the database was opened successfull or fails, respectively.

> **Note:** There can only be one writer for one database file. When you use dba on a webserver and more than one request requires write operations they can only be done one after another. Also read during write is not allowed. The dba extension uses locks to prevent this. See the follwoing table:

Table 1. DBA locking

already open	*mode* = "rl"	*mode* = "rlt"	*mode* = "wl"	*mode* = "wlt"	*mode* = "rd"	*mode* = "rdt"	*mode* = "wd"	*mode* = "wdt"
not open	ok	ok	ok	ok	ok	ok	ok	ok
mode = "rl"	ok	ok	wait	false	illegal	illegal	illegal	illegal
mode = "wl"	wait	false	wait	false	illegal	illegal	illegal	illegal
mode = "rd"	illegal	illegal	illegal	illegal	ok	ok	wait	false

already open	*mode* = "rl"	*mode* = "rlt"	*mode* = "wl"	*mode* = "wlt"	*mode* = "rd"	*mode* = "rdt"	*mode* = "wd"	*mode* = "wdt"
mode = "wd"	illegal	illegal	illegal	illegal	wait	false	wait	false

ok: the second call will be successfull.
wait: the second call waits until **dba_close()** is called for the first.
false: the second call returns false.
illegal: you must not mix "l" and "d" modifiers for *mode* parameter.

Note: Since PHP 4.3.0 it is possible to open database files over network connection. However in cases a socket connection will be used (as with http or ftp) the connection will be locked instead of the resource itself. This is important to know since in such cases locking is simply ignored on the resource and other solutions have to be found.

Note: Locking and the *mode* modifiers "l", "d", "-" and "t" were added in PHP 4.3.0. In PHP versions before PHP 4.3.0 you must use semaphores to guard against simultanious database access for any database handler with the exception of GDBM. See System V semaphore support.

See also: **dba_popen()** **dba_close()**

dba_optimize

(PHP 3>= 3.0.8, PHP 4)

dba_optimize -- Optimize database

Description

bool **dba_optimize** (resource handle)

dba_optimize() optimizes the underlying database specified by *handle*.

handle is a database handle returned by **dba_open()**.

dba_optimize() returns **TRUE** or **FALSE**, if the optimization succeeds or fails, respectively.

See also: **dba_sync()**

dba_popen

(PHP 3>= 3.0.8, PHP 4)

dba_popen -- Open database persistently

Description

resource **dba_popen** (string path, string mode, string handler [, ...])

dba_popen() establishes a persistent database instance for *path* with *mode* using *handler*.

path is commonly a regular path in your filesystem.

mode is "r" for read access, "w" for read/write access to an already existing database, "c" for read/write access and database creation if it doesn't currently exist, and "n" for create, truncate and read/write access.

handler is the name of the handler which shall be used for accessing *path*. It is passed all optional parameters given to **dba_popen()** and can act on behalf of them.

dba_popen() returns a positive handle or **FALSE**, in the case the open is successful or fails, respectively.

See also: **dba_open()** **dba_close()**

dba_replace

(PHP 3>= 3.0.8, PHP 4)

dba_replace -- Replace or insert entry

Description

bool **dba_replace** (string key, string value, resource handle)

dba_replace() replaces or inserts the entry described with *key* and *value* into the database specified by *handle*.

key is the key of the entry to be inserted.

value is the value to be inserted.

handle is a database handle returned by **dba_open()**.

dba_replace() returns **TRUE** or **FALSE**, depending on whether it succeeds of fails, respectively.

See also: **dba_exists()**, **dba_delete()**, **dba_fetch()**, and **dba_insert()**.

dba_sync

(PHP 3>= 3.0.8, PHP 4)

dba_sync -- Synchronize database

Description

bool **dba_sync** (resource handle)

dba_sync() synchronizes the database specified by *handle*. This will probably trigger a physical write to disk, if supported.

handle is a database handle returned by **dba_open()**.

dba_sync() returns **TRUE** or **FALSE**, if the synchronization succeeds or fails, respectively.

See also: **dba_optimize()**

XVIII. Date and Time functions

Introduction

These functions allow you to get the date and time from the server where your PHP scripts are running. You can use these functions to format the date and time in many different ways.

> **Note:** Please keep in mind that these functions are dependent on the locale settings of your server. Make sure to take daylight saving time and leap years into consideration when working with these functions.

Requirements

No external libraries are needed to build this extension.

Installation

There is no installation needed to use these functions; they are part of the PHP core.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Table of Contents

checkdate

(PHP 3, PHP 4)

checkdate -- Validate a gregorian date

Description

bool **checkdate** (int month, int day, int year)

Returns **TRUE** if the date given is valid; otherwise returns **FALSE**. Checks the validity of the date formed by the arguments. A date is considered valid if:

- year is between 1 and 32767 inclusive
- month is between 1 and 12 inclusive
- *Day* is within the allowed number of days for the given *month*. Leap *year*s are taken into consideration.

See also **mktime()** and **strtotime()**.

date

(PHP 3, PHP 4)

date -- Format a local time/date

Description

string **date** (string format [, int timestamp])

Returns a string formatted according to the given format string using the given integer *timestamp* or the current local time if no timestamp is given. In otherwords, *timestamp* is optional and defaults to the value of **time()**.

> **Note:** The valid range of a timestamp is typically from Fri, 13 Dec 1901 20:45:54 GMT to Tue, 19 Jan 2038 03:14:07 GMT.

(These are the dates that correspond to the minimum and maximum values for a 32-bit signed integer). On windows this range is limited from 01-01-1970 to 19-01-2038.

To generate a timestamp from a string representation of the date, you may be able to use **strtotime()**. Additionally, some databases have functions to convert their date formats into timestamps (such as MySQL's UNIX_TIMESTAMP function).

Table 1. The following characters are recognized in the *format* parameter string

format character	Description	Example returned values
a	Lowercase Ante meridiem and Post meridiem	am or pm
A	Uppercase Ante meridiem and Post meridiem	AM or PM
B	Swatch Internet time	000 through 999
d	Day of the month, 2 digits with leading zeros	01 to 31
D	A textual representation of a week, three letters	Mon through Sun
F	A full textual representation of a month, such as January or March	January through December
g	12-hour format of an hour without leading zeros	1 through 12
G	24-hour format of an hour without leading zeros	0 through 23
h	12-hour format of an hour with leading zeros	01 through 12
H	24-hour format of an hour with leading zeros	00 through 23
i	Minutes with leading zeros	00 to 59
I (capital i)	Whether or not the date is in daylights savings time	1 if Daylight Savings Time, 0 otherwise.
j	Day of the month without leading zeros	1 to 31
l (lowercase 'L')	A full textual representation of the day of the week	Sunday through Saturday
L	Whether it's a leap year	1 if it is a leap year, 0 otherwise.
m	Numeric representation of a month, with leading zeros	01 through 12
M	A short textual representation of a month, three letters	Jan through Dec
n	Numeric representation of a month, without leading zeros	1 through 12
O	Difference to Greenwich time (GMT) in hours	Example: +0200
r	RFC 822 formatted date	Example: Thu, 21 Dec 2000 16:01:07 +0200
s	Seconds, with leading zeros	00 through 59
S	English ordinal suffix for the day of the month, 2 characters	st, nd, rd or th. Works well with j
t	Number of days in the given month	28 through 31
T	Timezone setting of this machine	Examples: EST, MDT ...
U	Seconds since the Unix Epoch (January 1 1970 00:00:00 GMT)	See also **time()**
w	Numeric representation of the day of the week	0 (for Sunday) through 6 (for Saturday)
W	ISO-8601 week number of year, weeks starting on Monday (added in PHP 4.1.0)	Example: 42 (the 42nd week in the year)
Y	A full numeric representation of a year, 4 digits	Examples: 1999 or 2003
y	A two digit representation of a year	Examples: 99 or 03
z	The day of the year	0 through 366
Z	Timezone offset in seconds. The offset for timezones west of UTC is always negative, and for those east of UTC is always positive.	-43200 through 43200

Unrecognized characters in the format string will be printed as-is. The Z format will always return 0 when using **gmdate()**.

Example 1. date() examples

```
<?php
// Prints something like: Wednesday
echo date("l");

// Prints something like: Wednesday 15th of January 2003 05:51:38 AM
```

```
echo date ("l dS of F Y h:i:s A");

// Prints: July 1, 2000 is on a Saturday
echo "July 1, 2000 is on a " . date ("l", mktime(0,0,0,7,1,2000));
?>
```

You can prevent a recognized character in the format string from being expanded by escaping it with a preceding backslash. If the character with a backslash is already a special sequence, you may need to also escape the backslash.

Example 2. Escaping characters in date()

```
<?php
// prints something like: Wednesday the 15th
echo date("l \\t\h\e jS");
?>
```

It is possible to use **date()** and **mktime()** together to find dates in the future or the past.

Example 3. date() and mktime() example

```
<?php
$tomorrow  = mktime (0,0,0,date("m")  ,date("d")+1,date("Y"));
$lastmonth = mktime (0,0,0,date("m")-1,date("d"),  date("Y"));
$nextyear  = mktime (0,0,0,date("m"),  date("d"),  date("Y")+1);
?>
```

> **Note:** This can be more reliable than simply adding or subtracting the number of seconds in a day or month to a timestamp because of daylight savings time.

Some examples of **date()** formatting. Note that you should escape any other characters, as any which currently have a special meaning will produce undesirable results, and other characters may be assigned meaning in future PHP versions. When escaping, be sure to use single quotes to prevent characters like \n from becoming newlines.

Example 4. date() Formatting

```
<?php
// Assuming today is: March 10th, 2001, 5:16:18 pm

$today = date("F j, Y, g:i a");                 // March 10, 2001, 5:16 pm
$today = date("m.d.y");                         // 03.10.01
$today = date("j, n, Y");                       // 10, 3, 2001
$today = date("Ymd");                           // 20010310
$today = date('h-i-s, j-m-y, it is w Day z ');  // 05-16-17, 10-03-01, 1631 1618 6 Fripm01
$today = date('\i\t \i\s \t\h\e jS \d\a\y.');   // It is the 10th day.
$today = date("D M j G:i:s T Y");               // Sat Mar 10 15:16:08 MST 2001
$today = date('H:m:s \m \i\s\ \m\o\n\t\h');     // 17:03:17 m is month
$today = date("H:i:s");                         // 17:16:17
?>
```

To format dates in other languages, you should use the **setlocale()** and **strftime()** functions.

See also **getlastmod()**, **gmdate()**, **mktime()**, **strftime()** and **time()**.

getdate

(PHP 3, PHP 4)

getdate -- Get date/time information

Description

array **getdate** ([int timestamp])

Returns an associative array containing the date information of the *timestamp*, or the current local time if no timestamp is given, as the following array elements:

- "seconds" - seconds
- "minutes" - minutes
- "hours" - hours
- "mday" - day of the month
- "wday" - day of the week, numeric : from o as Sunday up to 6 as Saturday

- "mon" - month, numeric
- "year" - year, numeric
- "yday" - day of the year, numeric; i.e. "299"
- "weekday" - day of the week, textual, full; i.e. "Friday"
- "month" - month, textual, full; i.e. "January"

Example 1. getdate() example

```
$today = getdate();
$month = $today['month'];
$mday = $today['mday'];
$year = $today['year'];
echo "$month $mday, $year";
```

gettimeofday

(PHP 3>= 3.0.7, PHP 4)

gettimeofday -- Get current time

Description

array **gettimeofday** (void)

This is an interface to gettimeofday(2). It returns an associative array containing the data returned from the system call.

- "sec" - seconds
- "usec" - microseconds
- "minuteswest" - minutes west of Greenwich
- "dsttime" - type of dst correction

gmdate

(PHP 3, PHP 4)

gmdate -- Format a GMT/CUT date/time

Description

string **gmdate** (string format [, int timestamp])

Identical to the **date()** function except that the time returned is Greenwich Mean Time (GMT). For example, when run in Finland (GMT +0200), the first line below prints "Jan 01 1998 00:00:00", while the second prints "Dec 31 1997 22:00:00".

Example 1. gmdate() example

```
echo date ("M d Y H:i:s", mktime (0,0,0,1,1,1998));
echo gmdate ("M d Y H:i:s", mktime (0,0,0,1,1,1998));
```

See also **date()**, **mktime()**, **gmmktime()** and **strftime()**.

gmmktime

(PHP 3, PHP 4)

gmmktime -- Get UNIX timestamp for a GMT date

Description

int **gmmktime** (int hour, int minute, int second, int month, int day, int year [, int is_dst])

Identical to **mktime()** except the passed parameters represents a GMT date.

gmstrftime

(PHP 3>= 3.0.12, PHP 4)

gmstrftime -- Format a GMT/CUT time/date according to locale settings

Description

string **gmstrftime** (string format [, int timestamp])

Behaves the same as **strftime()** except that the time returned is Greenwich Mean Time (GMT). For example, when run in Eastern Standard Time (GMT -0500), the first line below prints "Dec 31 1998 20:00:00", while the second prints "Jan 01 1999 01:00:00".

Example 1. gmstrftime() example

```
setlocale (LC_TIME, 'en_US');
echo strftime ("%b %d %Y %H:%M:%S", mktime (20,0,0,12,31,98))."\n";
echo gmstrftime ("%b %d %Y %H:%M:%S", mktime (20,0,0,12,31,98))."\n";
```

See also **strftime()**.

localtime

(PHP 4)

localtime -- Get the local time

Description

array **localtime** ([int timestamp [, bool is_associative]])

The **localtime()** function returns an array identical to that of the structure returned by the C function call. The first argument to **localtime()** is the timestamp, if this is not given the current time is used. The second argument to the **localtime()** is the *is_associative*, if this is set to 0 or not supplied than the array is returned as a regular, numerically indexed array. If the argument is set to 1 then **localtime()** is an associative array containing all the different elements of the structure returned by the C function call to localtime. The names of the different keys of the associative array are as follows:

- "tm_sec" - seconds
- "tm_min" - minutes
- "tm_hour" - hour
- "tm_mday" - day of the month
- "tm_mon" - month of the year, starting with 0 for January
- "tm_year" - Years since 1900
- "tm_wday" - Day of the week
- "tm_yday" - Day of the year
- "tm_isdst" - Is daylight savings time in effect

microtime

(PHP 3, PHP 4)

microtime -- Return current UNIX timestamp with microseconds

Description

string **microtime** (void)

Returns the string "msec sec" where sec is the current time measured in the number of seconds since the Unix Epoch (0:00:00 January 1, 1970 GMT), and msec is the microseconds part. This function is only available on operating systems that support the gettimeofday() system call.

Both portions of the string are returned in units of seconds.

Example 1. microtime() example

```
function getmicrotime(){
    list($usec, $sec) = explode(" ",microtime());
    return ((float)$usec + (float)$sec);
    }

$time_start = getmicrotime();

for ($i=0; $i < 1000; $i++){
    //do nothing, 1000 times
    }

$time_end = getmicrotime();
$time = $time_end - $time_start;

echo "Did nothing in $time seconds";
```

See also **time()**.

mktime

(PHP 3, PHP 4)

mktime -- Get UNIX timestamp for a date

Description

int **mktime** (int hour, int minute, int second, int month, int day, int year [, int is_dst])

Warning: Note the strange order of arguments, which differs from the order of arguments in a regular UNIX mktime() call and which does not lend itself well to leaving out parameters from right to left (see below). It is a common error to mix these values up in a script.

Returns the Unix timestamp corresponding to the arguments given. This timestamp is a long integer containing the number of seconds between the Unix Epoch (January 1 1970) and the time specified.

Arguments may be left out in order from right to left; any arguments thus omitted will be set to the current value according to the local date and time.

is_dst can be set to 1 if the time is during daylight savings time, 0 if it is not, or -1 (the default) if it is unknown whether the time is within daylight savings time or not. If it's unknown, PHP tries to figure it out itself. This can cause unexpected (but not incorrect) results.

> **Note:** *is_dst* was added in 3.0.10.

mktime() is useful for doing date arithmetic and validation, as it will automatically calculate the correct value for out-of-range input. For example, each of the following lines produces the string "Jan-01-1998".

Example 1. mktime() example

```
echo date ("M-d-Y", mktime (0,0,0,12,32,1997));
echo date ("M-d-Y", mktime (0,0,0,13,1,1997));
echo date ("M-d-Y", mktime (0,0,0,1,1,1998));
echo date ("M-d-Y", mktime (0,0,0,1,1,98));
```

Year may be a two or four digit value, with values between 0-69 mapping to 2000-2069 and 70-99 to 1970-1999 (on systems where time_t is a 32bit signed integer, as most common today, the valid range for *year* is somewhere between 1902 and 2037).

The last day of any given month can be expressed as the "0" day of the next month, not the -1 day. Both of the following examples will produce the string "The last day in Feb 2000 is: 29".

Example 2. Last day of next month

```
$lastday = mktime (0,0,0,3,0,2000);
echo strftime ("Last day in Feb 2000 is: %d", $lastday);

$lastday = mktime (0,0,0,4,-31,2000);
echo strftime ("Last day in Feb 2000 is: %d", $lastday);
```

Date with year, month and day equal to zero is considered illegal (otherwise it what be regarded as 30.11.1999, which would be strange behavior).

See also **date()** and **time()**.

strftime

(PHP 3, PHP 4)

strftime -- Format a local time/date according to locale settings

Description

string **strftime** (string format [, int timestamp])

Returns a string formatted according to the given format string using the given *timestamp* or the current local time if no timestamp is given. Month and weekday names and other language dependent strings respect the current locale set with **setlocale()**.

The following conversion specifiers are recognized in the format string:

- %a - abbreviated weekday name according to the current locale
- %A - full weekday name according to the current locale
- %b - abbreviated month name according to the current locale
- %B - full month name according to the current locale
- %c - preferred date and time representation for the current locale
- %C - century number (the year divided by 100 and truncated to an integer, range 00 to 99)
- %d - day of the month as a decimal number (range 01 to 31)
- %D - same as %m/%d/%y
- %e - day of the month as a decimal number, a single digit is preceded by a space (range ' 1' to '31')
- %g - like %G, but without the century.
- %G - The 4-digit year corresponding to the ISO week number (see %V). This has the same format and value as %Y, except that if the ISO week number belongs to the previous or next year, that year is used instead.
- %h - same as %b
- %H - hour as a decimal number using a 24-hour clock (range 00 to 23)
- %I - hour as a decimal number using a 12-hour clock (range 01 to 12)
- %j - day of the year as a decimal number (range 001 to 366)
- %m - month as a decimal number (range 01 to 12)
- %M - minute as a decimal number
- %n - newline character
- %p - either `am' or `pm' according to the given time value, or the corresponding strings for the current locale
- %r - time in a.m. and p.m. notation

- %R - time in 24 hour notation
- %S - second as a decimal number
- %t - tab character
- %T - current time, equal to %H:%M:%S
- %u - weekday as a decimal number [1,7], with 1 representing Monday

Warning
Sun Solaris seems to start with Sunday as 1 although ISO 9889:1999 (the current C standard) clearly specifies that it should be Monday.

- %U - week number of the current year as a decimal number, starting with the first Sunday as the first day of the first week
- %V - The ISO 8601:1988 week number of the current year as a decimal number, range 01 to 53, where week 1 is the first week that has at least 4 days in the current year, and with Monday as the first day of the week. (Use %G or %g for the year component that corresponds to the week number for the specified timestamp.)
- %W - week number of the current year as a decimal number, starting with the first Monday as the first day of the first week
- %w - day of the week as a decimal, Sunday being 0
- %x - preferred date representation for the current locale without the time
- %X - preferred time representation for the current locale without the date
- %y - year as a decimal number without a century (range 00 to 99)
- %Y - year as a decimal number including the century
- %Z - time zone or name or abbreviation
- %% - a literal `%' character

Note: Not all conversion specifiers may be supported by your C library, in which case they will not be supported by PHP's **strftime()**. This means that e.g. %e, %T, %R and %D (there might be more) will not work on Windows.

Example 1. strftime() locale examples

```
setlocale (LC_TIME, "C");
print (strftime ("%A in Finnish is "));
setlocale (LC_TIME, "fi_FI");
print (strftime ("%A, in French "));
setlocale (LC_TIME, "fr_FR");
print (strftime ("%A and in German "));
setlocale (LC_TIME, "de_DE");
print (strftime ("%A.\n"));
```

This example works if you have the respective locales installed in your system.

Note: %G and %V, which are based on ISO 8601:1988 week numbers can give unexpected (albiet correct) results if the numbering system is not thoroughly understood. See %V above and example below.

Example 2. ISO 8601:1988 week number example

```
<?php
/*     December 2002 / January 2003
ISOWk  M   Tu  W   Thu F   Sa  Su
----- ----------------------------
51     16  17  18  19  20  21  22
52     23  24  25  26  27  28  29
1      30  31   1   2   3   4   5
2       6   7   8   9  10  11  12
3      13  14  15  16  17  18  19   */

// Outputs: 12/28/2002 - %V,%G,%Y = 52,2002,2002
print "12/28/2002 - %V,%G,%Y = " . strftime("%V,%G,%Y",strtotime("12/28/2002")) . "\n";

// Outputs: 12/30/2002 - %V,%G,%Y = 1,2003,2002
print "12/30/2002 - %V,%G,%Y = " . strftime("%V,%G,%Y",strtotime("12/30/2002")) . "\n";

// Outputs: 1/3/2003 - %V,%G,%Y = 1,2003,2003
print "1/3/2003 - %V,%G,%Y = " . strftime("%V,%G,%Y",strtotime("1/3/2003")) . "\n";

// Outputs: 1/10/2003 - %V,%G,%Y = 2,2003,2003
```

```
print "1/10/2003 - %V,%G,%Y = " . strftime("%V,%G,%Y",strtotime("1/10/2003")) . "\n";


/*     December 2004 / January 2005
ISOWk  M   Tu  W   Thu F   Sa  Su
----- ---------------------------
51     13  14  15  16  17  18  19
52     20  21  22  23  24  25  26
53     27  28  29  30  31   1   2
1       3   4   5   6   7   8   9
2      10  11  12  13  14  15  16   */

// Outputs: 12/23/2004 - %V,%G,%Y = 52,2004,2004
print "12/23/2004 - %V,%G,%Y = " . strftime("%V,%G,%Y",strtotime("12/23/2004")) . "\n";

// Outputs: 12/31/2004 - %V,%G,%Y = 53,2004,2004
print "12/31/2004 - %V,%G,%Y = " . strftime("%V,%G,%Y",strtotime("12/31/2004")) . "\n";

// Outputs: 1/2/2005 - %V,%G,%Y = 53,2004,2005
print "1/2/2005 - %V,%G,%Y = " . strftime("%V,%G,%Y",strtotime("1/2/2005")) . "\n";

// Outputs: 1/3/2005 - %V,%G,%Y = 1,2005,2005
print "1/3/2005 - %V,%G,%Y = " . strftime("%V,%G,%Y",strtotime("1/3/2005")) . "\n";

?>
```

See also **setlocale()** and **mktime()** and the Open Group specification of **strftime()**.

strtotime

(PHP 3>= 3.0.12, PHP 4)

strtotime -- Parse about any English textual datetime description into a UNIX timestamp

Description

int **strtotime** (string time [, int now])

The function expects to be given a string containing an English date format and will try to parse that format into a UNIX timestamp relative to the timestamp given in *now*, or the current time if none is supplied. Upon failure, `-1` is returned.

Because **strtotime()** behaves according to GNU date syntax, have a look at the GNU manual page titled Date Input Formats. Described there is valid syntax for the *time* parameter.

Example 1. strtotime() examples

```
echo strtotime ("now"), "\n";
echo strtotime ("10 September 2000"), "\n";
echo strtotime ("+1 day"), "\n";
echo strtotime ("+1 week"), "\n";
echo strtotime ("+1 week 2 days 4 hours 2 seconds"), "\n";
echo strtotime ("next Thursday"), "\n";
echo strtotime ("last Monday"), "\n";
```

Example 2. Checking for failure

```
$str = 'Not Good';
if (($timestamp = strtotime($str)) === -1) {
    echo "The string ($str) is bogus";
} else {
    echo "$str == ". date('l dS of F Y h:i:s A',$timestamp);
}
```

> **Note:** The valid range of a timestamp is typically from Fri, 13 Dec 1901 20:45:54 GMT to Tue, 19 Jan 2038 03:14:07 GMT. (These are the dates that correspond to the minimum and maximum values for a 32-bit signed integer.)

time

(PHP 3, PHP 4)

time -- Return current UNIX timestamp

Description

int **time** (void)

Returns the current time measured in the number of seconds since the Unix Epoch (January 1 1970 00:00:00 GMT).

See also **date()**.

XIX. dBase functions

Introduction

These functions allow you to access records stored in dBase-format (dbf) databases.

There is no support for indexes or memo fields. There is no support for locking, too. Two concurrent webserver processes modifying the same dBase file will very likely ruin your database.

dBase files are simple sequential files of fixed length records. Records are appended to the end of the file and delete records are kept until you call **dbase_pack()**.

We recommend that you do not use dBase files as your production database. Choose any real SQL server instead; MySQL or Postgres are common choices with PHP. dBase support is here to allow you to import and export data to and from your web database, because the file format is commonly understood by Windows spreadsheets and organizers.

Installation

In order to enable the bundled dbase library and to use these functions, you must compile PHP with the `--enable-dbase` option.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Table of Contents

dbase_add_record

(PHP 3, PHP 4)

dbase_add_record -- Add a record to a dBase database

Description

bool **dbase_add_record** (int dbase_identifier, array record)

Adds the data in the *record* to the database. If the number of items in the supplied record isn't equal to the number of fields in the database, the operation will fail and **FALSE** will be returned.

dbase_close

(PHP 3, PHP 4)

dbase_close -- Close a dBase database

Description

bool **dbase_close** (int dbase_identifier)

Closes the database associated with *dbase_identifier*.

dbase_create

(PHP 3, PHP 4)

dbase_create -- Creates a dBase database

Description

int **dbase_create** (string filename, array fields)

The *fields* parameter is an array of arrays, each array describing the format of one field in the database. Each field consists of a name, a character indicating the field type, a length, and a precision.

The types of fields available are:

L

Boolean. These do not have a length or precision.

M

Memo. (Note that these aren't supported by PHP.) These do not have a length or precision.

D

Date (stored as YYYYMMDD). These do not have a length or precision.

N

Number. These have both a length and a precision (the number of digits after the decimal point).

C

String.

If the database is successfully created, a dbase_identifier is returned, otherwise **FALSE** is returned.

Example 1. Creating a dBase database file

```
// "database" name
$dbname = "/tmp/test.dbf";

// database "definition"
$def =
    array(
        array("date",     "D"),
        array("name",     "C",  50),
        array("age",      "N",   3, 0),
        array("email",    "C", 128),
        array("ismember", "L")
    );

// creation
if (!dbase_create($dbname, $def))
    print "<strong>Error!</strong>";
```

dbase_delete_record

(PHP 3, PHP 4)

dbase_delete_record -- Deletes a record from a dBase database

Description

bool **dbase_delete_record** (int dbase_identifier, int record)

Marks *record* to be deleted from the database. To actually remove the record from the database, you must also call **dbase_pack()**.

dbase_get_record_with_names

(PHP 3>= 3.0.4, PHP 4)

dbase_get_record_with_names -- Gets a record from a dBase database as an associative array

Description

array **dbase_get_record_with_names** (int dbase_identifier, int record)

Returns the data from *record* in an associative array. The array also includes an associative member named 'deleted' which is set to 1 if the record has been marked for deletion (see **dbase_delete_record()**).

Each field is converted to the appropriate PHP type, except:

- Dates are left as strings
- Integers that would have caused an overflow (> 32 bits) are returned as strings

dbase_get_record

(PHP 3, PHP 4)

dbase_get_record -- Gets a record from a dBase database

Description

array **dbase_get_record** (int dbase_identifier, int record)

Returns the data from *record* in an array. The array is indexed starting at 0, and includes an associative member named 'deleted' which is set to 1 if the record has been marked for deletion (see **dbase_delete_record()**.

Each field is converted to the appropriate PHP type, except:

- Dates are left as strings
- Integers that would have caused an overflow (> 32 bits) are returned as strings

dbase_numfields

(PHP 3, PHP 4)

dbase_numfields -- Find out how many fields are in a dBase database

Description

int **dbase_numfields** (int dbase_identifier)

Returns the number of fields (columns) in the specified database. Field numbers are between 0 and dbase_numfields($db)-1, while record numbers are between 1 and dbase_numrecords($db).

Example 1. Using dbase_numfields()

```
$rec = dbase_get_record($db, $recno);
$nf  = dbase_numfields($db);
for ($i=0; $i < $nf; $i++) {
    print $rec[$i]."<br>\n";
}
```

dbase_numrecords

(PHP 3, PHP 4)

dbase_numrecords -- Find out how many records are in a dBase database

Description

int **dbase_numrecords** (int dbase_identifier)

Returns the number of records (rows) in the specified database. Record numbers are between 1 and dbase_numrecords($db), while field numbers are between 0 and dbase_numfields($db)-1.

dbase_open

(PHP 3, PHP 4)

dbase_open -- Opens a dBase database

Description

int **dbase_open** (string filename, int flags)

The flags correspond to those for the open() system call. (Typically 0 means read-only, 1 means write-only, and 2 means read and write.)

Returns a dbase_identifier for the opened database, or FALSE if the database couldn't be opened.

> **Note:** When safe mode is enabled, PHP checks whether the files or directories you are about to operate on have the same UID (owner) as the script that is being executed.

dbase_pack

(PHP 3, PHP 4)

dbase_pack -- Packs a dBase database

Description

bool **dbase_pack** (int dbase_identifier)

Packs the specified database (permanently deleting all records marked for deletion using **dbase_delete_record()**).

dbase_replace_record

(PHP 3>= 3.0.11, PHP 4)

dbase_replace_record -- Replace a record in a dBase database

Description

bool **dbase_replace_record** (int dbase_identifier, array record, int dbase_record_number)

Replaces the data associated with the record *record_number* with the data in the *record* in the database. If the number of items in the supplied record is not equal to the number of fields in the database, the operation will fail and **FALSE** will be returned.

dbase_record_number is an integer which spans from 1 to the number of records in the database (as returned by **dbase_numrecords()**).

XX. DBM Functions [deprecated]

Introduction

These functions allow you to store records stored in a dbm-style database. This type of database (supported by the Berkeley DB, GDBM, and some system libraries, as well as a built-in flatfile library) stores key/value pairs (as opposed to the full-blown records supported by relational databases).

> **Note:** However, dbm support is deprecated and you are encourged to use the Database (dbm-style) abstraction layer functions instead.

Requirements

To use this functions you have to compile PHP with support for an underlying database. See the list of supported Databases.

Installation

In order to use these functions, you must compile PHP with dbm support by using the `--with-db` option. In addition you must ensure support for an underlying database or you can use some sytem libraries.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

The function **dbmopen()** returns an database identifier which is used by the other dbm-functions.

Predefined Constants

This extension has no constants defined.

Examples

Example 1. DBM example

```
$dbm = dbmopen ("lastseen", "w");
if (dbmexists ($dbm, $userid)) {
    $last_seen = dbmfetch ($dbm, $userid);
} else {
    dbminsert ($dbm, $userid, time());
}
do_stuff();
dbmreplace ($dbm, $userid, time());
dbmclose ($dbm);
```

Table of Contents

dblist

(PHP 3, PHP 4)

dblist -- Describes the DBM-compatible library being used

Description

string **dblist** (void)

Example 1. Getting the information on the command line

```
[marcus@zaphod marcus]$ php -r 'echo dblist();'
This is GDBM version 1.8.0, as of May 19, 1999.
```

dbmclose

(PHP 3, PHP 4)

dbmclose -- Closes a dbm database

Description

bool **dbmclose** (resource dbm_identifier)

Unlocks and closes the specified database.

dbmdelete

(PHP 3, PHP 4)

dbmdelete -- Deletes the value for a key from a DBM database

Description

bool **dbmdelete** (resource dbm_identifier, string key)

Deletes the value for *key* in the database.

Returns **FALSE** if the key didn't exist in the database.

dbmexists

(PHP 3, PHP 4)

dbmexists -- Tells if a value exists for a key in a DBM database

Description

bool **dbmexists** (resource dbm_identifier, string key)

Returns **TRUE** if there is a value associated with the *key*.

dbmfetch

(PHP 3, PHP 4)

dbmfetch -- Fetches a value for a key from a DBM database

Description

string **dbmfetch** (resource dbm_identifier, string key)

Returns the value associated with *key*.

dbmfirstkey

(PHP 3, PHP 4)

dbmfirstkey -- Retrieves the first key from a DBM database

Description

string **dbmfirstkey** (resource dbm_identifier)

Returns the first key in the database. Note that no particular order is guaranteed since the database may be built using a hash-table, which doesn't guarantee any ordering.

dbminsert

(PHP 3, PHP 4)

dbminsert -- Inserts a value for a key in a DBM database

Description

int **dbminsert** (resource dbm_identifier, string key, string value)

Adds the value to the database with the specified key.

Returns -1 if the database was opened read-only, 0 if the insert was successful, and 1 if the specified key already exists. (To replace the value, use **dbmreplace()**.)

dbmnextkey

(PHP 3, PHP 4)

dbmnextkey -- Retrieves the next key from a DBM database

Description

string **dbmnextkey** (resource dbm_identifier, string key)

Returns the next key after *key*. By calling **dbmfirstkey()** followed by successive calls to **dbmnextkey()** it is possible to visit every key/value pair in the dbm database. For example:

Example 1. Visiting every key/value pair in a DBM database

```
$key = dbmfirstkey ($dbm_id);
while ($key) {
    echo "$key = " . dbmfetch ($dbm_id, $key) . "\n";
    $key = dbmnextkey ($dbm_id, $key);
}
```

dbmopen

(PHP 3, PHP 4)

dbmopen -- Opens a DBM database

Description

resource **dbmopen** (string filename, string flags)

The first argument is the full-path filename of the DBM file to be opened and the second is the file open mode which is one of "r", "n", "c" or "w" for read-only, new (implies read-write, and most likely will truncate an already-existing database of the same name), create (implies read-write, and will not truncate an already-existing database of the same name) and read-write respectively.

Returns an identifier to be passed to the other DBM functions on success, or **FALSE** on failure.

If NDBM support is used, NDBM will actually create filename.dir and filename.pag files. GDBM only uses one file, as does the internal flat-file support, and Berkeley DB creates a `filename.db` file. Note that PHP does its own file locking in addition to any file locking that may be done by the DBM library itself. PHP does not delete the `.lck` files it creates. It uses these files simply as fixed inodes on which to do the file locking. For more information on DBM files, see your Unix man pages, or obtain GNU's GDBM.

> **Note:** When safe mode is enabled, PHP checks whether the files or directories you are about to operate on have the same UID (owner) as the script that is being executed.

dbmreplace

(PHP 3, PHP 4)

dbmreplace -- Replaces the value for a key in a DBM database

Description

int **dbmreplace** (resource dbm_identifier, string key, string value)

Replaces the value for the specified key in the database.

This will also add the key to the database if it didn't already exist.

XXI. dbx functions

Introduction

The dbx module is a database abstraction layer (db 'X', where 'X' is a supported database). The dbx functions allow you to access all supported databases using a single calling convention. The dbx-functions themselves do not interface directly to the databases, but interface to the modules that are used to support these databases.

Requirements

To be able to use a database with the dbx-module, the module must be either linked or loaded into PHP, and the database module must be supported by the dbx-module. Currently, following databases are supported, but others will follow:

- FrontBase (available from PHP 4.1.0).
- Microsoft SQL Server
- MySQL
- ODBC
- PostgreSQL
- Sybase-CT (available from PHP 4.2.0).
- Oracle (oci8) (available from PHP 4.3.0).

Documentation for adding additional database support to dbx can be found at http://www.guidance.nl/php/dbx/doc/.

Installation

In order to have these functions available, you must compile PHP with dbx support by using the `--enable-dbx` option and all options for the databases that will be used, e.g. for MySQL you must also specify `--with-mysql=[DIR]`. To get other supported databases to work with the dbx-module refer to their specific documentation.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. DBX Configuration Options

Name	Default	Changeable
dbx.colnames_case	"unchanged"	PHP_INI_SYSTEM

For further details and definition of the PHP_INI_* constants see **ini_set()**.

> **Note:** This ini-option is available available from PHP 4.3.0.

Here is a short explanation of the configuration directives.

`dbx.colnames_case` **string**

> Columns names can be returned "unchanged" or converted to "uppercase" or "lowercase". This directive can be overridden with a flag to **dbx_query()**.

Resource Types

There are two resource types used in the dbx module. The first one is the link-**object** for a database connection, the second a result-**object** which helds the result of a query.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

DBX_MYSQL (integer)

DBX_ODBC (integer)

DBX_PGSQL (integer)

DBX_MSSQL (integer)

DBX_FBSQL (integer)

DBX_OCI8 (integer) (available from PHP 4.3.0)

DBX_SYBASECT (integer)

DBX_PERSISTENT (integer)

DBX_RESULT_INFO (integer)

DBX_RESULT_INDEX (integer)

DBX_RESULT_ASSOC (integer)

DBX_COLNAMES_UNCHANGED (integer) (available from PHP 4.3.0)

DBX_COLNAMES_UPPERCASE (integer) (available from PHP 4.3.0)

DBX_COLNAMES_LOWERCASE (integer) (available from PHP 4.3.0)

DBX_CMP_NATIVE (integer)

DBX_CMP_TEXT (integer)

DBX_CMP_NUMBER (integer)

DBX_CMP_ASC (integer)

DBX_CMP_DESC (integer)

Table of Contents

dbx_close

(PHP 4 >= 4.0.6)

dbx_close -- Close an open connection/database

Description

bool **dbx_close** (object link_identifier)

Returns **TRUE** on success or **FALSE** on failure.

Example 1. dbx_close() example

```
<?php
$link = dbx_connect(DBX_MYSQL, "localhost", "db", "username", "password")
    or die ("Could not connect");

print("Connected successfully");
dbx_close($link);
?>
```

Note: Always refer to the module-specific documentation as well.

See also: **dbx_connect()**.

dbx_compare

(PHP 4 >= 4.1.0)

dbx_compare -- Compare two rows for sorting purposes

Description

int **dbx_compare** (array row_a, array row_b, string column_key [, int flags])

dbx_compare() returns `0` if the `row_a[$column_key]` is equal to `row_b[$column_key]`, and `1` or `-1` if the former is greater or is smaller than the latter one, respectively, or vice versa if the *`flag`* is set to **DBX_CMP_DESC**. **dbx_compare()** is a helper function for **dbx_sort()** to ease the make and use of the custom sorting function.

The *`flags`* can be set to specify comparison direction:

- **DBX_CMP_ASC** - ascending order
- **DBX_CMP_DESC** - descending order

and the preferred comparison type:

- **DBX_CMP_NATIVE** - no type conversion
- **DBX_CMP_TEXT** - compare items as strings
- **DBX_CMP_NUMBER** - compare items numerically

One of the direction and one of the type constant can be combined with bitwise OR operator (|). The default value for the *`flags`* parameter is **DBX_CMP_ASC** | **DBX_CMP_NATIVE**.

Example 1. dbx_compare() example

```
<?php
function user_re_order ($a, $b) {
    $rv = dbx_compare ($a, $b, "parentid", DBX_CMP_DESC);
    if ( !$rv ) {
        $rv = dbx_compare ($a, $b, "id", DBX_CMP_NUMBER);
    }
    return $rv;
}

$link   = dbx_connect (DBX_ODBC, "", "db", "username", "password")
    or die ("Could not connect");

$result = dbx_query ($link, "SELECT id, parentid, description FROM table ORDER BY id");
    // data in $result is now ordered by id

dbx_sort ($result, "user_re_order");
    // date in $result is now ordered by parentid (descending), then by id

dbx_close ($link);
?>
```

See also **dbx_sort()**.

dbx_connect

(PHP 4 >= 4.0.6)

dbx_connect -- Open a connection/database

Description

object **dbx_connect** (mixed module, string host, string database, string username, string password [, int persistent])

dbx_connect() returns an object on success, **FALSE** on error. If a connection has been made but the database could not be selected, the connection is closed and **FALSE** is returned. The *persistent* parameter can be set to **DBX_PERSISTENT**, if so, a persistent connection will be created.

The *module* parameter can be either a string or a constant, though the latter form is preferred. The possible values are given below, but keep in mind that they only work if the module is actually loaded.

- **DBX_MYSQL** or "mysql"
- **DBX_ODBC** or "odbc"
- **DBX_PGSQL** or "pgsql"
- **DBX_MSSQL** or "mssql"
- **DBX_FBSQL** or "fbsql" (available from PHP 4.1.0)
- **DBX_SYBASECT** or "sybase_ct" (available from PHP 4.2.0)
- **DBX_OCI8** or "oci8" (available from PHP 4.3.0)

The *host*, *database*, *username* and *password* parameters are expected, but not always used depending on the connect functions for the abstracted module.

The returned `object` has three properties:

database

It is the name of the currently selected database.

handle

It is a valid handle for the connected database, and as such it can be used in module-specific functions (if required).

```
$link = dbx_connect (DBX_MYSQL, "localhost", "db", "username", "password");
mysql_close ($link->handle); // dbx_close($link) would be better here
```

module

It is used internally by dbx only, and is actually the module number mentioned above.

Example 1. dbx_connect() example

```
<?php
$link = dbx_connect (DBX_ODBC, "", "db", "username", "password", DBX_PERSISTENT)
    or die ("Could not connect");

print ("Connected successfully");
dbx_close ($link);
?>
```

Note: Always refer to the module-specific documentation as well.

See also: **dbx_close()**.

dbx_error

(PHP 4 >= 4.0.6)

dbx_error -- Report the error message of the latest function call in the module (not just in the connection)

Description

string **dbx_error** (object link_identifier)

dbx_error() returns a string containing the error message from the last function call of the abstracted module (e.g. mysql

module). If there are multiple connections in the same module, just the last error is given. If there are connections on different modules, the latest error is returned for the module specified by the *link_identifier* parameter.

Example 1. dbx_error() example

```
<?php
$link   = dbx_connect(DBX_MYSQL, "localhost", "db", "username", "password")
    or die ("Could not connect");

$result = dbx_query($link, "select id from non_existing_table");
if ( $result == 0 ) {
    echo dbx_error ($link);
}
dbx_close ($link);
?>
```

Note: Always refer to the module-specific documentation as well.

The error message for Microsoft SQL Server is actually the result of the **mssql_get_last_message()** function.

The error message for Oracle (oci8) is not implemented (yet).

dbx_escape_string

(PHP 4 >= 4.3.0)

dbx_escape_string -- Escape a string so it can safely be used in an sql-statement.

Description

string **dbx_escape_string** (object link_identifier, string text)

dbx_escape_string() returns the text, escaped where necessary (such as quotes, backslashes etc). It returns NULL on error.

Example 1. dbx_escape_string() example

```
<?php
$link   = dbx_connect(DBX_MYSQL, "localhost", "db", "username", "password")
    or die ("Could not connect");

$text = dbx_escape_string($link, "It\'s quoted and backslashed (\\).");
$result = dbx_query($link, "insert into tbl (txt) values ('".$text."')");
if ( $result == 0 ) {
    echo dbx_error ($link);
}
dbx_close ($link);
?>
```

See also: **dbx_query()**.

dbx_query

(PHP 4 >= 4.0.6)

dbx_query -- Send a query and fetch all results (if any)

Description

object **dbx_query** (object link_identifier, string sql_statement [, long flags])

dbx_query() returns an object or `1` on success, and `0` on failure. The result object is returned only if the query given in *sql_statement* produces a result set.

Example 1. How to handle the returned value

```
<?php
$link   = dbx_connect(DBX_ODBC, "", "db", "username", "password")
    or die("Could not connect");
```

```
$result = dbx_query($link, 'SELECT id, parentid, description FROM table');

if ( is_object($result) ) {
    // ... do some stuff here, see detailed examples below ...
    // first, print out field names and types
    // then, draw a table filled with the returned field values
}
else if ( $result == 1 ) {
    echo("Query executed successfully, but no result set returned");
}
else {
    exit("Query failed");
}

dbx_close($link);
?>
```

The *flags* parameter is used to control the amount of information that is returned. It may be any combination of the following constants with the bitwise OR operator (|). The DBX_COLNAMES_* flags override the dbx.colnames_case setting from `php.ini`.

DBX_RESULT_INDEX

It is *always* set, that is, the returned object has a data property which is a `2` dimensional array indexed numerically. For example, in the expression `data[2][3]` `2` stands for the row (or record) number and `3` stands for the column (or field) number. The first row and column are indexed at `0`.

If **DBX_RESULT_ASSOC** is also specified, the returning object contains the information related to **DBX_RESULT_INFO** too, even if it was not specified.

DBX_RESULT_INFO

It provides info about columns, such as field names and field types.

DBX_RESULT_ASSOC

It effects that the field values can be accessed with the respective column names used as keys to the returned object's data property.

Associated results are actually references to the numerically indexed data, so modifying `data[0][0]` causes that `data[0]['field_name_for_first_column']` is modified as well.

DBX_COLNAMES_UNCHANGED (available from PHP 4.3.0)

The case of the returned column names will not be changed.

DBX_COLNAMES_UPPERCASE (available from PHP 4.3.0)

The case of the returned column names will be changed to uppercase.

DBX_COLNAMES_LOWERCASE (available from PHP 4.3.0)

The case of the returned column names will be changed to lowercase.

Note that **DBX_RESULT_INDEX** is always used, regardless of the actual value of *flags* parameter. This means that the following combinations is effective only:

- **DBX_RESULT_INDEX**
- **DBX_RESULT_INDEX** | **DBX_RESULT_INFO**
- **DBX_RESULT_INDEX** | **DBX_RESULT_INFO** | **DBX_RESULT_ASSOC** - this is the default, if *flags* is not specified.

The returing `object` has four or five properties depending on *flags*:

handle

It is a valid handle for the connected database, and as such it can be used in module specific functions (if required).

```
$result = dbx_query ($link, "SELECT id FROM table");
mysql_field_len ($result->handle, 0);
```

cols and rows

These contain the number of columns (or fields) and rows (or records) respectively.

```
$result = dbx_query ($link, 'SELECT id FROM table');
echo $result->rows; // number of records
echo $result->cols; // number of fields
```

info (optional)

It is returned only if either **DBX_RESULT_INFO** or **DBX_RESULT_ASSOC** is specified in the *flags* parameter. It is a 2 dimensional array, that has two named rows (`name` and `type`) to retrieve column information.

Example 2. lists each field's name and type

```
$result = dbx_query ($link, 'SELECT id FROM table',
                     DBX_RESULT_INDEX | DBX_RESULT_INFO);

for ($i = 0; $i < $result->cols; $i++ ) {
    echo $result->info['name'][$i] . "\n";
    echo $result->info['type'][$i] . "\n";
}
```

data

This property contains the actual resulting data, possibly associated with column names as well depending on *flags*. If **DBX_RESULT_ASSOC** is set, it is possible to use `$result->data[2]["field_name"]`.

Example 3. outputs the content of data property into HTML table

```
$result = dbx_query ($link, 'SELECT id, parentid, description FROM table');

echo "<table>\n";
foreach ( $result->data as $row ) {
    echo "<tr>\n";
    foreach ( $row as $field ) {
        echo "<td>$field</td>";
    }
    echo "</tr>\n";
}
echo "</table>\n";
```

Note: Always refer to the module-specific documentation as well.

Column names for queries on an Oracle database are returned in lowercase.

See also: dbx_escape_string() and dbx_connect().

dbx_sort

(PHP 4 >= 4.0.6)

dbx_sort -- Sort a result from a dbx_query by a custom sort function

Description

bool **dbx_sort** (object result, string user_compare_function)

Returns **TRUE** on success or **FALSE** on failure.

Note: It is always better to use `ORDER BY` SQL clause instead of **dbx_sort()**, if possible.

Example 1. dbx_sort() example

```
<?php
function user_re_order ($a, $b) {
    $rv = dbx_compare ($a, $b, "parentid", DBX_CMP_DESC);
    if ( !$rv ) {
        $rv = dbx_compare ($a, $b, "id", DBX_CMP_NUMBER);
    }
    return $rv;
}

$link   = dbx_connect (DBX_ODBC, "", "db", "username", "password")
    or die ("Could not connect");

$result = dbx_query ($link, "SELECT id, parentid, description FROM tbl ORDER BY id");
    // data in $result is now ordered by id

dbx_sort ($result, "user_re_order");
    // data in $result is now ordered by parentid (descending), then by id

dbx_close ($link);
?>
```

See also **dbx_compare()**.

XXII. DB++ Functions

Warning
This extension is *EXPERIMENTAL*. The behaviour of this extension -- including the names of its functions and anything else documented about this extension -- may change without notice in a future release of PHP. Use this extension at your own risk.

Introduction

db++, made by the German company Concept asa, is a relational database system with high performance and low memory and disk usage in mind. While providing SQL as an additional language interface, it is not really a SQL database in the first place but provides its own AQL query language which is much more influenced by the relational algebra then SQL is.

Concept asa always had an interest in supporting open source languages, db++ has had Perl and Tcl call interfaces for years now and uses Tcl as its internal stored procedure language.

Requirements

This extension relies on external client libraries so you have to have a db++ client installed on the system you want to use this extension on.

Concept asa provides db++ Demo versions and documentation for Linux, some other UNIX versions. There is also a Windows version of db++, but this extension doesn't support it (yet).

Installation

In order to build this extension yourself you need the db++ client libraries and header files to be installed on your system (these are included in the db++ installation archives by default). You have to run **configure** with option `--with-dbplus` to build this extension.

configure looks for the client libraries and header files under the default paths `/usr/dbplus`, `/usr/local/dbplus` and `/opt/dblus`. If you have installed db++ in a different place you have add the installation path to the **configure** option like this: `--with-dbplus=/your/installation/path`.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

dbplus_relation

Most db++ functions operate on or return *`dbplus_relation`* resources. A *`dbplus_relation`* is a handle to a stored relation or a relation generated as the result of a query.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into

PHP or dynamically loaded at runtime.

---

db++ error codes

Table 1. DB++ Error Codes

PHP Constant	db++ constant	meaning
DBPLUS_ERR_NOERR (integer)	ERR_NOERR	Null error condition
DBPLUS_ERR_DUPLICATE (integer)	ERR_DUPLICATE	Tried to insert a duplicate tuple
DBPLUS_ERR_EOSCAN (integer)	ERR_EOSCAN	End of scan from rget()
DBPLUS_ERR_EMPTY (integer)	ERR_EMPTY	Relation is empty (server)
DBPLUS_ERR_CLOSE (integer)	ERR_CLOSE	The server can't close
DBPLUS_ERR_WLOCKED (integer)	ERR_WLOCKED	The record is write locked
DBPLUS_ERR_LOCKED (integer)	ERR_LOCKED	Relation was already locked
DBPLUS_ERR_NOLOCK (integer)	ERR_NOLOCK	Relation cannot be locked
DBPLUS_ERR_READ (integer)	ERR_READ	Read error on relation
DBPLUS_ERR_WRITE (integer)	ERR_WRITE	Write error on relation
DBPLUS_ERR_CREATE (integer)	ERR_CREATE	Create() system call failed
DBPLUS_ERR_LSEEK (integer)	ERR_LSEEK	Lseek() system call failed
DBPLUS_ERR_LENGTH (integer)	ERR_LENGTH	Tuple exceeds maximum length
DBPLUS_ERR_OPEN (integer)	ERR_OPEN	Open() system call failed
DBPLUS_ERR_WOPEN (integer)	ERR_WOPEN	Relation already opened for writing
DBPLUS_ERR_MAGIC (integer)	ERR_MAGIC	File is not a relation
DBPLUS_ERR_VERSION (integer)	ERR_VERSION	File is a very old relation
DBPLUS_ERR_PGSIZE (integer)	ERR_PGSIZE	Relation uses a different page size
DBPLUS_ERR_CRC (integer)	ERR_CRC	Invalid crc in the superpage
DBPLUS_ERR_PIPE (integer)	ERR_PIPE	Piped relation requires lseek()
DBPLUS_ERR_NIDX (integer)	ERR_NIDX	Too many secondary indices
DBPLUS_ERR_MALLOC (integer)	ERR_MALLOC	Malloc() call failed
DBPLUS_ERR_NUSERS (integer)	ERR_NUSERS	Error use of max users
DBPLUS_ERR_PREEXIT (integer)	ERR_PREEXIT	Caused by invalid usage
DBPLUS_ERR_ONTRAP (integer)	ERR_ONTRAP	Caused by a signal
DBPLUS_ERR_PREPROC (integer)	ERR_PREPROC	Error in the preprocessor
DBPLUS_ERR_DBPARSE (integer)	ERR_DBPARSE	Error in the parser
DBPLUS_ERR_DBRUNERR (integer)	ERR_DBRUNERR	Run error in db
DBPLUS_ERR_DBPREEXIT (integer)	ERR_DBPREEXIT	Exit condition caused by prexit() * procedure
DBPLUS_ERR_WAIT (integer)	ERR_WAIT	Wait a little (Simple only)
DBPLUS_ERR_CORRUPT_TUPLE (integer)	ERR_CORRUPT_TUPLE	A client sent a corrupt tuple
DBPLUS_ERR_WARNING0 (integer)	ERR_WARNINGo	The Simple routines encountered a non fatal error which was corrected
DBPLUS_ERR_PANIC (integer)	ERR_PANIC	The server should not really die but after a disaster send ERR_PANIC to all its clients
DBPLUS_ERR_FIFO (integer)	ERR_FIFO	Can't create a fifo
DBPLUS_ERR_PERM (integer)	ERR_PERM	Permission denied
DBPLUS_ERR_TCL (integer)	ERR_TCL	TCL_error
DBPLUS_ERR_RESTRICTED (integer)	ERR_RESTRICTED	Only two users
DBPLUS_ERR_USER (integer)	ERR_USER	An error in the use of the library by an application programmer
DBPLUS_ERR_UNKNOWN (integer)	ERR_UNKNOWN	

Table of Contents

dbplus_add

(4.1.0 - 4.2.3 only)

dbplus_add -- Add a tuple to a relation

Description

int **dbplus_add** (resource relation, array tuple)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

This function will add a tuple to a relation. The *tuple* data is an array of attribute/value pairs to be inserted into the given *relation*. After successful execution the *tuple* array will contain the complete data of the newly created tuple, including all implicitly set domain fields like sequences.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See **dbplus_errcode()** or the introduction to this chapter for more information on db++ error codes.

dbplus_aql

(4.1.0 - 4.2.3 only)

dbplus_aql -- Perform AQL query

Description

resource **dbplus_aql** (string query [, string server [, string dbpath]])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_aql() will execute an AQL *query* on the given *server* and *dbpath*.

On success it will return a relation handle. The result data may be fetched from this relation by calling **dbplus_next()** and **dbplus_current()**. Other relation access functions will not work on a result relation.

Further information on the AQL A... Query Language is provided in the original db++ manual.

dbplus_chdir

(4.1.0 - 4.2.3 only)

dbplus_chdir -- Get/Set database virtual current directory

Description

string **dbplus_chdir** ([string newdir])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_chdir() will change the virtual current directory where relation files will be looked for by **dbplus_open()**. **dbplus_chdir()** will return the absolute path of the current directory. Calling **dbplus_chdir()** without giving any *newdir* may be used to query the current working directory.

dbplus_close

(4.1.0 - 4.2.3 only)

dbplus_close -- Close a relation

Description

int **dbplus_close** (resource relation)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Calling **dbplus_close()** will close a relation previously opened by **dbplus_open()**.

dbplus_curr

(4.1.0 - 4.2.3 only)

dbplus_curr -- Get current tuple from relation

Description

int **dbplus_curr** (resource relation, array tuple)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_curr() will read the data for the current tuple for the given *relation* and will pass it back as an associative array in *tuple*.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See **dbplus_errcode()** or the introduction to this chapter for more information on db++ error codes.

See also **dbplus_first()**, **dbplus_prev()**, **dbplus_next()**, and **dbplus_last()**.

dbplus_errcode

(4.1.0 - 4.2.3 only)

dbplus_errcode -- Get error string for given errorcode or last error

Description

string **dbplus_errcode** (int errno)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_errcode() returns a cleartext error string for the error code passed as *errno* of for the result code of the last db++ operation if no parameter is given.

dbplus_errno

(4.1.0 - 4.2.3 only)

dbplus_errno -- Get error code for last operation

Description

int **dbplus_errno** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_errno() will return the error code returned by the last db++ operation.

See also **dbplus_errcode()**.

dbplus_find

(4.1.0 - 4.2.3 only)

dbplus_find -- Set a constraint on a relation

Description

int **dbplus_find** (resource relation, array constraints, mixed tuple)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_find() will place a constraint on the given relation. Further calls to functions like **dbplus_curr()** or **dbplus_next()** will only return tuples matching the given constraints.

Constraints are triplets of strings containing of a domain name, a comparison operator and a comparison value. The *constraints* parameter array may consist of a collection of string arrays, each of which contains a domain, an operator and a value, or of a single string array containing a multiple of three elements.

The comparison operator may be one of the following strings: '==', '>', '>=', '<', '<=', '!=', '~' for a regular expression match and 'BAND' or 'BOR' for bitwise operations.

See also **dbplus_unselect()**.

dbplus_first

(4.1.0 - 4.2.3 only)

dbplus_first -- Get first tuple from relation

Description

int **dbplus_first** (resource relation, array tuple)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_curr() will read the data for the first tuple for the given *relation*, make it the current tuple and pass it back as an associative array in *tuple*.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See **dbplus_errcode()** or the introduction to this chapter for more information on db++ error codes.

See also **dbplus_curr()**, **dbplus_prev()**, **dbplus_next()**, and **dbplus_last()**.

dbplus_flush

(4.1.0 - 4.2.3 only)

dbplus_flush -- Flush all changes made on a relation

Description

int **dbplus_flush** (resource relation)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_flush() will write all changes applied to *relation* since the last flush to disk.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See **dbplus_errcode()** or the introduction to this chapter for more information on db++ error codes.

dbplus_freealllocks

(4.1.0 - 4.2.3 only)

dbplus_freealllocks -- Free all locks held by this client

Description

int **dbplus_freealllocks** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_freeaalllocks() will free all tuple locks held by this client.

See also **dbplus_getlock()**, **dbplus_freelock()**, and **dbplus_freerlocks()**.

dbplus_freelock

(4.1.0 - 4.2.3 only)

dbplus_freelock -- Release write lock on tuple

Description

int **dbplus_freelock** (resource relation, string tname)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_freelock() will release a write lock on the given *`tuple`* previously obtained by **dbplus_getlock()**.

See also **dbplus_getlock()**, **dbplus_freerlocks()**, and **dbplus_freealllocks()**.

dbplus_freerlocks

(4.1.0 - 4.2.3 only)

dbplus_freerlocks -- Free all tuple locks on given relation

Description

int **dbplus_freerlocks** (resource relation)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_freerlocks() will free all tuple locks held on the given *`relation`*.

See also **dbplus_getlock()**, **dbplus_freelock()**, and **dbplus_freealllocks()**.

dbplus_getlock

(4.1.0 - 4.2.3 only)

dbplus_getlock -- Get a write lock on a tuple

Description

int **dbplus_getlock** (resource relation, string tname)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_getlock() will request a write lock on the specified `tuple`. It will return zero on success or a non-zero error code, especially DBPLUS_ERR_WLOCKED, on failure.

See also **dbplus_freelock()**, **dbplus_freerlocks()**, and **dbplus_freealllocks()**.

dbplus_getunique

(4.1.0 - 4.2.3 only)

dbplus_getunique -- Get a id number unique to a relation

Description

int **dbplus_getunique** (resource relation, int uniqueid)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_getunique() will obtain a number guaranteed to be unique for the given `relation` and will pass it back in the variable given as `uniqueid`.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See **dbplus_errcode()** or the introduction to this chapter for more information on db++ error codes.

dbplus_info

(4.1.0 - 4.2.3 only)

dbplus_info -- ???

Description

int **dbplus_info** (resource relation, string key, array)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Not implemented yet.

dbplus_last

(4.1.0 - 4.2.3 only)

dbplus_last -- Get last tuple from relation

Description

int **dbplus_last** (resource relation, array tuple)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_curr() will read the data for the last tuple for the given `relation`, make it the current tuple and pass it back as an associative array in `tuple`.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

See also dbplus_first(), dbplus_curr(), dbplus_prev(), and dbplus_next().

dbplus_lockrel

(no version information, might be only in CVS)

dbplus_lockrel -- Request write lock on relation

Description

int **dbplus_lockrel** (resource relation)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_lockrel() will request a write lock on the given relation. Other clients may still query the relation, but can't alter it while it is locked.

dbplus_next

(4.1.0 - 4.2.3 only)

dbplus_next -- Get next tuple from relation

Description

int **dbplus_next** (resource relation, array)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_curr() will read the data for the next tuple for the given *relation*, will make it the current tuple and will pass it back as an associative array in *tuple*.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

See also dbplus_first(), dbplus_curr(), dbplus_prev(), and dbplus_last().

dbplus_open

(4.1.0 - 4.2.3 only)

dbplus_open -- Open relation file

Description

resource **dbplus_open** (string name)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

The relation file *name* will be opened. *name* can be either a file name or a relative or absolute path name. This will be mapped in any case to an absolute relation file path on a specific host machine and server.

On success a relation file resource (cursor) is returned which must be used in any subsequent commands referencing the relation. Failure leads to a zero return value, the actual error code may be asked for by calling **dbplus_errno()**.

dbplus_prev

(4.1.0 - 4.2.3 only)

dbplus_prev -- Get previous tuple from relation

Description

int **dbplus_prev** (resource relation, array tuple)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_curr() will read the data for the next tuple for the given `relation`, will make it the current tuple and will pass it back as an associative array in `tuple`.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See **dbplus_errcode()** or the introduction to this chapter for more information on db++ error codes.

See also **dbplus_first()**, **dbplus_curr()**, **dbplus_next()**, and **dbplus_last()**.

dbplus_rchperm

(4.1.0 - 4.2.3 only)

dbplus_rchperm -- Change relation permissions

Description

int **dbplus_rchperm** (resource relation, int mask, string user, string group)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_rchperm() will change access permissions as specified by `mask`, `user` and `group`. The values for these are operating system specific.

dbplus_rcreate

(4.1.0 - 4.2.3 only)

dbplus_rcreate -- Creates a new DB++ relation

Description

resource **dbplus_rcreate** (string name, mixed domlist [, boolean overwrite])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_rcreate() will create a new relation named `name`. An existing relation by the same name will only be overwritten if the relation is currently not in use and `overwrite` is set to TRUE.

`domlist` should contain the domain specification for the new relation within an array of domain description strings. (**dbplus_rcreate()** will also accept a string with space delimited domain description strings, but it is recommended to use an

array). A domain description string consists of a domain name unique to this relation, a slash and a type specification character. See the db++ documentation, especially the dbcreate(1) manpage, for a description of available type specifiers and their meanings.

dbplus_rcrtexact

(4.1.0 - 4.2.3 only)

dbplus_rcrtexact -- Creates an exact but empty copy of a relation including indices

Description

resource **dbplus_rcrtexact** (string name, resource relation, boolean overwrite)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_rcrtexact() will create an exact but empty copy of the given *relation* under a new *name*. An existing relation by the same *name* will only be overwritten if *overwrite* is TRUE and no other process is currently using the relation.

dbplus_rcrtlike

(4.1.0 - 4.2.3 only)

dbplus_rcrtlike -- Creates an empty copy of a relation with default indices

Description

resource **dbplus_rcrtlike** (string name, resource relation, int flag)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_rcrtexact() will create an empty copy of the given *relation* under a new *name*, but with default indices. An existing relation by the same *name* will only be overwritten if *overwrite* is TRUE and no other process is currently using the relation.

dbplus_resolve

(4.1.0 - 4.2.3 only)

dbplus_resolve -- Resolve host information for relation

Description

int **dbplus_resolve** (string relation_name)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_resolve() will try to resolve the given *relation_name* and find out internal server id, real hostname and the database path on this host. The function will return an array containing these values under the keys 'sid', 'host' and 'host_path' or **FALSE** on error.

See also **dbplus_tcl()**.

dbplus_restorepos

(4.1.0 - 4.2.3 only)

dbplus_restorepos -- ???

Description

int **dbplus_restorepos** (resource relation, array tuple)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Not implemented yet.

dbplus_rkeys

(4.1.0 - 4.2.3 only)

dbplus_rkeys -- Specify new primary key for a relation

Description

resource **dbplus_rkeys** (resource relation, mixed domlist)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_rkeys() will replace the current primary key for `relation` with the combination of domains specified by `domlist`.

`domlist` may be passed as a single domain name string or as an array of domain names.

dbplus_ropen

(4.1.0 - 4.2.3 only)

dbplus_ropen -- Open relation file local

Description

resource **dbplus_ropen** (string name)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_ropen() will open the relation `file` locally for quick access without any client/server overhead. Access is read only and only **dbplus_current()** and **dbplus_next()** may be applied to the returned relation.

dbplus_rquery

(4.1.0 - 4.2.3 only)

dbplus_rquery -- Perform local (raw) AQL query

Description

int **dbplus_rquery** (string query, string dbpath)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_rquery() performs a local (raw) AQL query using an AQL interpreter embedded into the db++ client library. **dbplus_rquery()** is faster than **dbplus_aql()** but will work on local data only.

dbplus_rrename

(4.1.0 - 4.2.3 only)

dbplus_rrename -- Rename a relation

Description

int **dbplus_rrename** (resource relation, string name)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_rrename() will change the name of *relation* to *name*.

dbplus_rsecindex

(4.1.0 - 4.2.3 only)

dbplus_rsecindex -- Create a new secondary index for a relation

Description

resource **dbplus_rsecindex** (resource relation, mixed domlist, int type)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_rsecindex() will create a new secondary index for *relation* with consists of the domains specified by *domlist* and is of type *type*

domlist may be passed as a single domain name string or as an array of domain names.

dbplus_runlink

(4.1.0 - 4.2.3 only)

dbplus_runlink -- Remove relation from filesystem

Description

int **dbplus_runlink** (resource relation)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_unlink() will close and remove the *relation*.

dbplus_rzap

(4.1.0 - 4.2.3 only)

dbplus_rzap -- Remove all tuples from relation

Description

int **dbplus_rzap** (resource relation)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_rzap() will remove all tuples from `relation`.

dbplus_savepos

(4.1.0 - 4.2.3 only)

dbplus_savepos -- ???

Description

int **dbplus_savepos** (resource relation)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Not implemented yet.

dbplus_setindex

(4.1.0 - 4.2.3 only)

dbplus_setindex -- ???

Description

int **dbplus_setindex** (resource relation, string idx_name)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Not implemented yet.

dbplus_setindexbynumber

(4.1.0 - 4.2.3 only)

dbplus_setindexbynumber -- ???

Description

int **dbplus_setindexbynumber** (resource relation, int idx_number)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Not implemented yet.

dbplus_sql

(4.1.0 - 4.2.3 only)

dbplus_sql -- Perform SQL query

Description

resource **dbplus_sql** (string query, string server, string dbpath)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Not implemented yet.

dbplus_tcl

(4.1.0 - 4.2.3 only)

dbplus_tcl -- Execute TCL code on server side

Description

int **dbplus_tcl** (int sid, string script)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

A db++ server will prepare a TCL interpreter for each client connection. This interpreter will enable the server to execute TCL code provided by the client as a sort of stored procedures to improve the performance of database operations by avoiding client/server data transfers and context switches.

dbplus_tcl() needs to pass the client connection id the TCL *script* code should be executed by. **dbplus_resolve()** will provide this connection id. The function will return whatever the TCL code returns or a TCL error message if the TCL code fails.

See also **dbplus_resolve()**.

dbplus_tremove

(4.1.0 - 4.2.3 only)

dbplus_tremove -- Remove tuple and return new current tuple

Description

int **dbplus_tremove** (resource relation, array tuple [, array current])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_tremove() removes *tuple* from *relation* if it perfectly matches a tuple within the relation. *current*, if given, will contain the data of the new current tuple after calling **dbplus_tremove()**.

dbplus_undo

(4.1.0 - 4.2.3 only)

dbplus_undo -- ???

Description

int **dbplus_undo** (resource relation)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Not implemented yet.

dbplus_undoprepare

(4.1.0 - 4.2.3 only)

dbplus_undoprepare -- ???

Description

int **dbplus_undoprepare** (resource relation)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Not implemented yet.

dbplus_unlockrel

(4.1.0 - 4.2.3 only)

dbplus_unlockrel -- Give up write lock on relation

Description

int **dbplus_unlockrel** (resource relation)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_unlockrel() will release a write lock previously obtained by **dbplus_lockrel()**.

dbplus_unselect

(4.1.0 - 4.2.3 only)

dbplus_unselect -- Remove a constraint from relation

Description

int **dbplus_unselect** (resource relation)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Calling **dbplus_unselect()** will remove a constraint previously set by **dbplus_find()** on `relation`.

dbplus_update

(4.1.0 - 4.2.3 only)

dbplus_update -- Update specified tuple in relation

Description

int **dbplus_update** (resource relation, array old, array new)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_update() replaces the tuple given by `old` with the data from `new` if and only if `old` completely matches a tuple within `relation`.

dbplus_xlockrel

(4.1.0 - 4.2.3 only)

dbplus_xlockrel -- Request exclusive lock on relation

Description

int **dbplus_xlockrel** (resource relation)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_xlockrel() will request an exclusive lock on `relation` preventing even read access from other clients.

See also **dbplus_xunlockrel()**.

dbplus_xunlockrel

(4.1.0 - 4.2.3 only)

dbplus_xunlockrel -- Free exclusive lock on relation

Description

int **dbplus_xunlockrel** (resource relation)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

dbplus_xunlockrel() will release an exclusive lock on `relation` previously obtained by **dbplus_xlockrel()**.

XXIII. Direct IO functions

Introduction

PHP supports the direct io functions as described in the Posix Standard (Section 6) for performing I/O functions at a lower level than the C-Language stream I/O functions (**fopen()**, **fread()**,..). The use of the DIO functions should be considered only when direct control of a device is needed. In all other cases, the standard filesystem functions are more than adequate.

> **Note:** This extension is not available on Windows platforms.

Requirements

No external libraries are needed to build this extension.

Installation

To get these functions to work, you have to configure PHP with `--enable-dio`.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

One resource type is defined by this extension: a file descriptor returnded by **dio_open()**.

Predefined Constants

This extension has no constants defined.

Table of Contents

dio_close

(PHP 4 >= 4.2.0)

dio_close -- Closes the file descriptor given by fd

Description

void **dio_close** (resource fd)

The function **dio_close()** closes the file descriptor *resource*.

dio_fcntl

(PHP 4 >= 4.2.0)

dio_fcntl -- Performs a c library fcntl on fd

Description

mixed **dio_fcntl** (resource fd, int cmd [, mixed arg])

The **dio_fcntl()** function performs the operation specified by *cmd* on the file descriptor *fd*. Some commands require additional arguments *args* to be supplied.

arg is an associative array, when *cmd* is F_SETLK or F_SETLLW, with the following keys:

- "start" - offset where lock begins
- "length" - size of locked area. zero means to end of file
- "wenth" - Where l_start is relative to: can be SEEK_SET, SEEK_END and SEEK_CUR
- "type" - type of lock: can be F_RDLCK (read lock), F_WRLCK (write lock) or F_UNLCK (unlock)

cmd can be one of the following operations:

- F_SETLK - Lock is set or cleared. If the lock is held by someone else **dio_fcntl()** returns -1.
- F_SETLKW - like F_SETLK, but in case the lock is held by someone else, **dio_fcntl()** waits until the lock is released.
- F_GETLK - **dio_fcntl()** returns an associative array (as described above) if someone else prevents lock. If there is no obstruction key "type" will set to F_UNLCK.
- F_DUPFD - finds the lowest numbered available file descriptor greater or equal than *arg* and returns them.
- F_SETFL - Sets the file descriptors flags to the value specified by *arg*, Which can be O_APPEND,O_NONBLOCK or O_ASYNC . To use O_ASYNC you will need to use the pcntl extension.

dio_open

(PHP 4 >= 4.2.0)

dio_open -- Opens a new filename with specified permissions of flags and creation permissions of mode

Description

resource **dio_open** (string filename, int flags [, int mode])

dio_open() opens a file and returns a new file descriptor for it, or **FALSE** if any error occurred. If *flags* is O_CREAT, optional third parameter *mode* will set the mode of the file (creation permissions). The *flags* parameter can be one of the following options:

- O_RDONLY - opens the file for read access
- O_WRONLY - opens the file for write access
- O_RDWR - opens the file for both reading and writing

The *flags* parameter can also include any combination of the following flags:

- O_CREAT - creates the file, if it doesn't already exist
- O_EXCL - if both, O_CREAT and O_EXCL are set, **dio_open()** fails, if file already exists
- O_TRUNC - if file exists, and its opened for write access, file will be truncated to zero length.
- O_APPEND - write operations write data at the end of file
- O_NONBLOCK - sets non blocking mode

dio_read

(PHP 4 >= 4.2.0)

dio_read -- Reads n bytes from fd and returns them, if n is not specified, reads 1k block

Description

string **dio_read** (resource fd [, int n])

The function **dio_read()** reads and returns *n* bytes from file with descriptor *resource*. If *n* is not specified, **dio_read()** reads 1K sized block and returns them.

dio_seek

(PHP 4 >= 4.2.0)

dio_seek -- Seeks to pos on fd from whence

Description

int **dio_seek** (resource fd, int pos, int whence)

The function **dio_seek()** is used to change the file position of the file with descriptor *resource*. The parameter *whence* specifies how the position *pos* should be interpreted:

- SEEK_SET - specifies that *pos* is specified from the beginning of the file
- SEEK_CUR - Specifies that *pos* is a count of characters from the current file position. This count may be positive or negative
- SEEK_END - Specifies that *pos* is a count of characters from the end of the file. A negative count specifies a position within the current extent of the file; a positive count specifies a position past the current end. If you set the position past the current end, and actually write data, you will extend the file with zeros up to that position

dio_stat

(PHP 4 >= 4.2.0)

dio_stat -- Gets stat information about the file descriptor fd

Description

array **dio_stat** (resource fd)

Function **dio_stat()** returns information about the file with file descriptor *fd*. **dio_stat()** returns an associative array with the following keys:

- "device" - device
- "inode" - inode
- "mode" - mode
- "nlink" - number of hard links
- "uid" - user id
- "gid" - group id
- "device_type" - device type (if inode device)
- "size" - total size in bytes
- "blocksize" - blocksize

- "blocks" - number of blocks allocated
- "atime" - time of last access
- "mtime" - time of last modification
- "ctime" - time of last change

On error **dio_stat()** returns NULL.

dio_tcsetattr

(PHP 4 >= 4.3.0)

dio_tcsetattr -- Sets terminal attributes and baud rate for a serial port

Description

dio_tcsetattr (resource fd, array options)

The function **dio_tcsetattr()** sets the terminal attributes and baud rate of the open *resource*. The currently available options are

- 'baud' - baud rate of the port - can be 38400,19200,9600,4800,2400,1800,1200,600,300,200,150,134,110,75 or 50, default value is 9600
- 'bits' - data bits - can be 8,7,6 or 5 default value is 8
- 'stop' - stop bits - can be 1 or 2 default value is 1
- 'parity' - can be 0,1 or 2 default value is 0

Example 1. Setting the baud rate on a serial port

```
<?php

$fd = dio_open('/dev/ttyS0', O_RDWR | O_NOCTTY | O_NONBLOCK);

dio_fcntl($fd,F_SETFL, O_SYNC );

dio_tcsetattr($fd, array(
  'baud' => 9600,
  'bits' => 8,
  'stop'  =>1,
  'parity' => 0
));

while (1) {

  $data = dio_read($fd,256);

  if ($data) {
      echo $data;
  }
}

?>
```

Note: This function was introduced in PHP 4.3.0.

dio_truncate

(PHP 4 >= 4.2.0)

dio_truncate -- Truncates file descriptor fd to offset bytes

Description

bool **dio_truncate** (resource fd, int offset)

Function **dio_truncate()** causes the file referenced by *fd* to be truncated to at most *offset* bytes in size. If the file previously was larger than this size, the extra data is lost. If the file previously was shorter, it is unspecified whether the file is left unchanged or is extended. In the latter case the extended part reads as zero bytes. Returns 0 on success, otherwise -1.

dio_write

(PHP 4 >= 4.2.0)

dio_write -- Writes data to fd with optional truncation at length

Description

int **dio_write** (resource fd, string data [, int len])

The function **dio_write()** writes up to *len* bytes from *data* to file *fd*. If *len* is not specified, **dio_write()** writes all *data* to the specified file. **dio_write()** returns the number of bytes written to *fd*.

XXIV. Directory functions

Introduction

Requirements

No external libraries are needed to build this extension.

Installation

There is no installation needed to use these functions; they are part of the PHP core.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

Predefined Constants

This extension has no constants defined.

See Also

For related functions such as **dirname()**, **is_dir()**, **mkdir()**, and **rmdir()**, see the Filesystem section.

Table of Contents

readdir -- read entry from directory handle
rewinddir -- rewind directory handle

chdir

(PHP 3, PHP 4)

chdir -- Change directory

Description

bool **chdir** (string directory)

Changes PHP's current directory to *directory*. Returns **TRUE** on success or **FALSE** on failure..

See also **getcwd()**.

chroot

(PHP 4 >= 4.0.5)

chroot -- Change the root directory

Description

bool **chroot** (string directory)

Changes the root directory of the current process to *directory*. Returns **TRUE** on success or **FALSE** on failure..

> **Note:** It's not wise to use this function when running in a webserver environment, because it's not possible to reset the root directory to / again at the end of the request. This function will only function correct when you run PHP as command line too. (CLI)
>
> **Note:** This function is not implemented on Windows platforms.

dir

dir -- directory class

Description

```
 class dir {

  dir(string directory);
  string path;
  resource handle;

  string read();
  void rewind();
  void close();
}
```

A pseudo-object oriented mechanism for reading a directory. The given *directory* is opened. Two properties are available once the directory has been opened. The handle property can be used with other directory functions such as **readdir()**, **rewinddir()** and **closedir()**. The path property is set to path the directory that was opened. Three methods are available: read, rewind and close.

Please note the fashion in which **dir()**'s return value is checked in the example below. We are explicitly testing whether the return value is identical to (equal to and of the same type as--see Comparison Operators for more information) **FALSE** since otherwise, any directory entry whose name evaluates to **FALSE** will stop the loop.

Example 1. dir() example

```
$d = dir("/etc");
echo "Handle: ".$d->handle."<br>\n";
echo "Path: ".$d->path."<br>\n";
while (false !== ($entry = $d->read())) {
    echo $entry."<br>\n";
```

```
}
$d->close();
```

Note: The order in which directory entries are returned by the read method is system-dependent.

Note: This defines the internal class **Directory**, meaning that you will not be able to define your own classes with that name. For a full list of predefined classes in PHP, please see Predefined Classes.

closedir

(PHP 3, PHP 4)

closedir -- close directory handle

Description

void **closedir** (resource dir_handle)

Closes the directory stream indicated by *dir_handle*. The stream must have previously been opened by **opendir()**.

getcwd

(PHP 4)

getcwd -- gets the current working directory

Description

string **getcwd** (void)

Returns the current working directory.

See also **chdir()**.

opendir

(PHP 3, PHP 4)

opendir -- open directory handle

Description

resource **opendir** (string path)

Returns a directory handle to be used in subsequent **closedir()**, **readdir()**, and **rewinddir()** calls.

If *path* is not a valid directory or the directory can not be opened due to permission restrictions or filesystem errors, **opendir()** returns **FALSE** and generates a PHP error. You can suppress the error output of **opendir()** by prepending `@' to the front of the function name.

Example 1. opendir() example

```
<?php

if ($dir = @opendir("/tmp")) {
  while (($file = readdir($dir)) !== false) {
    echo "$file\n";
  }
  closedir($dir);
}

?>
```

See also **is_dir()**.

readdir

(PHP 3, PHP 4)

readdir -- read entry from directory handle

Description

string **readdir** (resource dir_handle)

Returns the filename of the next file from the directory. The filenames are returned in the order in which they are stored by the filesystem.

Please note the fashion in which **readdir()**'s return value is checked in the examples below. We are explicitly testing whether the return value is identical to (equal to and of the same type as--see Comparison Operators for more information) **FALSE** since otherwise, any directory entry whose name evaluates to **FALSE** will stop the loop (e.g. a directory named "0").

Example 1. List all files in a directory

```
// Note that !== did not exist until 4.0.0-RC2
<?php
if ($handle = opendir('/path/to/files')) {
    echo "Directory handle: $handle\n";
    echo "Files:\n";

    /* This is the correct way to loop over the directory. */
    while (false !== ($file = readdir($handle))) {
        echo "$file\n";
    }

    /* This is the WRONG way to loop over the directory. */
    while ($file = readdir($handle)) {
        echo "$file\n";
    }

    closedir($handle);
}
?>
```

Note that **readdir()** will return the . and .. entries. If you don't want these, simply strip them out:

Example 2. List all files in the current directory and strip out . and ..

```
<?php
if ($handle = opendir('.')) {
    while (false !== ($file = readdir($handle))) {
        if ($file != "." && $file != "..") {
            echo "$file\n";
        }
    }
    closedir($handle);
}
?>
```

See also **is_dir()**.

rewinddir

(PHP 3, PHP 4)

rewinddir -- rewind directory handle

Description

void **rewinddir** (resource dir_handle)

Resets the directory stream indicated by *dir_handle* to the beginning of the directory.

XXV. DOM XML functions

Introduction

Warning
This extension is *EXPERIMENTAL*. The behaviour of this extension -- including the names of its functions and anything else documented about this extension -- may change without notice in a future release of PHP. Use this extension at your own risk.

The DOM XML extension has been overhauled in PHP 4.3.0 to better comply with the DOM standard. The extension still contains many old functions, but they should no longer be used. In particular, functions that are not object-oriented should be avoided.

The extension allows you to operate on an XML document with the DOM API. It also provides a function **domxml_xmltree()** to turn the complete XML document into a tree of PHP objects. Currently, this tree should be considered read-only — you can modify it, but this would not make any sense since **DomDocument_dump_mem()** cannot be applied to it. Therefore, if you want to read an XML file and write a modified version, use **DomDocument_create_element()**, **DomDocument_create_text_node()**, **set_attribute()**, etc. and finally the **DomDocument_dump_mem()** function.

Requirements

This extension makes use of the GNOME XML library. Download and install this library. You will need at least libxml-2.4.14. To use DOM XSLT features you can use the libxslt library and EXSLT enhancements from http://www.exslt.org/. Download and install these libraries if you plan to use (enhanced> XSLT features. You will need at least libxslt-1.0.18.

Installation

This extension is only available if PHP was configured with `--with-dom[=DIR]`. Add `--with-dom-xslt[=DIR]` to include DOM XSLT support. DIR is the libxslt install directory. Add `--with-dom-exslt[=DIR]` to include DOM EXSLT support, where DIR is the libexslt install directory.

> **Note to Win32 Users:** In order to enable this module on a Windows environment, you must copy *libxml2.dll* from the DLL folder of the PHP/Win32 binary package to the SYSTEM32 folder of your windows machine. (Ex: C:\WINNT\SYSTEM32 or C:\WINDOWS\SYSTEM32)

Deprecated functions

There are quite a few functions that do not fit into the DOM standard and should no longer be used. These functions are listed in the following table. The function **DomNode_append_child()** has changed its behaviour. It now adds a child and not a sibling. If this breaks your application, use the non-DOM function **DomNode_append_sibling()**.

Table 1. Deprecated functions and their replacements

Old function	New function
xmldoc	**domxml_open_mem()**
xmldocfile	**domxml_open_file()**
domxml_new_xmldoc	**domxml_new_doc()**
domxml_dump_mem	**DomDocument_dump_mem()**
domxml_dump_mem_file	**DomDocument_dump_file()**
DomDocument_dump_mem_file	**DomDocument_dump_file()**
DomDocument_add_root	**DomDocument_create_element()** followed by **DomNode_append_child()**
DomDocument_dtd	**DomDocument_doctype()**
DomDocument_root	**DomDocument_document_element()**
DomDocument_children	**DomNode_child_nodes()**
DomDocument_imported_node	No replacement.
DomNode_add_child	Create a new node with e.g. **DomDocument_create_element()** und add it with **DomNode_append_child()**.

Old function	New function
DomNode_children	**DomNode_child_nodes()**
DomNode_parent	**DomNode_parent_node()**
DomNode_new_child	Create a new node with e.g. **DomDocument_create_element()** and add it with **DomNode_append_child()**.
DomNode_set_content	Create a new node with e.g. **DomDocument_create_text_node()** and add it with **DomNode_append_child()**.
DomNode_get_content	Content is just a text node and can be accessed with **DomNode_child_nodes()**.
DomNode_set_content	Content is just a text node and can be added with **DomNode_append_child()**.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

Table 2. XML constants

Constant	Value	Description
`XML_ELEMENT_NODE` (integer)	1	Node is an element
`XML_ATTRIBUTE_NODE` (integer)	2	Node is an attribute
`XML_TEXT_NODE` (integer)	3	Node is a piece of text
`XML_CDATA_SECTION_NODE` (integer)	4	
`XML_ENTITY_REF_NODE` (integer)	5	
`XML_ENTITY_NODE` (integer)	6	Node is an entity like
`XML_PI_NODE` (integer)	7	Node is a processing instruction
`XML_COMMENT_NODE` (integer)	8	Node is a comment
`XML_DOCUMENT_NODE` (integer)	9	Node is a document
`XML_DOCUMENT_TYPE_NODE` (integer)	10	
`XML_DOCUMENT_FRAG_NODE` (integer)	11	
`XML_NOTATION_NODE` (integer)	12	
`XML_GLOBAL_NAMESPACE` (integer)	1	
`XML_LOCAL_NAMESPACE` (integer)	2	
`XML_HTML_DOCUMENT_NODE` (integer)		
`XML_DTD_NODE` (integer)		
`XML_ELEMENT_DECL_NODE` (integer)		
`XML_ATTRIBUTE_DECL_NODE` (integer)		
`XML_ENTITY_DECL_NODE` (integer)		
`XML_NAMESPACE_DECL_NODE` (integer)		
`XML_ATTRIBUTE_CDATA` (integer)		
`XML_ATTRIBUTE_ID` (integer)		
`XML_ATTRIBUTE_IDREF` (integer)		
`XML_ATTRIBUTE_IDREFS` (integer)		
`XML_ATTRIBUTE_ENTITY` (integer)		
`XML_ATTRIBUTE_NMTOKEN` (integer)		
`XML_ATTRIBUTE_NMTOKENS` (integer)		
`XML_ATTRIBUTE_ENUMERATION` (integer)		
`XML_ATTRIBUTE_NOTATION` (integer)		
`XPATH_UNDEFINED` (integer)		
`XPATH_NODESET` (integer)		
`XPATH_BOOLEAN` (integer)		
`XPATH_NUMBER` (integer)		
`XPATH_STRING` (integer)		

Constant	Value	Description
XPATH_POINT (integer)		
XPATH_RANGE (integer)		
XPATH_LOCATIONSET (integer)		
XPATH_USERS (integer)		
XPATH_NUMBER (integer)		

Classes

The API of the module follows the DOM Level 2 standard as closely as possible. Consequently, the API is fully object-oriented. It is a good idea to have the DOM standard available when using this module. Though the API is object-oriented, there are many functions which can be called in a non-object-oriented way by passing the object to operate on as the first argument. These functions are mainly to retain compatibilty to older versions of the extension, and should not be used when creating new scripts.

This API differs from the official DOM API in two ways. First, all class attributes are implemented as functions with the same name. Secondly, the function names follow the PHP naming convention. This means that a DOM function lastChild() will be written as last_child().

This module defines a number of classes, which are listed — including their method — in the following tables. Classes with an equivalent in the DOM standard are named DOMxxx.

Table 3. List of classes

Class name	Parent classes
DomAttribute	DomNode
DomCData	DomNode
DomComment	DomCData : DomNode
DomDocument	DomNode
DomDocumentType	DomNode
DomElement	DomNode
DomEntity	DomNode
DomEntityReference	DomNode
DomProcessingInstruction	DomNode
DomText	DomCData : DomNode
Parser	Currently still called DomParser
XPathContext	

Table 4. DomDocument class (DomDocument : DomNode)

Method name	Function name	Remark
doctype	**DomDocument_doctype()**	
document_elemnent	**DomDocument_document_element()**	
create_element	**DomDocument_create_element()**	
create_text_node	**DomDocument_create_text_node()**	
create_comment	**DomDocument_create_comment()**	
create_cdata_section	**DomDocument_create_cdata_section()**	
create_processing_instruction	**DomDocument_create_processing_instruction()**	
create_attribute	**DomDocument_create_attribute()**	
create_entity_reference	**DomDocument_create_entity_reference()**	
get_elements_by_tagname	**DomDocument_get_elements_by_tagname()**	
get_element_by_id	**DomDocument_get_element_by_id()**	
dump_mem	**DomDocument_dump_mem()**	not DOM standard
dump_file	**DomDocument_dump_file()**	not DOM standard
html_dump_mem	**DomDocument_html_dump_mem()**	not DOM standard
xpath_init	xpath_init	not DOM standard

Method name	Function name	Remark
xpath_new_context	xpath_new_context	not DOM standard
xptr_new_context	xptr_new_context	not DOM standard

Table 5. DomElement class (DomElement : DomNode)

Method name	Function name	Remark
tagname	**DomElement_tagname()**	
get_attribute	**DomElement_get_attribute()**	
set_attribute	**DomElement_set_attribute()**	
remove_attribute	**DomElement_remove_attribute()**	
get_attribute_node	**DomElement_get_attribute_node()**	
get_elements_by_tagname	**DomElement_get_elements_by_tagname()**	
has_attribute	**DomElement_has_attribute()**	

Table 6. DomNode class

Method name	Remark
DomNode_node_name()	
DomNode_node_value()	
DomNode_node_type()	
DomNode_last_child()	
DomNode_first_child()	
DomNode_child_nodes()	
DomNode_previous_sibling()	
DomNode_next_sibling()	
DomNode_parent_node()	
DomNode_owner_document()	
DomNode_insert_before()	
DomNode_append_child()	
DomNode_append_sibling()	Not in DOM standard. This function emulates the former behaviour of **DomNode_append_child()**.
DomNode_remove_child()	
DomNode_has_child_nodes()	
DomNode_has_attributes()	
DomNode_clone_node()	
DomNode_attributes()	
DomNode_unlink_node()	Not in DOM standard
DomNode_replace_node()	Not in DOM standard
DomNode_set_content()	Not in DOM standard, deprecated
DomNode_get_content()	Not in DOM standard, deprecated
DomNode_dump_node()	Not in DOM standard
DomNode_is_blank_node()	Not in DOM standard

Table 7. DomAttribute class (DomAttribute : DomNode)

Method name		Remark
name	**DomAttribute_name()**	
value	**DomAttribute_value()**	
specified	**DomAttribute_specified()**	

Table 8. DomProcessingInstruction class (DomProcessingInstruction : DomNode)

Method name	Function name	Remark
target	**DomProcessingInstruction_target()**	

Method name	Function name	Remark
data	**DomProcessingInstruction_data()**	

Table 9. Parser class

Method name	Function name	Remark
add_chunk	**Parser_add_chunk()**	
end	**Parser_end()**	

Table 10. XPathContext class

Method name	Function name	Remark
eval	**XPathContext_eval()**	
eval_expression	**XPathContext_eval_expression()**	
register_ns	**XPathContext_register_ns()**	

Table 11. DomDocumentType class (DomDocumentType : DomNode)

Method name	Function name	Remark
name	**DomDocumentType_name()**	
entities	**DomDocumentType_entities()**	
notations	**DomDocumentType_notations()**	
public_id	**DomDocumentType_public_id()**	
system_id	**DomDocumentType_system_id()**	
internal_subset	**DomDocumentType_internal_subset()**	

The classes DomDtd is derived from DomNode. DomComment is derived from DomCData.

Examples

Many examples in this reference require an XML string. Instead of repeating this string in every example, it will be put into a file which will be included by each example. This include file is shown in the following example section. Alternatively, you could create an XML document and read it with **DomDocument_open_file()**.

Example 1. Include file example.inc with XML string

```
<?php
$xmlstr = "<?xml version='1.0' standalone='yes'?>
<!DOCTYPE chapter SYSTEM '/share/sgml/Norman_Walsh/db3xml10/db3xml10.dtd'
[ <!ENTITY sp \"spanish\">
]>
<!-- lsfj  -->
<chapter language='en'><title language='en'>Title</title>
 <para language='ge'>
  &amp;sp;
  <!-- comment -->
  <informaltable ID='findme' language='&amp;sp;'>
   <tgroup cols='3'>
    <tbody>
     <row><entry>a1</entry><entry
morerows='1'>b1</entry><entry>c1</entry></row>
<row><entry>a2</entry><entry>c2</entry></row>
     <row><entry>a3</entry><entry>b3</entry><entry>c3</entry></row>
    </tbody>
   </tgroup>
  </informaltable>
 </para>
</chapter>";
?>
```

Table of Contents

DomDocument->create_comment -- Create new comment node
DomDocument->create_element_ns -- Create new element node with an associated namespace
DomDocument->create_element -- Create new element node
DomDocument->create_entity_reference --
DomDocument->create_processing_instruction -- Creates new PI node
DomDocument->create_text_node -- Create new text node
DomDocument->doctype -- Returns the document type
DomDocument->document_element -- Returns root element node
DomDocument->dump_file -- Dumps the internal XML tree back into a file
DomDocument->dump_mem -- Dumps the internal XML tree back into a string
DomDocument->get_element_by_id -- Searches for an element with a certain id
DomDocument->get_elements_by_tagname --
DomDocument->html_dump_mem -- Dumps the internal XML tree back into a string as HTML
DomDocument->xinclude -- Substitutes XIncludes in a DomDocument Object.
DomDocumentType->entities -- Returns list of entities
DomDocumentType->internal_subset -- Returns internal subset
DomDocumentType->name -- Returns name of document type
DomDocumentType->notations -- Returns list of notations
DomDocumentType->public_id -- Returns public id of document type
DomDocumentType->system_id -- Returns system id of document type
DomElement->get_attribute_node -- Returns value of attribute
DomElement->get_attribute -- Returns value of attribute
DomElement->get_elements_by_tagname -- Gets elements by tagname
DomElement->has_attribute -- Checks to see if attribute exists
DomElement->remove_attribute -- Removes attribute
DomElement->set_attribute -- Adds new attribute
DomElement->tagname -- Returns name of element
DomNode->add_namespace -- Adds a namespace declaration to a node.
DomNode->append_child -- Adds new child at the end of the children
DomNode->append_sibling -- Adds new sibling to a node
DomNode->attributes -- Returns list of attributes
DomNode->child_nodes -- Returns children of node
DomNode->clone_node -- Clones a node
DomNode->dump_node -- Dumps a single node
DomNode->first_child -- Returns first child of node
DomNode->get_content -- Gets content of node
DomNode->has_attributess -- Checks if node has attributes
DomNode->has_child_nodes -- Checks if node has children
DomNode->insert_before -- Inserts new node as child
DomNode->is_blank_node -- Checks if node is blank
DomNode->last_child -- Returns last child of node
DomNode->next_sibling -- Returns the next sibling of node
DomNode->node_name -- Returns name of node
DomNode->node_type -- Returns type of node
DomNode->node_value -- Returns value of a node
DomNode->owner_document -- Returns the document this node belongs to
DomNode->parent_node -- Returns the parent of the node
DomNode->prefix -- Returns name space prefix of node
DomNode->previous_sibling -- Returns the previous sibling of node
DomNode->remove_child -- Removes child from list of children
DomNode->replace_child -- Replaces a child
DomNode->replace_node -- Replaces node
DomNode->set_content -- Sets content of node
DomNode->set_name -- Sets name of node
DomNode->set_namespace -- Sets namespace of a node.
DomNode->unlink_node -- Deletes node
DomProcessingInstruction->data -- Returns data of pi node
DomProcessingInstruction->target -- Returns target of pi node
DomXsltStylesheet->process -- Applies the XSLT-Transformation on a DomDocument Object.
DomXsltStylesheet->result_dump_file -- Dumps the result from a XSLT-Transformation into a file
DomXsltStylesheet->result_dump_mem -- Dumps the result from a XSLT-Transformation back into a string
domxml_new_doc -- Creates new empty XML document
domxml_open_file -- Creates a DOM object from XML file
domxml_open_mem -- Creates a DOM object of an XML document
domxml_version -- Get XML library version
domxml_xmltree -- Creates a tree of PHP objects from an XML document
domxml_xslt_stylesheet_doc -- Creates a DomXsltStylesheet Object from a DomDocument Object.
domxml_xslt_stylesheet_file -- Creates a DomXsltStylesheet Object from a xsl document in a file.
domxml_xslt_stylesheet -- Creates a DomXsltStylesheet Object from a xml document in a string.
xpath_eval_expression -- Evaluates the XPath Location Path in the given string

xpath_eval -- Evaluates the XPath Location Path in the given string
xpath_new_context -- Creates new xpath context
xptr_eval -- Evaluate the XPtr Location Path in the given string
xptr_new_context -- Create new XPath Context

DomAttribute->name

(no version information, might be only in CVS)

DomAttribute->name -- Returns name of attribute

Description

bool **DomAttribute->name** (void)

This function returns the name of the attribute.

See also **domattribute_value()**.

DomAttribute->specified

(no version information, might be only in CVS)

DomAttribute->specified -- Checks if attribute is specified

Description

bool **DomAttribute->specified** (void)

Check DOM standard for a detailed explanation.

DomAttribute->value

(no version information, might be only in CVS)

DomAttribute->value -- Returns value of attribute

Description

bool **DomAttribute->value** (void)

This function returns the value of the attribute.

See also **domattribute_name()**.

DomDocument->add_root [deprecated]

(no version information, might be only in CVS)

DomDocument->add_root [deprecated] -- Adds a root node

Description

resource **DomDocument->add_root** (string name)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Adds a root element node to a dom document and returns the new node. The element name is given in the passed parameter.

Example 1. Creating a simple HTML document header

```
<?php
$doc = domxml_new_doc("1.0");
$root = $doc->add_root("HTML");
$head = $root->new_child("HEAD", "");
$head->new_child("TITLE", "Hier der Titel");
echo htmlentities($doc->dump_mem());
?>
```

DomDocument->create_attribute

(no version information, might be only in CVS)

DomDocument->create_attribute -- Create new attribute

Description

object **DomDocument->create_attribute** (string name, string value)

This function returns a new instance of class **DomAttribute**. The name of the attribute is the value of the first parameter. The value of the attribute is the value of the second parameter. This node will not show up in the document unless it is inserted with e.g. **domnode_append_child()**.

The return value is **FALSE** if an error occured.

See also **domnode_append_child()**, **domdocument_create_element()**, **domdocument_create_text()**, **domdocument_create_cdata_section()**, **domdocument_create_processing_instruction()**, **domdocument_create_entity_reference()**, and **domnode_insert_before()**.

DomDocument->create_cdata_section

(no version information, might be only in CVS)

DomDocument->create_cdata_section -- Create new cdata node

Description

string **DomDocument->create_cdata_section** (string content)

This function returns a new instance of class **DomCData**. The content of the cdata is the value of the passed parameter. This node will not show up in the document unless it is inserted with e.g. **domnode_append_child()**.

The return value is **FALSE** if an error occured.

See also **domnode_append_child()**, **domdocument_create_element()**, **domdocument_create_text()**, **domdocument_create_attribute()**, **domdocument_create_processing_instruction()**, **domdocument_create_entity_reference()**, and **domnode_insert_before()**.

DomDocument->create_comment

(no version information, might be only in CVS)

DomDocument->create_comment -- Create new comment node

Description

object **DomDocument->create_comment** (string content)

This function returns a new instance of class **DomComment**. The content of the comment is the value of the passed parameter. This node will not show up in the document unless it is inserted with e.g. **domnode_append_child()**.

The return value is **FALSE** if an error occured.

See also **domnode_append_child()**, **domdocument_create_element()**, **domdocument_create_text()**, **domdocument_create_attribute()**, **domdocument_create_processing_instruction()**, **domdocument_create_entity_reference()** and **domnode_insert_before()**.

DomDocument->create_element_ns

(no version information, might be only in CVS)

DomDocument->create_element_ns -- Create new element node with an associated namespace

Description

object **DomDocument->create_element_ns** (string uri, string name [, string prefix])

This function returns a new instance of class **DomElement**. The tag name of the element is the value of the passed parameter *name*. The URI of the namespace is the value of the passed parameter *uri*. If there is already a namespace declaration with the same uri in the root-node of the document, the prefix of this is taken, otherwise it will take the one provided in the optional parameter *prefix* or generate a random one. This node will not show up in the document unless it is inserted with e.g. **domnode_append_child()**.

The return value is **FALSE** if an error occured.

See also **domdocument_create_element_ns()**, **domnode_add_namespace()**, **domnode_set_namespace()**, **domnode_append_child()**, **domdocument_create_text()**, **domdocument_create_comment()**, **domdocument_create_attribute()**, **domdocument_create_processing_instruction()**, **domdocument_create_entity_reference()**, and **domnode_insert_before()**.

DomDocument->create_element

(no version information, might be only in CVS)

DomDocument->create_element -- Create new element node

Description

object **DomDocument->create_element** (string name)

This function returns a new instance of class **DomElement**. The tag name of the element is the value of the passed parameter. This node will not show up in the document unless it is inserted with e.g. **domnode_append_child()**.

The return value is **FALSE** if an error occured.

See also **domdocument_create_element_ns()**, **domnode_append_child()**, **domdocument_create_text()**, **domdocument_create_comment()**, **domdocument_create_attribute()**, **domdocument_create_processing_instruction()**, **domdocument_create_entity_reference()**, and **domnode_insert_before()**.

DomDocument->create_entity_reference

(no version information, might be only in CVS)

DomDocument->create_entity_reference --

Description

object **DomDocument->create_entity_reference** (string content)

This function returns a new instance of class **DomEntityReference**. The content of the entity reference is the value of the passed parameter. This node will not show up in the document unless it is inserted with e.g. **domnode_append_child()**.

The return value is **FALSE** if an error occured.

See also **domnode_append_child()**, **domdocument_create_element()**, **domdocument_create_text()**, **domdocument_create_cdata_section()**, **domdocument_create_processing_instruction()**, **domdocument_create_attribute()**, and **domnode_insert_before()**.

DomDocument->create_processing_instruction

(no version information, might be only in CVS)

DomDocument->create_processing_instruction -- Creates new PI node

Description

string **DomDocument->create_processing_instruction** (string content)

This function returns a new instance of class **DomCData**. The content of the pi is the value of the passed parameter. This node will not show up in the document unless it is inserted with e.g. **domnode_append_child()**.

The return value is **FALSE** if an error occured.

See also **domnode_append_child()**, **domdocument_create_element()**, **domdocument_create_text()**, **domdocument_create_cdata_section()**, **domdocument_create_attribute()**, **domdocument_create_entity_reference()**, and **domnode_insert_before()**.

DomDocument->create_text_node

(no version information, might be only in CVS)

DomDocument->create_text_node -- Create new text node

Description

object **DomDocument->create_text_node** (string content)

This function returns a new instance of class **DomText**. The content of the text is the value of the passed parameter. This node will not show up in the document unless it is inserted with e.g. **domnode_append_child()**.

The return value is **FALSE** if an error occured.

See also **domnode_append_child()**, **domdocument_create_element()**, **domdocument_create_comment()**, **domdocument_create_text()**, **domdocument_create_attribute()**, **domdocument_create_processing_instruction()**, **domdocument_create_entity_reference()**, and **domnode_insert_before()**.

DomDocument->doctype

(no version information, might be only in CVS)

DomDocument->doctype -- Returns the document type

Description

object **DomDocument->doctype** (void)

This function returns an object of class **DomDocumentType**. In versions of PHP before 4.3 this has been the class **Dtd**, but the DOM Standard does not know such a class.

See also the methods of class **DomDocumentType**.

DomDocument->document_element

(no version information, might be only in CVS)

DomDocument->document_element -- Returns root element node

Description

object **DomDocument->document_element** (void)

This function returns the root element node of a document.

The following example returns just the element with name CHAPTER and prints it. The other node -- the comment -- is not returned.

Example 1. Retrieving root element

```
<?php
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$root = $dom->document_element();
print_r($root);
?>
```

DomDocument->dump_file

(no version information, might be only in CVS)

DomDocument->dump_file -- Dumps the internal XML tree back into a file

Description

string **DomDocument->dump_file** (string filename [, bool compressionmode [, bool format]])

Creates an XML document from the dom representation. This function usually is called after building a new dom document from scratch as in the example below. The *format* specifies whether the output should be neatly formatted, or not. The first parameter specifies the name of the filename and the second parameter, whether it should be compressed or not.

Example 1. Creating a simple HTML document header

```
<?php
$doc = domxml_new_doc("1.0");
$root = $doc->create_element("HTML");
$root = $doc->append_child($root);
$head = $doc->create_element("HEAD");
$head = $root->append_child($head);
$title = $doc->create_element("TITLE");
$title = $head->append_child($title);
$text = $doc->create_text_node("This is the title");
$text = $title->append_child($text);
$doc->dump_file("/tmp/test.xml", false, true);
?>
```

See also **domdocument_dump_mem() domdocument_html_dump_mem()**.

DomDocument->dump_mem

(no version information, might be only in CVS)

DomDocument->dump_mem -- Dumps the internal XML tree back into a string

Description

string **DomDocument->dump_mem** ([bool format [, string encoding]])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Creates an XML document from the dom representation. This function usually is called after building a new dom document from scratch as in the example below. The *format* specifies whether the output should be neatly formatted, or not.

Example 1. Creating a simple HTML document header

```
<?php
$doc = domxml_new_doc("1.0");
$root = $doc->create_element("HTML");
$root = $doc->append_child($root);
$head = $doc->create_element("HEAD");
$head = $root->append_child($head);
$title = $doc->create_element("TITLE");
$title = $head->append_child($title);
$text = $doc->create_text_node("This is the title");
$text = $title->append_child($text);
echo "<PRE>";
echo htmlentities($doc->dump_mem(true));
echo "</PRE>";
?>
```

Note: The first parameter was added in PHP 4.3.0.

See also **domdocument_dump_file()**, **domdocument_html_dump_mem()**.

DomDocument->get_element_by_id

(no version information, might be only in CVS)

DomDocument->get_element_by_id -- Searches for an element with a certain id

Description

object **DomDocument->get_element_by_id** (string id)

This function is similar to **domdocument_get_elements_by_tagname()** but searches for an element with a given id. According to the DOM standard this requires a DTD which defines the attribute ID to be of type ID, though the current implementation simply does an xpath search for "//*[@ID = '%s']". This does not comply to the DOM standard which requires to return null if it is not known which attribute is of type id. This behaviour is likely to be fixed, so do not rely on the current behaviour.

See also **domdocument_get_elements_by_tagname()**

DomDocument->get_elements_by_tagname

(no version information, might be only in CVS)

DomDocument->get_elements_by_tagname --

Description

array **DomDocument->get_elements_by_tagname** (string name)

See also **domdocument_add_root()**

DomDocument->html_dump_mem

(no version information, might be only in CVS)

DomDocument->html_dump_mem -- Dumps the internal XML tree back into a string as HTML

Description

string **DomDocument->html_dump_mem** (void)

Creates an HTML document from the dom representation. This function usually is called after building a new dom document from scratch as in the example below.

Example 1. Creating a simple HTML document header

```
<?php
$doc = domxml_new_doc("1.0");
$root = $doc->create_element("HTML");
$root = $doc->append_child($root);
$head = $doc->create_element("HEAD");
$head = $root->append_child($head);
$title = $doc->create_element("TITLE");
$title = $head->append_child($title);
$text = $doc->create_text_node("This is the title");
$text = $title->append_child($text);
echo "<PRE>";
echo htmlentities($doc->html_dump_mem());
echo "</PRE>";
?>
```

See also **domdocument_dump_file()**, **domdocument_html_dump_mem()**.

DomDocument->xinclude

(no version information, might be only in CVS)

DomDocument->xinclude -- Substitutes XIncludes in a DomDocument Object.

Description

int **DomDocument->xinclude** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

DomDocumentType->entities

(no version information, might be only in CVS)

DomDocumentType->entities -- Returns list of entities

Description

array **DomDocumentType->entities** (void)

Warning
This function is currently not documented; only the argument list is available.

DomDocumentType->internal_subset

(no version information, might be only in CVS)

DomDocumentType->internal_subset -- Returns internal subset

Description

bool **DomDocumentType->internal_subset** (void)

Warning
This function is currently not documented; only the argument list is available.

DomDocumentType->name

(no version information, might be only in CVS)

DomDocumentType->name -- Returns name of document type

Description

string **DomDocumentType->name** (void)

This function returns the name of the document type.

DomDocumentType->notations

(no version information, might be only in CVS)

DomDocumentType->notations -- Returns list of notations

Description

array **DomDocumentType->notations** (void)

Warning
This function is currently not documented; only the argument list is available.

DomDocumentType->public_id

(no version information, might be only in CVS)

DomDocumentType->public_id -- Returns public id of document type

Description

string **DomDocumentType->public_id** (void)

This function returns the public id of the document type.

The following example echos nothing.

Example 1. Retrieving the public id

```
<?php
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$doctype = $dom->doctype();
echo $doctype->public_id();
?>
```

DomDocumentType->system_id

(no version information, might be only in CVS)

DomDocumentType->system_id -- Returns system id of document type

Description

string **DomDocumentType->system_id** (void)

Returns the system id of the document type.

The following example echos '/share/sgml/Norman_Walsh/db3xml10/db3xml10.dtd'.

Example 1. Retrieving the system id

```
<?php
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$doctype = $dom->doctype();
echo $doctype->system_id();
?>
```

DomElement->get_attribute_node

(no version information, might be only in CVS)

DomElement->get_attribute_node -- Returns value of attribute

Description

object **DomElement->get_attribute_node** (object attr)

Warning
This function is currently not documented; only the argument list is available.

DomElement->get_attribute

(no version information, might be only in CVS)

DomElement->get_attribute -- Returns value of attribute

Description

object **DomElement->get_attribute** (string name)

Returns the attribute with name *name* of the current node.

(PHP >= 4.3 only) If no attribute with given name is found, an empty string is returned.

See also **domelement_set_attribute()**

DomElement->get_elements_by_tagname

(no version information, might be only in CVS)

DomElement->get_elements_by_tagname -- Gets elements by tagname

Description

bool **DomElement->get_elements_by_tagname** (string name)

Warning
This function is currently not documented; only the argument list is available.

DomElement->has_attribute

(no version information, might be only in CVS)

DomElement->has_attribute -- Checks to see if attribute exists

Description

bool **DomElement->has_attribute** (string name)

Warning
This function is currently not documented; only the argument list is available.

DomElement->remove_attribute

(no version information, might be only in CVS)

DomElement->remove_attribute -- Removes attribute

Description

bool **DomElement->remove_attribute** (string name)

Warning
This function is currently not documented; only the argument list is available.

DomElement->set_attribute

(no version information, might be only in CVS)

DomElement->set_attribute -- Adds new attribute

Description

bool **DomElement->set_attribute** (string name, string value)

Sets an attribute with name *name* ot the given value. If the attribute does not exist, it will be created.

Example 1. Setting an attribute

```
<?php
$doc = domxml_new_doc("1.0");
$node = $doc->create_element("para");
$newnode = $doc->append_child($node);
$newnode->set_attribute("align", "left");
?>
```

See also **domelement_get_attribute()**

DomElement->tagname

(no version information, might be only in CVS)

DomElement->tagname -- Returns name of element

Description

string **DomElement->tagname** (void)

Warning
This function is currently not documented; only the argument list is available.

DomNode->add_namespace

(no version information, might be only in CVS)

DomNode->add_namespace -- Adds a namespace declaration to a node.

Description

bool **DomNode->add_namespace** (string uri, string prefix)

See also **domdocument_create_element_ns()**, **domnode_set_namespace()**

DomNode->append_child

(no version information, might be only in CVS)

DomNode->append_child -- Adds new child at the end of the children

Description

object **DomNode->append_child** (object newnode)

This functions appends a child to an existing list of children or creates a new list of children. The child can be created with e.g. **domdocument_create_element()**, **domdocument_create_text()** etc. or simply by using any other node.

(PHP < 4.3) Before a new child is appended it is first duplicated. Therefore the new child is a completely new copy which can be modified without changing the node which was passed to this function. If the node passed has children itself, they will be duplicated as well, which makes it quite easy to duplicate large parts of a xml document. The return value is the appended child. If you plan to do further modifications on the appended child you must use the returned node.

(PHP >= 4.3) The new child *newnode* is first unlinked from its existing context, if it already existed in a document. Therefore the node is moved and not copies anymore. This is the behaviour according to the W3C specifications. If you want to duplicate large parts of a xml document, use DomNode->clone_node() before appending.

The following example will add a new element node to a fresh document and sets the attribute "align" to "left".

Example 1. Adding a child

```
<?php
$doc = domxml_new_doc("1.0");
$node = $doc->create_element("para");
$newnode = $doc->append_child($node);
$newnode->set_attribute("align", "left");
?>
```

The above example could also be written as the following:

Example 2. Adding a child

```
<?php
$doc = domxml_new_doc("1.0");
$node = $doc->create_element("para");
$node->set_attribute("align", "left");
$newnode = $doc->append_child($node);
?>
```

A more comples example is the one below. It first searches for a certain element, duplicates it including its children and adds it as a sibling. Finally a new attribute is added to one of the children of the new sibling and the whole document is dumped.

Example 3. Adding a child

```
<?php
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$elements = $dom->get_elements_by_tagname("informaltable");
print_r($elements);
$element = $elements[0];

$parent = $element->parent_node();
$newnode = $parent->append_child($element);
$children = $newnode->children();
$attr = $children[1]->set_attribute("align", "left");
```

```
echo "<PRE>";
$xmlfile = $dom->dump_mem();
echo htmlentities($xmlfile);
echo "</PRE>";
?>
```

The above example could also be done with **domnode_insert_before()** instead of **domnode_append_child()**.

See also **domnode_insert_before()**, **domnode_clone_node()**.

DomNode->append_sibling

(no version information, might be only in CVS)

DomNode->append_sibling -- Adds new sibling to a node

Description

object **DomNode->append_sibling** (object newnode)

This functions appends a sibling to an existing node. The child can be created with e.g. **domdocument_create_element()**, **domdocument_create_text()** etc. or simply by using any other node.

Before a new sibling is added it is first duplicated. Therefore the new child is a completely new copy which can be modified without changing the node which was passed to this function. If the node passed has children itself, they will be duplicated as well, which makes it quite easy to duplicate large parts of a xml document. The return value is the added sibling. If you plan to do further modifications on the added sibling you must use the returned node.

This function has been added to provide the behaviour of **domnode_append_child()** as it works till PHP 4.2.

See also **domnode_append_before()**.

DomNode->attributes

(no version information, might be only in CVS)

DomNode->attributes -- Returns list of attributes

Description

array **DomNode->attributes** (void)

This function only returns an array of attributes if the node is of type XML_ELEMENT_NODE.

(PHP >= 4.3 only) If no attributes are found, NULL is returned.

DomNode->child_nodes

(no version information, might be only in CVS)

DomNode->child_nodes -- Returns children of node

Description

array **DomNode->child_nodes** (void)

Returns all children of the node.

See also **domnode_next_sibling()**, **domnode_previous_sibling()**.

DomNode->clone_node

(no version information, might be only in CVS)

DomNode->clone_node -- Clones a node

Description

object **DomNode->clone_node** (void)

Warning
This function is currently not documented; only the argument list is available.

DomNode->dump_node

(no version information, might be only in CVS)

DomNode->dump_node -- Dumps a single node

Description

string **DomNode->dump_node** (void)

Warning
This function is currently not documented; only the argument list is available.

See also **domdocument_dump_mem()**.

DomNode->first_child

(no version information, might be only in CVS)

DomNode->first_child -- Returns first child of node

Description

bool **DomNode->first_child** (void)

Returns the first child of the node.

(PHP >= 4.3 only) If no first child is found, NULL is returned.

See also **domnode_last_child()**, **domnode_next_sibling()**, **domnode_previous_sibling()**.

DomNode->get_content

(no version information, might be only in CVS)

DomNode->get_content -- Gets content of node

Description

string **DomNode->get_content** (void)

Warning
This function is currently not documented; only the argument list is available.

DomNode->has_attributess

(no version information, might be only in CVS)

DomNode->has_attributess -- Checks if node has attributes

Description

bool **DomNode->has_attributes** (void)

This function checks if the node has attributes.

See also **domnode_has_child_nodes()**.

DomNode->has_child_nodes

(no version information, might be only in CVS)

DomNode->has_child_nodes -- Checks if node has children

Description

bool **DomNode->has_child_nodes** (void)

This function checks if the node has children.

See also **domnode_child_nodes()**.

DomNode->insert_before

(no version information, might be only in CVS)

DomNode->insert_before -- Inserts new node as child

Description

object **DomNode->insert_before** (object newnode, object refnode)

This function inserts the new node *newnode* right before the node *refnode*. The return value is the inserted node. If you plan to do further modifications on the appended child you must use the returned node.

(PHP >= 4.3 only) If *newnode* already is part of a document, it will be first unlinked from its existing context. If *refnode* is NULL, then *newnode* will be inserted at the end of the list of children.

domnode_insert_before() is very similar to **domnode_append_child()** as the following example shows which does the same as the example at **domnode_append_child()**.

Example 1. Adding a child

```
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$elements = $dom->get_elements_by_tagname("informaltable");
print_r($elements);
$element = $elements[0];

$newnode = $element->insert_before($element, $element);
$children = $newnode->children();
$attr = $children[1]->set_attribute("align", "left");

echo "<PRE>";
$xmlfile = $dom->dump_mem();
echo htmlentities($xmlfile);
echo "</PRE>";
```

See also **domnode_append_child()**.

DomNode->is_blank_node

(no version information, might be only in CVS)

DomNode->is_blank_node -- Checks if node is blank

Description

bool **DomNode->is_blank_node** (void)

Warning
This function is currently not documented; only the argument list is available.

DomNode->last_child

(no version information, might be only in CVS)

DomNode->last_child -- Returns last child of node

Description

object **DomNode->last_child** (void)

Returns the last child of the node.

(PHP >= 4.3 only) If no last child is found, NULL is returned.

See also **domnode_first_child()**, **domnode_next_sibling()**, **domnode_previous_sibling()**.

DomNode->next_sibling

(no version information, might be only in CVS)

DomNode->next_sibling -- Returns the next sibling of node

Description

object **DomNode->next_sibling** (void)

This function returns the next sibling of the current node. If there is no next sibling it returns **FALSE** (< 4.3) or null (>= 4.3). You can use this function to iterate over all children of a node as shown in the example.

Example 1. Iterate over children

```
<?php
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$elements = $dom->get_elements_by_tagname("tbody");
$element = $elements[0];
$child = $element->first_child();

while($child) {
   print_r($child);
   $child = $child->next_sibling();
}
?>
```

See also **domnode_previous_sibling()**.

DomNode->node_name

(no version information, might be only in CVS)

DomNode->node_name -- Returns name of node

Description

string **DomNode->node_name** (void)

Returns name of the node. The name has different meanings for the different types of nodes as illustrated in the following table.

Table 1. Meaning of value

Type	Meaning
DomAttribute	value of attribute
DomAttribute	
DomCDataSection	#cdata-section
DomComment	#comment
DomDocument	#document
DomDocumentType	document type name
DomElement	tag name
DomEntity	name of entity
DomEntityReference	name of entity reference
DomNotation	notation name
DomProcessingInstruction	target
DomText	#text

DomNode->node_type

(no version information, might be only in CVS)

DomNode->node_type -- Returns type of node

Description

int **DomNode->node_type** (void)

Returns the type of the node. All possible types are listed in the table in the introduction.

DomNode->node_value

(no version information, might be only in CVS)

DomNode->node_value -- Returns value of a node

Description

string **DomNode->node_value** (void)

Returns value of the node. The value has different meanings for the different types of nodes as illustrated in the following table.

Table 1. Meaning of value

Type	Meaning
DomAttribute	value of attribute
DomAttribute	
DomCDataSection	content
DomComment	content of comment

Type	Meaning
DomDocument	null
DomDocumentType	null
DomElement	null
DomEntity	null
DomEntityReference	null
DomNotation	null
DomProcessingInstruction	entire content without target
DomText	content of text

DomNode->owner_document

(no version information, might be only in CVS)

DomNode->owner_document -- Returns the document this node belongs to

Description

object **DomNode->owner_document** (void)

This function returns the document the current node belongs to.

The following example will create two identical lists of children.

Example 1. Finding the document of a node

```
<?php
$doc = domxml_new_doc("1.0");
$node = $doc->create_element("para");
$node = $doc->append_child($node);
$children = $doc->children();
print_r($children);

$doc2 = $node->owner_document();
$children = $doc2->children();
print_r($children);
?>
```

See also **domnode_insert_before()**.

DomNode->parent_node

(no version information, might be only in CVS)

DomNode->parent_node -- Returns the parent of the node

Description

object **DomNode->parent_node** (void)

This function returns the parent node.

(PHP >= 4.3 only) If no parent is found, NULL is returned.

The following example will show two identical lists of children.

Example 1. Finding the document of a node

```
<?php
$doc = domxml_new_doc("1.0");
$node = $doc->create_element("para");
$node = $doc->append_child($node);
$children = $doc->children();
print_r($children);

$doc2 = $node->parent_node();
$children = $doc2->children();
```

```
print_r($children);
?>
```

DomNode->prefix

(no version information, might be only in CVS)

DomNode->prefix -- Returns name space prefix of node

Description

string **DomNode->prefix** (void)

Returns the name space prefix of the node.

DomNode->previous_sibling

(no version information, might be only in CVS)

DomNode->previous_sibling -- Returns the previous sibling of node

Description

object **DomNode->previous_sibling** (void)

This function returns the previous sibling of the current node. If there is no previous sibling it returns **FALSE** (< 4.3) or **NULL** (>= 4.3). You can use this function to iterate over all children of a node as shown in the example.

See also **domnode_next_sibling()**.

DomNode->remove_child

(no version information, might be only in CVS)

DomNode->remove_child -- Removes child from list of children

Description

object **DomNode->remove_child** (object oldchild)

This functions removes a child from a list of children. If child cannot be removed or is not a child the function will return **FALSE**. If the child could be removed the functions returns the old child.

Example 1. Removing a child

```
<?php
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$elements = $dom->get_elements_by_tagname("tbody");
$element = $elements[0];
$children = $element->child_nodes();
$child = $element->remove_child($children[0]);

echo "<PRE>";
$xmlfile = $dom->dump_mem(true);
echo htmlentities($xmlfile);
echo "</PRE>";
?>
```

See also **domnode_append_child()**.

DomNode->replace_child

(no version information, might be only in CVS)

DomNode->replace_child -- Replaces a child

Description

object **DomNode->replace_child** (object oldnode, object newnode)

(PHP 4.2) This function replaces the child *oldnode* with the passed new node. If the new node is already a child it will not be added a second time. If the old node cannot be found the function returns **FALSE**. If the replacement succeds the old node is returned.

(PHP 4.3) This function replaces the child *oldnode* with the passed *newnode*, even if the new node already is a child of the DomNode. If *newnode* was already inserted in the document it is first unlinked from its existing context. If the old node cannot be found the function returns **FALSE**. If the replacement succeds the old node is returned. (This behaviour is according to the W3C specs).

See also **domnode_append_child()**

DomNode->replace_node

(no version information, might be only in CVS)

DomNode->replace_node -- Replaces node

Description

object **DomNode->replace_node** (object newnode)

(PHP 4.2) This function replaces an existing node with the passed new node. Before the replacement *newnode* is copied if it has a parent to make sure a node which is already in the document will not be inserted a second time. This behaviour enforces doing all modifications on the node before the replacement or to refetch the inserted node afterwards with functions like **domnode_first_child()**, **domnode_child_nodes()** etc..

(PHP 4.3) This function replaces an existing node with the passed new node. It is not copied anymore. If *newnode* was already inserted in the document it is first unlinked from its existing context. If the replacement succeds the old node is returned.

See also **domnode_append_child()**

DomNode->set_content

(no version information, might be only in CVS)

DomNode->set_content -- Sets content of node

Description

bool **DomNode->set_content** (void)

Warning
This function is currently not documented; only the argument list is available.

DomNode->set_name

(no version information, might be only in CVS)

DomNode->set_name -- Sets name of node

Description

bool **DomNode->set_name** (void)

Sets name of node.

See also **domnode_node_name()**.

DomNode->set_namespace

(no version information, might be only in CVS)

DomNode->set_namespace -- Sets namespace of a node.

Description

void **DomNode->set_namespace** (string uri [, string prefix])

Sets the namespace of a node to *uri*. If there is already a namespace declaration with the same uri in one of the parent nodes of the node, the prefix of this is taken, otherwise it will take the one provided in the optional parameter *prefix* or generate a random one.

See also **domdocument_create_element_ns()**, **domnode_add_namespace()**

DomNode->unlink_node

(no version information, might be only in CVS)

DomNode->unlink_node -- Deletes node

Description

object **DomNode->unlink_node** (void)

Warning
This function is currently not documented; only the argument list is available.

DomProcessingInstruction->data

(no version information, might be only in CVS)

DomProcessingInstruction->data -- Returns data of pi node

Description

string **DomProcessingInstruction->data** (void)

Warning
This function is currently not documented; only the argument list is available.

DomProcessingInstruction->target

(no version information, might be only in CVS)

DomProcessingInstruction->target -- Returns target of pi node

Description

string **DomProcessingInstruction->target** (void)

Warning
This function is currently not documented; only the argument list is available.

DomXsltStylesheet->process

(no version information, might be only in CVS)

DomXsltStylesheet->process -- Applies the XSLT-Transformation on a DomDocument Object.

Description

object **DomXsltStylesheet->process** (object DomDocument [, array xslt_parameters [, bool param_is_xpath]])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

See also **domxml_xslt_stylesheet()**, **domxml_xslt_stylesheet_file()**, **domxml_xslt_stylesheet_doc()**

DomXsltStylesheet->result_dump_file

(no version information, might be only in CVS)

DomXsltStylesheet->result_dump_file -- Dumps the result from a XSLT-Transformation into a file

Description

string **DomXsltStylesheet->result_dump_file** (object DomDocument, string filename)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

This function is only available since PHP 4.3

Since DomXsltStylesheet->process() always returns a well-formed XML DomDocument, no matter what output method was declared in <xsl:output> and similar attributes/elements, it's of not much use, if you want to output HTML 4 or text data. This function on the contrary honors <xsl:output method="html|text"> and other output control directives. See the example for instruction of how to use it.

Example 1. Saving the result of a XSLT transformation in a file

```
<?php
$filename = "stylesheet.xsl";
$xmldoc = domxml_open_file("data.xml");
$xsldoc = domxml_xslt_stylesheet_file($filename);
$result =  $xsldoc->process($xmldoc);
print $xsldoc->result_dump_file($result,"filename");
?>
```

See also **domxml_xslt_result_dump_mem()**, **domxml_xslt_process()**

DomXsltStylesheet->result_dump_mem

(no version information, might be only in CVS)

DomXsltStylesheet->result_dump_mem -- Dumps the result from a XSLT-Transformation back into a string

Description

string **DomXsltStylesheet->result_dump_mem** (object DomDocument)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

This function is only available since PHP 4.3

Since DomXsltStylesheet->process() always returns a well-formed XML DomDocument, no matter what output method was declared in <xsl:output> and similar attributes/elements, it's of not much use, if you want to output HTML 4 or text data. This function on the contrary honors <xsl:output method="html|text"> and other output control directives. See the example for instruction of how to use it.

Example 1. Outputting the result of a XSLT transformation

```
<?php
$filename = "stylesheet.xsl";
$xmldoc = domxml_open_file("data.xml");
$xsldoc = domxml_xslt_stylesheet_file($filename);
$result =  $xsldoc->process($xmldoc);
print $xsldoc->result_dump_mem($result);
?>
```

See also **domxml_xslt_result_dump_file()**, **domxml_xslt_process()**

domxml_new_doc

(PHP 4 >= 4.2.1)

domxml_new_doc -- Creates new empty XML document

Description

object **domxml_new_doc** (string version)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Creates a new dom document from scratch and returns it.

See also **domdocument_add_root()**

domxml_open_file

(PHP 4 >= 4.2.1)

domxml_open_file -- Creates a DOM object from XML file

Description

object **domxml_open_file** (string filename)

The function parses the XML document in the file named *filename* and returns an object of class "Dom document", having the properties as listed above. The file is accessed read-only.

Example 1. Opening a xml document from a file

```
<?php

if(!$dom = domxml_open_file("example.xml")) {
  echo "Error while parsing the document\n";
  exit;
}

$root = $dom->document_element();
?>
```

See also **domxml_open_mem()**, **domxml_new_doc()**.

domxml_open_mem

(PHP 4 >= 4.2.1)

domxml_open_mem -- Creates a DOM object of an XML document

Description

object **domxml_open_mem** (string str)

The function parses the XML document in *str* and returns an object of class "Dom document", having the properties as listed above. This function, **domxml_open_file()** or **domxml_new_doc()** must be called before any other function calls.

Example 1. Opening a xml document in a string

```
<?php
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$root = $dom->document_element();
?>
```

See also **domxml_open_file()**, **domxml_new_doc()**.

domxml_version

(PHP 4 >= 4.1.0)

domxml_version -- Get XML library version

Description

string **domxml_version** (void)

This function returns the version of the XML library version currently used.

domxml_xmltree

(PHP 4 >= 4.2.1)

domxml_xmltree -- Creates a tree of PHP objects from an XML document

Description

object **domxml_xmltree** (string str)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

The function parses the XML document in *str* and returns a tree PHP objects as the parsed document. This function is isolated from the other functions, which means you cannot access the tree with any of the other functions. Modifying it, for example by adding nodes, makes no sense since there is currently no way to dump it as an XML file. However this function may be valuable if you want to read a file and investigate the content.

domxml_xslt_stylesheet_doc

(PHP 4 >= 4.2.0)

domxml_xslt_stylesheet_doc -- Creates a DomXsltStylesheet Object from a DomDocument Object.

Description

object **domxml_xslt_stylesheet_doc** (object DocDocument Object)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

See also **domxsltstylesheet->process()**, domxml_xslt_stylesheet(), domxml_xslt_stylesheet_file()

domxml_xslt_stylesheet_file

(PHP 4 >= 4.2.0)

domxml_xslt_stylesheet_file -- Creates a DomXsltStylesheet Object from a xsl document in a file.

Description

object **domxml_xslt_stylesheet_file** (string xsl file)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

See also **domxsltstylesheet->process()**, domxml_xslt_stylesheet(), domxml_xslt_stylesheet_doc()

domxml_xslt_stylesheet

(PHP 4 >= 4.2.0)

domxml_xslt_stylesheet -- Creates a DomXsltStylesheet Object from a xml document in a string.

Description

object **domxml_xslt_stylesheet** (string xsl document)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

See also **domxsltstylesheet->process()**, **domxml_xslt_stylesheet_file()**, **domxml_xslt_stylesheet_doc()**

xpath_eval_expression

(PHP 4 >= 4.0.4)

xpath_eval_expression -- Evaluates the XPath Location Path in the given string

Description

array **xpath_eval_expression** (object xpath_context)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

See also **xpath_eval()**

xpath_eval

(PHP 4 >= 4.0.4)

xpath_eval -- Evaluates the XPath Location Path in the given string

Description

array **xpath_eval** (object xpath context, string xpath expression [, object contextnode])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

The optional *contextnode* can be specified for doing relative XPath queries.

See also **xpath_new_context()**

xpath_new_context

(PHP 4 >= 4.0.4)

xpath_new_context -- Creates new xpath context

Description

object **xpath_new_context** (object dom document)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

See also **xpath_eval()**

xptr_eval

(PHP 4 >= 4.0.4)

xptr_eval -- Evaluate the XPtr Location Path in the given string

Description

int **xptr_eval** ([object xpath_context, string eval_str])

Warning
This function is currently not documented; only the argument list is available.

xptr_new_context

(PHP 4 >= 4.0.4)

xptr_new_context -- Create new XPath Context

Description

string **xptr_new_context** ([object doc_handle])

Warning
This function is currently not documented; only the argument list is available.

XXVI. .NET functions

Introduction

Warning
This extension is *EXPERIMENTAL*. The behaviour of this extension -- including the names of its functions and anything else documented about this extension -- may change without notice in a future release of PHP. Use this extension at your own risk.

Table of Contents

dotnet_load

(no version information, might be only in CVS)

dotnet_load -- Loads a DOTNET module

Description

int **dotnet_load** (string assembly_name [, string datatype_name [, int codepage]])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

XXVII. Error Handling and Logging Functions

Introduction

These are functions dealing with error handling and logging. They allow you to define your own error handling rules, as well as modify the way the errors can be logged. This allows you to change and enhance error reporting to suit your needs.

With the logging functions, you can send messages directly to other machines, to an email (or email to pager gateway!), to system logs, etc., so you can selectively log and monitor the most important parts of your applications and websites.

The error reporting functions allow you to customize what level and kind of error feedback is given, ranging from simple notices to customized functions returned during errors.

Requirements

No external libraries are needed to build this extension.

Installation

There is no installation needed to use these functions; they are part of the PHP core.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. Errors and Logging Configuration Options

Name	Default	Changeable
error_reporting	E_ALL & ~E_NOTICE	PHP_INI_ALL
display_errors	"1"	PHP_INI_ALL
display_startup_errors	"0"	PHP_INI_ALL
log_errors	"0"	PHP_INI_ALL
log_errors_max_len	"1024"	PHP_INI_ALL
ignore_repeated_errors	"0"	PHP_INI_ALL
ignore_repeated_source	"0"	PHP_INI_ALL
report_memleaks	"1"	PHP_INI_SYSTEM
track_errors	"0"	PHP_INI_ALL
html_errors	"1"	PHP_INI_ALL
docref_root	"http://www.php.net/"	PHP_INI_ALL
docref_ext	".html"	PHP_INI_ALL
error_prepend_string	NULL	PHP_INI_ALL
error_append_string	NULL	PHP_INI_ALL
error_log	NULL	PHP_INI_ALL
warn_plus_overloading	NULL	PHP_INI??

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Here is a short explanation of the configuration directives.

`error_reporting` **integer**

Set the error reporting level. The parameter is either an integer representing a bit field, or named constants. The error_reporting levels and constants are described in Predefined Constants, and in `php.ini`. To set at runtime, use the **error_reporting()** function. See also the display_errors directive.

In PHP 4 the default value does not show **`E_NOTICE`** level errors. You may want to show them during development.

> **Note:** Enabling **`E_NOTICE`** during development has some benefits. For debugging purposes: NOTICE messages will warn you about possibls bugs in your code. For example, use of unassigned values are warned. It is extremely useful to find typos and to save time for debugging. NOTICE messages will warn you about bad style. For example, $arr[item] is better to be written as $arr['item'] since PHP tries to treat "item" as constant. If it is not a constant, PHP assumes it is a string index for the array.

In PHP 3, the default setting is `(E_ERROR | E_WARNING | E_PARSE)`, meaning the same thing. Note, however, that since

constants are not supported in PHP 3's `php3.ini`, the error_reporting setting there must be numeric; hence, it is `7`.

`display_errors` **boolean**

This determines whether errors should be printed to the screen as part of the HTML output or not.

`display_startup_errors` **boolean**

Even when display_errors is on, errors that occur during PHP's startup sequence are not displayed. It's strongly recommended to keep display_startup_errors off, except for debugging.

`log_errors` **boolean**

Tells whether script error messages should be logged to the server's error log or error_log. This option is thus server-specific.

Note: You're strongly advised to use error logging in place of error displaying on production web sites.

`log_errors_max_len` **integer**

Set the maximum length of log_errors in kbytes. In error_log information about the source is added. The default is 1024 and 0 allows to not apply any maximum length at all.

`ignore_repeated_errors` **boolean**

Do not log repeated messages. Repeated errors must occur in the same file on the same line until ignore_repeated_source is set true.

`ignore_repeated_source` **boolean**

Ignore source of message when ignoring repeated messages. When this setting is On you will not log errors with repeated messages from different files or sourcelines.

`report_memleaks` **boolean**

If this parameter is set to Off, then memory leaks will not be shown (on stdout or in the log). This has only effect in a debug compile, and if error_reporting includes E_WARNING in the allowed list

`track_errors` **boolean**

If enabled, the last error message will always be present in the variable $php_errormsg.

`html_errors` **boolean**

Turn off HTML tags in error messages. The new format for html errors produces clickable messages that direct the user to a page describing the error or function in causing the error. These references are affected by docref_root and docref_ext.

`docref_root` **string**

The new error format contains a reference to a page describing the error or function in causing the error. In case of manual pages you can download the manual in your language and set this ini directive to the url of your local copy. If your local copy of the manual can be reached by '/manual/' you can simply use **`docref_root=/manual/`**. Additional you have to set docref_ext to match the fileextensions of your copy **`docref_ext=.html`**. It is possible to use external references. For example you can use **`docref_root=http://manual/en/`** or **`docref_root="http://landonize.it/?how=url&theme=classic&filter=Landon&url=http%3A%2F%2Fwww.php.net%2F"`**

Most of the time you want the docref_root value to end with a slash '/'. But see the second example above which does not have nor need it.

`docref_ext` **string**

See docref_root.

Note: The value of docref_ext must begin with a dot '.'.

`error_prepend_string` **string**

String to output before an error message.

`error_append_string` **string**

String to output after an error message.

`error_log` **string**

Name of the file where script errors should be logged. If the special value `syslog` is used, the errors are sent to the system logger instead. On UNIX, this means syslog(3) and on Windows NT it means the event log. The system logger is not supported on Windows 95. See also: **syslog()**.

`warn_plus_overloading` **boolean**

If enabled, this option makes PHP output a warning when the plus (+) operator is used on strings. This is to make it easier to find scripts that need to be rewritten to using the string concatenator instead (`.`).

Predefined Constants

The constants below are always available as part of the PHP core.

Note: You may use these constant names in `php.ini` but not outside of PHP, like in `httpd.conf`, where you'd use the bitmask values instead.

Table 2. Errors and Logging

Value	Constant	Description	Note
1	**`E_ERROR`** (**integer**)	Fatal run-time errors. These indicate errors that can not be recovered from, such as a memory allocation problem. Execution of the script is halted.	
2	**`E_WARNING`** (**integer**)	Run-time warnings (non-fatal errors). Execution of the script is not halted.	
4	**`E_PARSE`** (**integer**)	Compile-time parse errors. Parse errors should only be generated by the parser.	
8	**`E_NOTICE`** (**integer**)	Run-time notices. Indicate that the script encountered something that could indicate an error, but could also happen in the normal course of running a script.	
16	**`E_CORE_ERROR`** (**integer**)	Fatal errors that occur during PHP's initial startup. This is like an **`E_ERROR`**, except it is generated by the core of PHP.	PHP 4 only
32	**`E_CORE_WARNING`** (**integer**)	Warnings (non-fatal errors) that occur during PHP's initial startup. This is like an **`E_WARNING`**, except it is generated by the core of PHP.	PHP 4 only
64	**`E_COMPILE_ERROR`** (**integer**)	Fatal compile-time errors. This is like an **`E_ERROR`**, except it is generated by the Zend Scripting Engine.	PHP 4 only
128	**`E_COMPILE_WARNING`** (**integer**)	Compile-time warnings (non-fatal errors). This is like an **`E_WARNING`**, except it is generated by the Zend Scripting Engine.	PHP 4 only
256	**`E_USER_ERROR`** (**integer**)	User-generated error message. This is like an **`E_ERROR`**, except it is generated in PHP code by using the PHP function **trigger_error()**.	PHP 4 only
512	**`E_USER_WARNING`** (**integer**)	User-generated warning message. This is like an **`E_WARNING`**, except it is generated in PHP code by using the PHP function **trigger_error()**.	PHP 4 only
1024	**`E_USER_NOTICE`** (**integer**)	User-generated notice message. This is like an **`E_NOTICE`**, except it is generated in PHP code by using the PHP function **trigger_error()**.	PHP 4 only
2047	**`E_ALL`** (**integer**)	All errors and warnings, as supported.	

The above values (either numerical or symbolic) are used to build up a bitmask that specifies which errors to report. You can use the bitwise operators to combine these values or mask out certain types of errors. Note that only '|', '~', '!', and '&' will be understood within `php.ini`, however, and that no bitwise operators will be understood within `php3.ini`.

Examples

Below we can see an example of using the error handling capabilities in PHP. We define a error handling function which logs the information into a file (using an XML format), and e-mails the developer in case a critical error in the logic happens.

Example 1. Using error handling in a script

```
<?php
// we will do our own error handling
error_reporting(0);

// user defined error handling function
function userErrorHandler ($errno, $errmsg, $filename, $linenum, $vars) {
    // timestamp for the error entry
    $dt = date("Y-m-d H:i:s (T)");

    // define an assoc array of error string
    // in reality the only entries we should
```

```
    // consider are 2,8,256,512 and 1024
    $errortype = array (
                1   =>  "Error",
                2   =>  "Warning",
                4   =>  "Parsing Error",
                8   =>  "Notice",
                16  =>  "Core Error",
                32  =>  "Core Warning",
                64  =>  "Compile Error",
                128 =>  "Compile Warning",
                256 =>  "User Error",
                512 =>  "User Warning",
                1024=>  "User Notice"
                );
    // set of errors for which a var trace will be saved
    $user_errors = array(E_USER_ERROR, E_USER_WARNING, E_USER_NOTICE);

    $err = "<errorentry>\n";
    $err .= "\t<datetime>".$dt."</datetime>\n";
    $err .= "\t<errornum>".$errno."</errornum>\n";
    $err .= "\t<errortype>".$errortype[$errno]."</errortype>\n";
    $err .= "\t<errormsg>".$errmsg."</errormsg>\n";
    $err .= "\t<scriptname>".$filename."</scriptname>\n";
    $err .= "\t<scriptlinenum>".$linenum."</scriptlinenum>\n";

    if (in_array($errno, $user_errors))
        $err .= "\t<vartrace>".wddx_serialize_value($vars,"Variables")."</vartrace>\n";
    $err .= "</errorentry>\n\n";

    // for testing
    // echo $err;

    // save to the error log, and e-mail me if there is a critical user error
    error_log($err, 3, "/usr/local/php4/error.log");
    if ($errno == E_USER_ERROR)
        mail("phpdev@example.com","Critical User Error",$err);
}


function distance ($vect1, $vect2) {
    if (!is_array($vect1) || !is_array($vect2)) {
        trigger_error("Incorrect parameters, arrays expected", E_USER_ERROR);
        return NULL;
    }

    if (count($vect1) != count($vect2)) {
        trigger_error("Vectors need to be of the same size", E_USER_ERROR);
        return NULL;
    }

    for ($i=0; $i<count($vect1); $i++) {
        $c1 = $vect1[$i]; $c2 = $vect2[$i];
        $d = 0.0;
        if (!is_numeric($c1)) {
            trigger_error("Coordinate $i in vector 1 is not a number, using zero",
                          E_USER_WARNING);
            $c1 = 0.0;
        }
        if (!is_numeric($c2)) {
            trigger_error("Coordinate $i in vector 2 is not a number, using zero",
                          E_USER_WARNING);
            $c2 = 0.0;
        }
        $d += $c2*$c2 - $c1*$c1;
    }
    return sqrt($d);
}

$old_error_handler = set_error_handler("userErrorHandler");

// undefined constant, generates a warning
$t = I_AM_NOT_DEFINED;

// define some "vectors"
$a = array(2,3,"foo");
$b = array(5.5, 4.3, -1.6);
$c = array (1,-3);

// generate a user error
$t1 = distance($c,$b)."\n";

// generate another user error
$t2 = distance($b,"i am not an array")."\n";

// generate a warning
$t3 = distance($a,$b)."\n";

?>
```

See Also

See also **syslog()**.

Table of Contents

debug_backtrace

(PHP 4 >= 4.3.0)

debug_backtrace -- Generates a backtrace

Description

array **debug_backtrace** (void)

debug_backtrace() generates a PHP backtrace and returns this information as an associative **array**. The possible returned elements are listed in the following table:

Table 1. Possible returned elements from debug_backtrace()

Name	Type	Description
function	**string**	The current function name. See also __FUNCTION__.
line	**integer**	The current line number. See also __LINE__.
file	**string**	The current file name. See also __FILE__.
class	**string**	The current class name. See also __CLASS__
type	**string**	The current class type.
args	**array**	If inside a function, this lists the functions arguments. If inside a included file, this lists the included file name(s).

The following is a simple example.

Example 1. debug_backtrace() example

```
// filename: a.php
<?php

function a_test($str) {

    print "\nHi: $str";

    var_dump(debug_backtrace());
}

a_test('friend');
?>

// filename: b.php
<?php
include_once '/tmp/a.php';
?>

/* Results when executing /tmp/b.php

Hi: friend
array(2) {
  [0]=>
  array(4) {
    ["file"] => string(10) "/tmp/a.php"
    ["line"] => int(10)
    ["function"] => string(6) "a_test"
    ["args"]=>
    array(1) {
      [0] => &string(6) "friend"
    }
  }
  [1]=>
  array(4) {
    ["file"] => string(10) "/tmp/b.php"
```

```
    ["line"] => int(2)
    ["args"] =>
    array(1) {
      [0] => string(10) "/tmp/a.php"
    }
    ["function"] => string(12) "include_once"
  }
}
*/
```

See also **trigger_error()**.

error_log

(PHP 3, PHP 4)

error_log -- send an error message somewhere

Description

int **error_log** (string message [, int message_type [, string destination [, string extra_headers]]])

Sends an error message to the web server's error log, a TCP port or to a file. The first parameter, *message*, is the error message that should be logged. The second parameter, *message_type* says where the message should go:

Table 1. error_log() log types

0	*message* is sent to PHP's system logger, using the Operating System's system logging mechanism or a file, depending on what the error_log configuration directive is set to.
1	*message* is sent by email to the address in the *destination* parameter. This is the only message type where the fourth parameter, *extra_headers* is used. This message type uses the same internal function as **mail()** does.
2	*message* is sent through the PHP debugging connection. This option is only available if remote debugging has been enabled. In this case, the *destination* parameter specifies the host name or IP address and optionally, port number, of the socket receiving the debug information.
3	*message* is appended to the file *destination*.

Warning
Remote debugging via TCP/IP is a PHP 3 feature that is *not* available in PHP 4.

Example 1. error_log() examples

```
// Send notification through the server log if we can not
// connect to the database.
if (!Ora_Logon ($username, $password)) {
    error_log ("Oracle database not available!", 0);
}

// Notify administrator by email if we run out of FOO
if (!($foo = allocate_new_foo()) {
    error_log ("Big trouble, we're all out of FOOs!", 1,
               "operator@mydomain.com");
}

// other ways of calling error_log():
error_log ("You messed up!", 2, "127.0.0.1:7000");
error_log ("You messed up!", 2, "loghost");
error_log ("You messed up!", 3, "/var/tmp/my-errors.log");
```

error_reporting

(PHP 3, PHP 4)

error_reporting -- set which PHP errors are reported

Description

int **error_reporting** ([int level])

The **error_reporting()** function sets the error_reporting directive at runtime. PHP has many levels of errors, using this function sets that level for the duration (runtime) of your script.

error_reporting() sets PHP's error reporting level, and returns the old level. The *level* parameter takes on either a bitmask, or named constants. Using named constants is strongly encouraged to ensure compatibility for future versions. As error levels are added, the range of integers increases, so older integer-based error levels will not always behave as expected.

Some example uses:

Example 1. error_reporting() examples

```
<?php

// Turn off all error reporting
error_reporting(0);

// Report simple running errors
error_reporting  (E_ERROR | E_WARNING | E_PARSE);

// Reporting E_NOTICE can be good too (to report uninitialized
// variables or catch variable name misspellings ...)
error_reporting (E_ERROR | E_WARNING | E_PARSE | E_NOTICE);

// Report all errors except E_NOTICE
// This is the default value set in php.ini
error_reporting (E_ALL ^ E_NOTICE);

// Report all PHP errors (bitwise 63 may be used in PHP 3)
error_reporting (E_ALL);

// Same as error_reporting(E_ALL);
ini_set ('error_reporting', E_ALL);

?>
```

The available error level constants are listed below. The actual meanings of these error levels are described in the predefined constants.

Table 1. error_reporting() level constants and bit values

value	constant
1	E_ERROR
2	E_WARNING
4	E_PARSE
8	E_NOTICE
16	E_CORE_ERROR
32	E_CORE_WARNING
64	E_COMPILE_ERROR
128	E_COMPILE_WARNING
256	E_USER_ERROR
512	E_USER_WARNING
1024	E_USER_NOTICE
2047	E_ALL

See also the display_errors directive and **ini_set()**.

restore_error_handler

(PHP 4 >= 4.0.1)

restore_error_handler -- Restores the previous error handler function

Description

void **restore_error_handler** (void)

Used after changing the error handler function using **set_error_handler()**, to revert to the previous error handler (which could be the built-in or a user defined function)

See also error_reporting(), set_error_handler(), trigger_error(), user_error()

set_error_handler

(PHP 4 >= 4.0.1)

set_error_handler -- Sets a user-defined error handler function.

Description

string **set_error_handler** (callback error_handler)

Sets a user function (*error_handler*) to handle errors in a script. Returns the previously defined error handler (if any), or **FALSE** on error. This function can be used for defining your own way of handling errors during runtime, for example in applications in which you need to do cleanup of data/files when a critical error happens, or when you need to trigger an error under certain conditions (using trigger_error())

The user function needs to accept 2 parameters: the error code, and a string describing the error. From PHP 4.0.2, an additional 3 optional parameters are supplied: the filename in which the error occurred, the line number in which the error occurred, and the context in which the error occurred (an array that points to the active symbol table at the point the error occurred).

> **Note:** Instead of a function name, an array containing an object reference and a method name can also be supplied. (Since PHP 4.3.0)

> **Note:** The following error types cannot be handled with a user defined function: **E_ERROR**, **E_PARSE**, **E_CORE_ERROR**, **E_CORE_WARNING**, **E_COMPILE_ERROR** and **E_COMPILE_WARNING**.

The example below shows the handling of internal exceptions by triggering errors and handling them with a user defined function:

Example 1. Error handling with set_error_handler() and trigger_error()

```
<?php

// redefine the user error constants - PHP 4 only
define ("FATAL",E_USER_ERROR);
define ("ERROR",E_USER_WARNING);
define ("WARNING",E_USER_NOTICE);

// set the error reporting level for this script
error_reporting (FATAL | ERROR | WARNING);

// error handler function
function myErrorHandler ($errno, $errstr, $errfile, $errline) {
  switch ($errno) {
  case FATAL:
    echo "<b>FATAL</b> [$errno] $errstr<br>\n";
    echo "  Fatal error in line ".$errline." of file ".$errfile;
    echo ", PHP ".PHP_VERSION." (".PHP_OS.")<br>\n";
    echo "Aborting...<br>\n";
    exit(1);
    break;
  case ERROR:
    echo "<b>ERROR</b> [$errno] $errstr<br>\n";
    break;
  case WARNING:
    echo "<b>WARNING</b> [$errno] $errstr<br>\n";
    break;
    default:
    echo "Unkown error type: [$errno] $errstr<br>\n";
    break;
  }
}

// function to test the error handling
function scale_by_log ($vect, $scale) {
  if ( !is_numeric($scale) || $scale <= 0 )
    trigger_error("log(x) for x <= 0 is undefined, you used: scale = $scale",
      FATAL);
  if (!is_array($vect)) {
    trigger_error("Incorrect input vector, array of values expected", ERROR);
    return null;
  }
  for ($i=0; $i<count($vect); $i++) {
    if (!is_numeric($vect[$i]))
      trigger_error("Value at position $i is not a number, using 0 (zero)",
        WARNING);
    $temp[$i] = log($scale) * $vect[$i];
  }
  return $temp;
}
```

```
// set to the user defined error handler
$old_error_handler = set_error_handler("myErrorHandler");

// trigger some errors, first define a mixed array with a non-numeric item
echo "vector a\n";
$a = array(2,3,"foo",5.5,43.3,21.11);
print_r($a);

// now generate second array, generating a warning
echo "----\nvector b - a warning (b = log(PI) * a)\n";
$b = scale_by_log($a, M_PI);
print_r($b);

// this is trouble, we pass a string instead of an array
echo "----\nvector c - an error\n";
$c = scale_by_log("not array",2.3);
var_dump($c);

// this is a critical error, log of zero or negative number is undefined
echo "----\nvector d - fatal error\n";
$d = scale_by_log($a, -2.5);

?>
```

And when you run this sample script, the output will be

```
vector a
Array
(
    [0] => 2
    [1] => 3
    [2] => foo
    [3] => 5.5
    [4] => 43.3
    [5] => 21.11
)
----
vector b - a warning (b = log(PI) * a)
<b>WARNING</b> [1024] Value at position 2 is not a number, using 0 (zero)<br>
Array
(
    [0] => 2.2894597716988
    [1] => 3.4341896575482
    [2] => 0
    [3] => 6.2960143721717
    [4] => 49.566804057279
    [5] => 24.165247890281
)
----
vector c - an error
<b>ERROR</b> [512] Incorrect input vector, array of values expected<br>
NULL
----
vector d - fatal error
<b>FATAL</b> [256] log(x) for x <= 0 is undefined, you used: scale = -2.5<br>
  Fatal error in line 36 of file trigger_error.php, PHP 4.0.2 (Linux)<br>
Aborting...<br>
```

It is important to remember that the standard PHP error handler is completely bypassed. **error_reporting()** settings will have no effect and your error handler will be called regardless - however you are still able to read the current value of error_reporting and act appropriately. Of particular note is that this value will be o if the statement that caused the error was prepended by the @ error-control operator.

Also note that it is your responsibility to **die()** if necessary. If the error-handler function returns, script execution will continue with the next statement after the one that caused an error.

> **Note:** If errors occur before the script is executed (e.g. on file uploads) the custom error handler cannot be called since it is not registered at that time.

See also **error_reporting()**, **restore_error_handler()**, **trigger_error()**, **user_error()**

trigger_error

(PHP 4 >= 4.0.1)

trigger_error -- Generates a user-level error/warning/notice message

Description

void **trigger_error** (string error_msg [, int error_type])

Used to trigger a user error condition, it can be used by in conjunction with the built-in error handler, or with a user defined function that has been set as the new error handler (**set_error_handler()**). It only works with the E_USER family of constants,

and will default to **E_USER_NOTICE**.

This function is useful when you need to generate a particular response to an exception at runtime. For example:

```
 if (assert ($divisor == 0))
    trigger_error ("Cannot divide by zero", E_USER_ERROR);
```

Note: See **set_error_handler()** for a more extensive example.

Note: *error_msg* is limited to 1024 characters in length. Any additional characters beyond 1024 will be truncated.

See also **error_reporting()**, **set_error_handler()**, **restore_error_handler()**, **user_error()**

user_error

(PHP 4)

user_error -- Generates a user-level error/warning/notice message

Description

void **user_error** (string error_msg [, int error_type])

This is an alias for the function **trigger_error()**.

See also **error_reporting()**, **set_error_handler()**, **restore_error_handler()**, and **trigger_error()**

XXVIII. FrontBase Functions

Introduction

These functions allow you to access FrontBase database servers. More information about FrontBase can be found at http://www.frontbase.com/.

Documentation for FrontBase can be found at http://www.frontbase.com/cgi-bin/WebObjects/FrontBase.woa/wa/productsPage?currentPage=Documentation.

Frontbase support has been added to PHP 4.0.6.

Requirements

You must install the FrontBase database server or at least the fbsql client libraries to use this functions. You can get FrontBase from http://www.frontbase.com/.

Installation

In order to have these functions available, you must compile PHP with fbsql support by using the `--with-fbsql[=DIR]` option. If you use this option without specifying the path to fbsql, PHP will search for the fbsql client libraries in the default installation location for the platform. Users who installed FrontBase in a non standard directory should always specify the path to fbsql: `--with-fbsql=/path/to/fbsql`. This will force PHP to use the client libraries installed by FrontBase, avoiding any conflicts.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. FrontBase configuration options

Name	Default	Changeable
fbsql.allow_persistent	"1"	PHP_INI_SYSTEM
fbsql.generate_warnings	"0"	PHP_INI_SYSTEM
fbsql.autocommit	"1"	PHP_INI_SYSTEM
fbsql.max_persistent	"-1"	PHP_INI_SYSTEM
fbsql.max_links	"128"	PHP_INI_SYSTEM
fbsql.max_connections	"128"	PHP_INI_SYSTEM
fbsql.max_results	"128"	PHP_INI_SYSTEM
fbsql.batchSize	"1000"	PHP_INI_SYSTEM
fbsql.default_host	NULL	PHP_INI_SYSTEM
fbsql.default_user	"_SYSTEM"	PHP_INI_SYSTEM
fbsql.default_password	""	PHP_INI_SYSTEM
fbsql.default_database	""	PHP_INI_SYSTEM
fbsql.default_database_password	""	PHP_INI_SYSTEM

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Resource Types

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

FBSQL_ASSOC (integer)

FBSQL_NUM (integer)

FBSQL_BOTH (integer)

FBSQL_LOCK_DEFERRED (integer)

FBSQL_LOCK_OPTIMISTIC (integer)

FBSQL_LOCK_PESSIMISTIC (integer)

FBSQL_ISO_READ_UNCOMMITTED (integer)

FBSQL_ISO_READ_COMMITTED (integer)

FBSQL_ISO_REPEATABLE_READ (integer)

FBSQL_ISO_SERIALIZABLE (integer)

FBSQL_ISO_VERSIONED (integer)

FBSQL_UNKNOWN (integer)

FBSQL_STOPPED (integer)

FBSQL_STARTING (integer)

FBSQL_RUNNING (integer)

FBSQL_STOPPING (integer)

FBSQL_NOEXEC (integer)

FBSQL_LOB_DIRECT (integer)

FBSQL_LOB_HANDLE (integer)

Table of Contents

fbsql_affected_rows

(PHP 4 >= 4.0.6)

fbsql_affected_rows -- Get number of affected rows in previous FrontBase operation

Description

int **fbsql_affected_rows** ([resource link_identifier])

fbsql_affected_rows() returns the number of rows affected by the last INSERT, UPDATE or DELETE query associated with

link_identifier. If the link identifier isn't specified, the last link opened by **fbsql_connect()** is assumed.

> **Note:** If you are using transactions, you need to call **fbsql_affected_rows()** after your INSERT, UPDATE, or DELETE query, not after the commit.

If the last query was a DELETE query with no WHERE clause, all of the records will have been deleted from the table but this function will return zero.

> **Note:** When using UPDATE, FrontBase will not update columns where the new value is the same as the old value. This creates the possibility that **fbsql_affected_rows()** may not actually equal the number of rows matched, only the number of rows that were literally affected by the query.

If the last query failed, this function will return -1.

See also: **fbsql_num_rows()**.

fbsql_autocommit

(PHP 4 >= 4.0.6)

fbsql_autocommit -- Enable or disable autocommit

Description

bool **fbsql_autocommit** (resource link_identifier [, bool OnOff])

fbsql_autocommit() returns the current autocommit status. if the optional OnOff parameter is given the auto commit status will be changed. With OnOff set to **TRUE** each statement will be committed automatically, if no errors was found. With OnOff set to **FALSE** the user must commit or rollback the transaction using either **fbsql_commit()** or **fbsql_rollback()**.

See also: **fbsql_commit()** and **fbsql_rollback()**

fbsql_change_user

(no version information, might be only in CVS)

fbsql_change_user -- Change logged in user of the active connection

Description

resource **fbsql_change_user** (string user, string password [, string database [, resource link_identifier]])

fbsql_change_user() changes the logged in user of the current active connection, or the connection given by the optional parameter link_identifier. If a database is specified, this will default or current database after the user has been changed. If the new user and password authorization fails, the current connected user stays active.

fbsql_close

(PHP 4 >= 4.0.6)

fbsql_close -- Close FrontBase connection

Description

boolean **fbsql_close** ([resource link_identifier])

Returns: **TRUE** on success, **FALSE** on error.

fbsql_close() closes the connection to the FrontBase server that's associated with the specified link identifier. If *link_identifier* isn't specified, the last opened link is used.

Using **fbsql_close()** isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

Example 1. fbsql_close() example

```
<?php
    $link = fbsql_connect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    print ("Connected successfully");
    fbsql_close ($link);
?>
```

See also: fbsql_connect() and fbsql_pconnect().

fbsql_commit

(PHP 4 >= 4.0.6)

fbsql_commit -- Commits a transaction to the database

Description

bool **fbsql_commit** ([resource link_identifier])

Returns **TRUE** on success or **FALSE** on failure.

fbsql_commit() ends the current transaction by writing all inserts, updates and deletes to the disk and unlocking all row and table locks held by the transaction. This command is only needed if autocommit is set to false.

See also: fbsql_autocommit() and fbsql_rollback()

fbsql_connect

(PHP 4 >= 4.0.6)

fbsql_connect -- Open a connection to a FrontBase Server

Description

resource **fbsql_connect** ([string hostname [, string username [, string password]]])

Returns a positive FrontBase link identifier on success, or an error message on failure.

fbsql_connect() establishes a connection to a FrontBase server. The following defaults are assumed for missing optional parameters: *hostname* = '**NULL**', *username* = '_SYSTEM' and *password* = empty password.

If a second call is made to **fbsql_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling fbsql_close().

Example 1. fbsql_connect() example

```
<?php

    $link = fbsql_connect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    print ("Connected successfully");
    fbsql_close ($link);

?>
```

See also fbsql_pconnect() and fbsql_close().

fbsql_create_blob

(PHP 4 >= 4.2.0)

fbsql_create_blob -- Create a BLOB

Description

string **fbsql_create_blob** (string blob_data [, resource link_identifier])

Returns: A resource handle to the newly created blob.

fbsql_create_blob() creates a blob from blob_data. The returned resource handle can be used with insert and update commands to store the blob in the database.

Example 1. fbsql_create_blob() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    $filename = "blobfile.bin";
    $fp = fopen($filename, "rb");
    $blobdata = fread($fp, filesize($filename));
    fclose($fp);

    $blobHandle = fbsql_create_blob($blobdata, $link);

    $sql = "INSERT INTO BLOB_TABLE (BLOB_COLUMN) VALUES ($blobHandle);";
    $rs = fbsql_query($sql, $link);
?>
```

See also: fbsql_create_clob(), fbsql_read_blob(), fbsql_read_clob(), and fbsql_set_lob_mode().

fbsql_create_clob

(PHP 4 >= 4.2.0)

fbsql_create_clob -- Create a CLOB

Description

string **fbsql_create_clob** (string clob_data [, resource link_identifier])

Returns: A resource handle to the newly created CLOB.

fbsql_create_clob() creates a clob from clob_data. The returned resource handle can be used with insert and update commands to store the clob in the database.

Example 1. fbsql_create_clob() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    $filename = "clob_file.txt";
    $fp = fopen($filename, "rb");
    $clobdata = fread($fp, filesize($filename));
    fclose($fp);

    $clobHandle = fbsql_create_clob($clobdata, $link);

    $sql = "INSERT INTO CLOB_TABLE (CLOB_COLUMN) VALUES ($clobHandle);";
    $rs = fbsql_query($sql, $link);
?>
```

See also: fbsql_create_blob(), fbsql_read_blob(), fbsql_read_clob(), and fbsql_set_lob_mode().

fbsql_create_db

(PHP 4 >= 4.0.6)

fbsql_create_db -- Create a FrontBase database

Description

bool **fbsql_create_db** (string database name [, resource link_identifier])

fbsql_create_db() attempts to create a new database on the server associated with the specified link identifier.

Example 1. fbsql_create_db() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    if (fbsql_create_db ("my_db")) {
        print("Database created successfully\n");
    } else {
        printf("Error creating database: %s\n", fbsql_error ());
    }
?>
```

See also: **fbsql_drop_db()**.

fbsql_data_seek

(PHP 4 >= 4.0.6)

fbsql_data_seek -- Move internal result pointer

Description

bool **fbsql_data_seek** (resource result_identifier, int row_number)

Returns **TRUE** on success or **FALSE** on failure.

fbsql_data_seek() moves the internal row pointer of the FrontBase result associated with the specified result identifier to point to the specified row number. The next call to **fbsql_fetch_row()** would return that row.

Row_number starts at 0.

Example 1. fbsql_data_seek() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");

    fbsql_select_db ("samp_db")
        or die ("Could not select database");

    $query = "SELECT last_name, first_name FROM friends;";
    $result = fbsql_query ($query)
        or die ("Query failed");

    # fetch rows in reverse order

    for ($i = fbsql_num_rows ($result) - 1; $i >=0; $i--) {
        if (!fbsql_data_seek ($result, $i)) {
            printf ("Cannot seek to row %d\n", $i);
            continue;
        }

        if(!($row = fbsql_fetch_object ($result)))
            continue;

        printf("%s %s<BR>\n", $row->last_name, $row->first_name);
    }

    fbsql_free_result ($result);
?>
```

fbsql_database_password

(PHP 4 >= 4.0.6)

fbsql_database_password -- Sets or retrieves the password for a FrontBase database

Description

string **fbsql_database_password** (resource link_identifier [, string database_password])

Returns: The database password associated with the link identifier.

fbsql_database_password() sets and retrieves the database password used by the connection. if a database is protected by a database password, the user must call this function before calling **fbsql_select_db()**. if the second optional parameter is given the function sets the database password for the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if **fbsql_connect()** was called, and use it.

This function does not change the database password in the database nor can it be used to retrive the database password for a database.

Example 1. fbsql_create_clob() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    fbsql_database_password($link, "secret db password");
    fbsql_select_db($database, $link);
?>
```

See also: **fbsql_connect()**, **fbsql_pconnect()** and **fbsql_select_db()**.

fbsql_database

(PHP 4 >= 4.0.6)

fbsql_database -- Get or set the database name used with a connection

Description

string **fbsql_database** (resource link_identifier [, string database])

Warning
This function is currently not documented; only the argument list is available.

fbsql_db_query

(PHP 4 >= 4.0.6)

fbsql_db_query -- Send a FrontBase query

Description

resource **fbsql_db_query** (string database, string query [, resource link_identifier])

Returns: A positive FrontBase result identifier to the query result, or **FALSE** on error.

fbsql_db_query() selects a database and executes a query on it. If the optional link identifier isn't specified, the function will try to find an open link to the FrontBase server and if no such link is found it'll try to create one as if **fbsql_connect()** was called with no arguments

See also **fbsql_connect()**.

fbsql_db_status

(PHP 4 >= 4.1.0)

fbsql_db_status -- Get the status for a given database

Description

int **fbsql_db_status** (string database_name [, resource link_identifier])

Returns: An integer value with the current status.

fbsql_db_status() requests the current status of the database specified by *database_name*. If the *link_identifier* is omitted the default link_identifier will be used.

The return value can be one of the following constants:

- **FALSE** - The exec handler for the host was invalid. This error will occur when the link_identifier connects directly to a database by using a port number. FBExec can be available on the server but no connection has been made for it.
- FBSQL_UNKNOWN - The Status is unknown.
- FBSQL_STOPPED - The database is not running. Use **fbsql_start_db()** to start the database.
- FBSQL_STARTING - The database is starting.
- FBSQL_RUNNING - The database is running and can be used to perform SQL operations.
- FBSQL_STOPPING - The database is stopping.
- FBSQL_NOEXEC - FBExec is not running on the server and it is not possible to get the status of the database.

See also: **fbsql_start_db()** and **fbsql_stop_db()**.

fbsql_drop_db

(PHP 4 >= 4.0.6)

fbsql_drop_db -- Drop (delete) a FrontBase database

Description

bool **fbsql_drop_db** (string database_name [, resource link_identifier])

Returns **TRUE** on success or **FALSE** on failure.

fbsql_drop_db() attempts to drop (remove) an entire database from the server associated with the specified link identifier.

fbsql_errno

(PHP 4 >= 4.0.6)

fbsql_errno -- Returns the numerical value of the error message from previous FrontBase operation

Description

int **fbsql_errno** ([resource link_identifier])

Returns the error number from the last fbsql function, or `0` (zero) if no error occurred.

Errors coming back from the fbsql database backend don't issue warnings. Instead, use **fbsql_errno()** to retrieve the error code. Note that this function only returns the error code from the most recently executed fbsql function (not including **fbsql_error()** and **fbsql_errno()**), so if you want to use it, make sure you check the value before calling another fbsql function.

```
<?php
fbsql_connect("marliesle");
echo fbsql_errno().": ".fbsql_error()."<BR>";
fbsql_select_db("nonexistentdb");
echo fbsql_errno().": ".fbsql_error()."<BR>";
$conn = fbsql_query("SELECT * FROM nonexistenttable;");
echo fbsql_errno().": ".fbsql_error()."<BR>";
?>
```

See also: **fbsql_error()** and **fbsql_warnings()**.

fbsql_error

(PHP 4 >= 4.0.6)

fbsql_error -- Returns the text of the error message from previous FrontBase operation

Description

string **fbsql_error** ([resource link_identifier])

Returns the error text from the last fbsql function, or '' (the empty string) if no error occurred.

Errors coming back from the fbsql database backend don't issue warnings. Instead, use **fbsql_error()** to retrieve the error text. Note that this function only returns the error text from the most recently executed fbsql function (not including **fbsql_error()** and fbsql_errno()), so if you want to use it, make sure you check the value before calling another fbsql function.

```
<?php
fbsql_connect("marliesle");
echo fbsql_errno().": ".fbsql_error()."<BR>";
fbsql_select_db("nonexistentdb");
echo fbsql_errno().": ".fbsql_error()."<BR>";
$conn = fbsql_query("SELECT * FROM nonexistenttable;");
echo fbsql_errno().": ".fbsql_error()."<BR>";
?>
```

See also: fbsql_errno() and fbsql_warnings().

fbsql_fetch_array

(PHP 4 >= 4.0.6)

fbsql_fetch_array -- Fetch a result row as an associative array, a numeric array, or both

Description

array **fbsql_fetch_array** (resource result [, int result_type])

Returns an array that corresponds to the fetched row, or **FALSE** if there are no more rows.

fbsql_fetch_array() is an extended version of fbsql_fetch_row(). In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you must the numeric index of the column or make an alias for the column.

```
select t1.f1 as foo t2.f1 as bar from t1, t2
```

An important thing to note is that using **fbsql_fetch_array()** is NOT significantly slower than using fbsql_fetch_row(), while it provides a significant added value.

The optional second argument *result_type* in **fbsql_fetch_array()** is a constant and can take the following values: FBSQL_ASSOC, FBSQL_NUM, and FBSQL_BOTH.

For further details, see also fbsql_fetch_row() and fbsql_fetch_assoc().

Example 1. fbsql_fetch_array() example

```
<?php
fbsql_connect ($host, $user, $password);
$result = fbsql_db_query ("database","select user_id, fullname from table");
while ($row = fbsql_fetch_array ($result)) {
    echo "user_id: ".$row["user_id"]."<br>\n";
    echo "user_id: ".$row[0]."<br>\n";
    echo "fullname: ".$row["fullname"]."<br>\n";
    echo "fullname: ".$row[1]."<br>\n";
}
fbsql_free_result ($result);
?>
```

fbsql_fetch_assoc

(PHP 4 >= 4.0.6)

fbsql_fetch_assoc -- Fetch a result row as an associative array

Description

array **fbsql_fetch_assoc** (resource result)

Returns an associative array that corresponds to the fetched row, or **FALSE** if there are no more rows.

fbsql_fetch_assoc() is equivalent to calling **fbsql_fetch_array()** with FBSQL_ASSOC for the optional second parameter. It only returns an associative array. This is the way **fbsql_fetch_array()** originally worked. If you need the numeric indices as well as the associative, use **fbsql_fetch_array()**.

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you must use **fbsql_fetch_array()** and have it return the numeric indices as well.

An important thing to note is that using **fbsql_fetch_assoc()** is NOT significantly slower than using **fbsql_fetch_row()**, while it provides a significant added value.

For further details, see also **fbsql_fetch_row()** and **fbsql_fetch_array()**.

Example 1. fbsql_fetch_assoc() example

```
<?php
fbsql_connect ($host, $user, $password);
$result = fbsql_db_query ("database","select * from table");
while ($row = fbsql_fetch_assoc ($result)) {
    echo $row["user_id"];
    echo $row["fullname"];
}
fbsql_free_result ($result);
?>
```

fbsql_fetch_field

(PHP 4 >= 4.0.6)

fbsql_fetch_field -- Get column information from a result and return as an object

Description

object **fbsql_fetch_field** (resource result [, int field_offset])

Returns an object containing field information.

fbsql_fetch_field() can be used in order to obtain information about fields in a certain query result. If the field offset isn't specified, the next field that wasn't yet retrieved by **fbsql_fetch_field()** is retrieved.

The properties of the object are:

- name - column name
- table - name of the table the column belongs to
- max_length - maximum length of the column
- not_null - 1 if the column cannot be **NULL**
- type - the type of the column

Example 1. fbsql_fetch_field() example

```
<?php
fbsql_connect ($host, $user, $password)
    or die ("Could not connect");
$result = fbsql_db_query ("database", "select * from table")
    or die ("Query failed");
# get column metadata
$i = 0;
```

```
while ($i < fbsql_num_fields ($result)) {
    echo "Information for column $i:<BR>\n";
    $meta = fbsql_fetch_field ($result);
    if (!$meta) {
        echo "No information available<BR>\n";
    }
    echo "<PRE>
max_length:   $meta->max_length
name:         $meta->name
not_null:     $meta->not_null
table:        $meta->table
type:         $meta->type
</PRE>";
    $i++;
}
fbsql_free_result ($result);
?>
```

See also **fbsql_field_seek()**.

fbsql_fetch_lengths

(PHP 4 >= 4.0.6)

fbsql_fetch_lengths -- Get the length of each output in a result

Description

array **fbsql_fetch_lengths** ([resource result])

Returns: An array that corresponds to the lengths of each field in the last row fetched by **fbsql_fetch_row()**, or **FALSE** on error.

fbsql_fetch_lengths() stores the lengths of each result column in the last row returned by **fbsql_fetch_row()**, **fbsql_fetch_array()** and **fbsql_fetch_object()** in an array, starting at offset 0.

See also: **fbsql_fetch_row()**.

fbsql_fetch_object

(PHP 4 >= 4.0.6)

fbsql_fetch_object -- Fetch a result row as an object

Description

object **fbsql_fetch_object** (resource result [, int result_type])

Returns an object with properties that correspond to the fetched row, or **FALSE** if there are no more rows.

fbsql_fetch_object() is similar to **fbsql_fetch_array()**, with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

The optional argument *result_type* is a constant and can take the following values: FBSQL_ASSOC, FBSQL_NUM, and FBSQL_BOTH.

Speed-wise, the function is identical to **fbsql_fetch_array()**, and almost as quick as **fbsql_fetch_row()** (the difference is insignificant).

Example 1. fbsql_fetch_object() example

```
<?php
fbsql_connect ($host, $user, $password);
$result = fbsql_db_query ("database", "select * from table");
while ($row = fbsql_fetch_object ($result)) {
    echo $row->user_id;
    echo $row->fullname;
}
fbsql_free_result ($result);
?>
```

See also: **fbsql_fetch_array()** and **fbsql_fetch_row()**.

fbsql_fetch_row

(PHP 4 >= 4.0.6)

fbsql_fetch_row -- Get a result row as an enumerated array

Description

array **fbsql_fetch_row** (resource result)

Returns: An array that corresponds to the fetched row, or **FALSE** if there are no more rows.

fbsql_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to **fbsql_fetch_row()** would return the next row in the result set, or **FALSE** if there are no more rows.

See also: **fbsql_fetch_array()**, **fbsql_fetch_object()**, **fbsql_data_seek()**, **fbsql_fetch_lengths()**, and **fbsql_result()**.

fbsql_field_flags

(PHP 4 >= 4.0.6)

fbsql_field_flags -- Get the flags associated with the specified field in a result

Description

string **fbsql_field_flags** (resource result, int field_offset)

fbsql_field_flags() returns the field flags of the specified field. The flags are reported as a single word per flag separated by a single space, so that you can split the returned value using **explode()**.

fbsql_field_len

(PHP 4 >= 4.0.6)

fbsql_field_len -- Returns the length of the specified field

Description

int **fbsql_field_len** (resource result, int field_offset)

fbsql_field_len() returns the length of the specified field.

fbsql_field_name

(PHP 4 >= 4.0.6)

fbsql_field_name -- Get the name of the specified field in a result

Description

string **fbsql_field_name** (resource result, int field_index)

fbsql_field_name() returns the name of the specified field index. *result* must be a valid result identifier and *field_index* is the numerical offset of the field.

> **Note:** *field_index* starts at 0.
>
> e.g. The index of the third field would actually be 2, the index of the fourth field would be 3 and so on.

Example 1. fbsql_field_name() example

```
// The users table consists of three fields:
//   user_id
//   username
//   password.

$res = fbsql_db_query("users", "select * from users", $link);

echo fbsql_field_name($res, 0) . "\n";
echo fbsql_field_name($res, 2);
```

The above example would produce the following output:

```
user_id
password
```

fbsql_field_seek

(PHP 4 >= 4.0.6)

fbsql_field_seek -- Set result pointer to a specified field offset

Description

bool **fbsql_field_seek** (resource result, int field_offset)

Seeks to the specified field offset. If the next call to **fbsql_fetch_field()** doesn't include a field offset, the field offset specified in **fbsql_field_seek()** will be returned.

See also: **fbsql_fetch_field()**.

fbsql_field_table

(PHP 4 >= 4.0.6)

fbsql_field_table -- Get name of the table the specified field is in

Description

string **fbsql_field_table** (resource result, int field_offset)

Returns the name of the table that the specified field is in.

fbsql_field_type

(PHP 4 >= 4.0.6)

fbsql_field_type -- Get the type of the specified field in a result

Description

string **fbsql_field_type** (resource result, int field_offset)

fbsql_field_type() is similar to the **fbsql_field_name()** function. The arguments are identical, but the field type is returned instead. The field type will be one of "int", "real", "string", "blob", and others as detailed in the FrontBase documentation.

Example 1. fbsql_field_type() example

```
<?php

fbsql_connect ("localhost", "_SYSTEM", "");
fbsql_select_db ("wisconsin");
$result = fbsql_query ("SELECT * FROM onek;");
$fields = fbsql_num_fields ($result);
```

```
$rows   = fbsql_num_rows ($result);
$i = 0;
$table = fbsql_field_table ($result, $i);
echo "Your '".$table."' table has ".$fields." fields and ".$rows." records <BR>";
echo "The table has the following fields <BR>";
while ($i < $fields) {
    $type  = fbsql_field_type  ($result, $i);
    $name  = fbsql_field_name  ($result, $i);
    $len   = fbsql_field_len   ($result, $i);
    $flags = fbsql_field_flags ($result, $i);
    echo $type." ".$name." ".$len." ".$flags."<BR>";
    $i++;
}
fbsql_close();

?>
```

fbsql_free_result

(PHP 4 >= 4.0.6)

fbsql_free_result -- Free result memory

Description

bool **fbsql_free_result** (resource result)

fbsql_free_result() will free all memory associated with the result identifier *result*.

fbsql_free_result() only needs to be called if you are concerned about how much memory is being used for queries that return large result sets. All associated result memory is automatically freed at the end of the script's execution.

fbsql_get_autostart_info

(PHP 4 >= 4.1.0)

fbsql_get_autostart_info -- No description given yet

Description

array **fbsql_get_autostart_info** ([resource link_identifier])

Warning
This function is currently not documented; only the argument list is available.

fbsql_hostname

(PHP 4 >= 4.0.6)

fbsql_hostname -- Get or set the host name used with a connection

Description

string **fbsql_hostname** (resource link_identifier [, string host_name])

Warning
This function is currently not documented; only the argument list is available.

fbsql_insert_id

(PHP 4 >= 4.0.6)

fbsql_insert_id -- Get the id generated from the previous INSERT operation

Description

int **fbsql_insert_id** ([resource link_identifier])

fbsql_insert_id() returns the ID generated for an column defined as DEFAULT UNIQUE by the previous INSERT query using the given *link_identifier*. If *link_identifier* isn't specified, the last opened link is assumed.

fbsql_insert_id() returns o if the previous query does not generate an DEFAULT UNIQUE value. If you need to save the value for later, be sure to call fbsql_insert_id() immediately after the query that generates the value.

> **Note:** The value of the FrontBase SQL function `LAST_INSERT_ID()` always contains the most recently generated DEFAULT UNIQUE value, and is not reset between queries.

fbsql_list_dbs

(PHP 4 >= 4.0.6)

fbsql_list_dbs -- List databases available on a FrontBase server

Description

resource **fbsql_list_dbs** ([resource link_identifier])

fbsql_list_dbs() will return a result pointer containing the databases available from the current fbsql daemon. Use the **fbsql_tablename()** function to traverse this result pointer.

Example 1. fbsql_list_dbs() example

```
$link = fbsql_connect('localhost', 'myname', 'secret');
$db_list = fbsql_list_dbs($link);

while ($row = fbsql_fetch_object($db_list)) {
    echo $row->Database . "\n";
}
```

The above example would produce the following output:

```
database1
database2
database3
...
```

> **Note:** The above code would just as easily work with **fbsql_fetch_row()** or other similar functions.

fbsql_list_fields

(PHP 4 >= 4.0.6)

fbsql_list_fields -- List FrontBase result fields

Description

resource **fbsql_list_fields** (string database_name, string table_name [, resource link_identifier])

fbsql_list_fields() retrieves information about the given tablename. Arguments are the database name and the table name. A result pointer is returned which can be used with **fbsql_field_flags()**, **fbsql_field_len()**, **fbsql_field_name()**, and **fbsql_field_type()**.

A result identifier is a positive integer. The function returns **FALSE** if an error occurs. A string describing the error will be placed in `$phperrmsg`, and unless the function was called as `@fbsql()` then this error string will also be printed out.

Example 1. fbsql_list_fields() example

```
$link = fbsql_connect('localhost', 'myname', 'secret');
```

```
$fields = fbsql_list_fields("database1", "table1", $link);
$columns = fbsql_num_fields($fields);

for ($i = 0; $i < $columns; $i++) {
    echo fbsql_field_name($fields, $i) . "\n";;
}
```

The above example would produce the following output:

```
field1
field2
field3
...
```

fbsql_list_tables

(PHP 4 >= 4.0.6)

fbsql_list_tables -- List tables in a FrontBase database

Description

resource **fbsql_list_tables** (string database [, resource link_identifier])

fbsql_list_tables() takes a database name and returns a result pointer much like the fbsql_db_query() function. The fbsql_tablename() function should be used to extract the actual table names from the result pointer.

fbsql_next_result

(PHP 4 >= 4.0.6)

fbsql_next_result -- Move the internal result pointer to the next result

Description

bool **fbsql_next_result** (resource result_id)

When sending more than one SQL statement to the server or executing a stored procedure with multiple results will cause the server to return multiple result sets. This function will test for additional results available form the server. If an additional result set exists it will free the existing result set and prepare to fetch the words from the new result set. The function will return **TRUE** if an additional result set was available or **FALSE** otherwise.

Example 1. fbsql_next_result() example

```
<?php
    $link = fbsql_connect ("localhost", "_SYSTEM", "secret");
    fbsql_select_db("MyDB", $link);
    $SQL = "Select * from table1; select * from table2;";
    $rs = fbsql_query($SQL, $link);
    do {
        while ($row = fbsql_fetch_row($rs)) {
        }
    } while (fbsql_next_result($rs));
    fbsql_free_result($rs);
    fbsql_close ($link);
?>
```

fbsql_num_fields

(PHP 4 >= 4.0.6)

fbsql_num_fields -- Get number of fields in result

Description

int **fbsql_num_fields** (resource result)

fbsql_num_fields() returns the number of fields in a result set.

See also: fbsql_db_query(), fbsql_query(), fbsql_fetch_field(), and fbsql_num_rows().

fbsql_num_rows

(PHP 4 >= 4.0.6)

fbsql_num_rows -- Get number of rows in result

Description

int **fbsql_num_rows** (resource result)

fbsql_num_rows() returns the number of rows in a result set. This command is only valid for SELECT statements. To retrieve the number of rows returned from a INSERT, UPDATE or DELETE query, use fbsql_affected_rows().

Example 1. fbsql_num_rows() example

```
<?php

$link = fbsql_connect("localhost", "username", "password");
fbsql_select_db("database", $link);

$result = fbsql_query("SELECT * FROM table1;", $link);
$num_rows = fbsql_num_rows($result);

echo "$num_rows Rows\n";

?>
```

See also: fbsql_affected_rows(), fbsql_connect(), fbsql_select_db(), and fbsql_query().

fbsql_password

(PHP 4 >= 4.0.6)

fbsql_password -- Get or set the user password used with a connection

Description

string **fbsql_password** (resource link_identifier [, string password])

Warning
This function is currently not documented; only the argument list is available.

fbsql_pconnect

(PHP 4 >= 4.0.6)

fbsql_pconnect -- Open a persistent connection to a FrontBase Server

Description

resource **fbsql_pconnect** ([string hostname [, string username [, string password]]])

Returns: A positive FrontBase persistent link identifier on success, or **FALSE** on error.

fbsql_pconnect() establishes a connection to a FrontBase server. The following defaults are assumed for missing optional parameters: *host* = 'localhost', *username* = "_SYSTEM" and *password* = empty password.

fbsql_pconnect() acts very much like fbsql_connect() with two major differences.

To set Frontbase server port number, use fbsql_select_db().

First, when connecting, the function would first try to find a (persistent) link that's already open with the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use.

This type of links is therefore called 'persistent'.

fbsql_query

(PHP 4 >= 4.0.6)

fbsql_query -- Send a FrontBase query

Description

resource **fbsql_query** (string query [, resource link_identifier])

fbsql_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If *link_identifier* isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if fbsql_connect() was called with no arguments, and use it.

> **Note:** The query string shall always end with a semicolon.

fbsql_query() returns **TRUE** (non-zero) or **FALSE** to indicate whether or not the query succeeded. A return value of **TRUE** means that the query was legal and could be executed by the server. It does not indicate anything about the number of rows affected or returned. It is perfectly possible for a query to succeed but affect no rows or return no rows.

The following query is syntactically invalid, so **fbsql_query()** fails and returns **FALSE**:

Example 1. fbsql_query() example

```
<?php
$result = fbsql_query ("SELECT * WHERE 1=1")
    or die ("Invalid query");
?>
```

The following query is semantically invalid if `my_col` is not a column in the table `my_tbl`, so **fbsql_query()** fails and returns **FALSE**:

Example 2. fbsql_query() example

```
<?php
$result = fbsql_query ("SELECT my_col FROM my_tbl")
    or die ("Invalid query");
?>
```

fbsql_query() will also fail and return **FALSE** if you don't have permission to access the table(s) referenced by the query.

Assuming the query succeeds, you can call fbsql_num_rows() to find out how many rows were returned for a SELECT statement or fbsql_affected_rows() to find out how many rows were affected by a DELETE, INSERT, REPLACE, or UPDATE statement.

For SELECT statements, **fbsql_query()** returns a new result identifier that you can pass to fbsql_result(). When you are done with the result set, you can free the resources associated with it by calling fbsql_free_result(). Although, the memory will automatically be freed at the end of the script's execution.

See also: fbsql_affected_rows(), fbsql_db_query(), fbsql_free_result(), fbsql_result(), fbsql_select_db(), and fbsql_connect().

fbsql_read_blob

(PHP 4 >= 4.2.0)

fbsql_read_blob -- Read a BLOB from the database

Description

string **fbsql_read_blob** (string blob_handle [, resource link_identifier])

Returns: A string containing the BLOB specified by blob_handle.

fbsql_read_blob() reads BLOB data from the database. If a select statement contains BLOB and/or BLOB columns FrontBase will return the data directly when data is fetched. This default behavior can be changed with fbsql_set_lob_mode() so the fetch functions will return handles to BLOB and CLOB data. If a handle is fetched a user must call **fbsql_read_blob()** to get the actual BLOB data from the database.

Example 1. fbsql_read_blob() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    $sql = "SELECT BLOB_COLUMN FROM BLOB_TABLE;";
    $rs = fbsql_query($sql, $link);
    $row_data = fbsql_fetch_row($rs);
    // $row_data[0] will now contain the blob data for teh first row
    fbsql_free_result($rs);

    $rs = fbsql_query($sql, $link);
    fbsql_set_lob_mode($rs, FBSQL_LOB_HANDLE);
    $row_data = fbsql_fetch_row($rs);
    // $row_data[0] will now contain a handle to the BLOB data in the first row
    $blob_data = fbsql_read_blob($row_data[0]);
    fbsql_free_result($rs);

?>
```

See also: fbsql_create_blob(), **fbsql_read_blob()**, fbsql_read_clob(), and fbsql_set_lob_mode().

fbsql_read_clob

(PHP 4 >= 4.2.0)

fbsql_read_clob -- Read a CLOB from the database

Description

string **fbsql_read_clob** (string clob_handle [, resource link_identifier])

Returns: A string containing the CLOB specified by clob_handle.

fbsql_read_clob() reads CLOB data from the database. If a select statement contains BLOB and/or CLOB columns FrontBase will return the data directly when data is fetched. This default behavior can be changed with fbsql_set_lob_mode() so the fetch functions will return handles to BLOB and CLOB data. If a handle is fetched a user must call **fbsql_read_clob()** to get the actual CLOB data from the database.

Example 1. fbsql_read_clob() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    $sql = "SELECT CLOB_COLUMN FROM CLOB_TABLE;";
    $rs = fbsql_query($sql, $link);
    $row_data = fbsql_fetch_row($rs);
    // $row_data[0] will now contain the clob data for teh first row
    fbsql_free_result($rs);

    $rs = fbsql_query($sql, $link);
    fbsql_set_lob_mode($rs, FBSQL_LOB_HANDLE);
    $row_data = fbsql_fetch_row($rs);
    // $row_data[0] will now contain a handle to the CLOB data in the first row
    $clob_data = fbsql_read_clob($row_data[0]);
    fbsql_free_result($rs);

?>
```

See also: fbsql_create_blob(), fbsql_read_blob(), **fbsql_read_clob()**, and fbsql_set_lob_mode().

fbsql_result

(PHP 4 >= 4.0.6)

fbsql_result -- Get result data

Description

mixed **fbsql_result** (resource result, int row [, mixed field])

fbsql_result() returns the contents of one cell from a FrontBase result set. The field argument can be the field's offset, or the field's name, or the field's table dot field's name (tabledname.fieldname). If the column name has been aliased ('select foo as bar from...'), use the alias instead of the column name.

When working on large result sets, you should consider using one of the functions that fetch an entire row (specified below). As these functions return the contents of multiple cells in one function call, they're MUCH quicker than **fbsql_result()**. Also, note that specifying a numeric offset for the field argument is much quicker than specifying a fieldname or tablename.fieldname argument.

Calls to **fbsql_result()** should not be mixed with calls to other functions that deal with the result set.

Recommended high-performance alternatives: **fbsql_fetch_row()**, **fbsql_fetch_array()**, and **fbsql_fetch_object()**.

fbsql_rollback

(PHP 4 >= 4.0.6)

fbsql_rollback -- Rollback a transaction to the database

Description

bool **fbsql_rollback** ([resource link_identifier])

Returns **TRUE** on success or **FALSE** on failure.

fbsql_rollback() ends the current transaction by rolling back all statements issued since last commit. This command is only needed if autocommit is set to false.

See also: **fbsql_autocommit()** and **fbsql_commit()**

fbsql_select_db

(PHP 4 >= 4.0.6)

fbsql_select_db -- Select a FrontBase database

Description

bool **fbsql_select_db** (string database_name [, resource link_identifier])

Returns: **TRUE** on success, **FALSE** on error.

fbsql_select_db() sets the current active database on the server that's associated with the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if **fbsql_connect()** was called, and use it.

The client contacts FBExec to obtain the port number to use for the connection to the database. If the database name is a number the system will use that as a port number and it will not ask FBExec for the port number. The FrontBase server can be stared as FRontBase -FBExec=No -port=<port number> <database name>.

Every subsequent call to **fbsql_query()** will be made on the active database.

if the database is protected with a database password, the user must call **fbsql_database_password()** before selecting the database.

See also: **fbsql_connect()**, **fbsql_pconnect()**, **fbsql_database_password()** and **fbsql_query()**.

fbsql_set_lob_mode

(PHP 4 >= 4.2.0)

fbsql_set_lob_mode -- Set the LOB retrieve mode for a FrontBase result set

Description

bool **fbsql_set_lob_mode** (resource result, string database_name)

Returns: **TRUE** on success, **FALSE** on error.

fbsql_set_lob_mode() sets the mode for retrieving LOB data from the database. When BLOB and CLOB data is stored in FrontBase it can be stored direct or indirect. Direct stored LOB data will always be fetched no matter the setting of the lob mode. If the LOB data is less than 512 bytes it will always be stored directly.

- FBSQL_LOB_DIRECT - LOB data is retrieved directly. When data is fetched from the database with **fbsql_fetch_row()**, and other fetch functions, all CLOB and BLOB columns will be returned as ordinary columns. This is the default value on a new FrontBase result.
- FBSQL_LOB_HANDLE - LOB data is retrieved as handles to the data. When data is fetched from the database with **fbsql_fetch_row ()**, and other fetch functions, LOB data will be returned as a handle to the data if the data is stored indirect or the data if it is stored direct. If a handle is returned it will be a 27 byte string formatted as "@'oooooooooooooooooooooooo'".

See also: **fbsql_create_blob()**, **fbsql_create_clob()**, **fbsql_read_blob()**, and **fbsql_read_clob()**.

fbsql_set_transaction

(PHP 4 >= 4.2.0)

fbsql_set_transaction -- Set the transaction locking and isolation

Description

void **fbsql_set_transaction** (resource link_identifier, int Locking, int Isolation)

Warning
This function is currently not documented; only the argument list is available.

fbsql_start_db

(PHP 4 >= 4.0.6)

fbsql_start_db -- Start a database on local or remote server

Description

bool **fbsql_start_db** (string database_name [, resource link_identifier])

Returns **TRUE** on success or **FALSE** on failure.

fbsql_start_db()

See also: **fbsql_db_status()** and **fbsql_stop_db()**.

fbsql_stop_db

(PHP 4 >= 4.0.6)

fbsql_stop_db -- Stop a database on local or remote server

Description

bool **fbsql_stop_db** (string database_name [, resource link_identifier])

Returns **TRUE** on success or **FALSE** on failure.

fbsql_stop_db()

See also: **fbsql_db_status()** and **fbsql_start_db()**.

fbsql_tablename

(PHP 4 >= 4.2.0)

fbsql_tablename -- Get table name of field

Description

string **fbsql_tablename** (resource result, int i)

fbsql_tablename() takes a result pointer returned by the **fbsql_list_tables()** function as well as an integer index and returns the name of a table. The **fbsql_num_rows()** function may be used to determine the number of tables in the result pointer.

Example 1. fbsql_tablename() example

```
<?php
fbsql_connect ("localhost", "_SYSTEM", "");
$result = fbsql_list_tables ("wisconsin");
$i = 0;
while ($i < fbsql_num_rows ($result)) {
    $tb_names[$i] = fbsql_tablename ($result, $i);
    echo $tb_names[$i] . "<BR>";
    $i++;
}
?>
```

fbsql_username

(PHP 4 >= 4.0.6)

fbsql_username -- Get or set the host user used with a connection

Description

string **fbsql_username** (resource link_identifier [, string username])

Warning
This function is currently not documented; only the argument list is available.

fbsql_warnings

(PHP 4 >= 4.0.6)

fbsql_warnings -- Enable or disable FrontBase warnings

Description

bool **fbsql_warnings** ([bool OnOff])

Returns **TRUE** if warnings is turned on otherwise **FALSE**.

fbsql_warnings() enables or disables FrontBase warnings.

XXIX. filePro functions

Introduction

These functions allow read-only access to data stored in filePro databases.

filePro is a registered trademark of fP Technologies, Inc. You can find more information about filePro at http://www.fptech.com/.

Installation

filePro support in PHP is not enabled by default. To enable the bundled *read-only* filePro support you need to use the `--enable-filepro` configuration option when compiling PHP.

Table of Contents

filepro_fieldcount

(PHP 3, PHP 4)

filepro_fieldcount -- Find out how many fields are in a filePro database

Description

int **filepro_fieldcount** (void)

Returns the number of fields (columns) in the opened filePro database.

See also **filepro()**.

filepro_fieldname

(PHP 3, PHP 4)

filepro_fieldname -- Gets the name of a field

Description

string **filepro_fieldname** (int field_number)

Returns the name of the field corresponding to *field_number*.

filepro_fieldtype

(PHP 3, PHP 4)

filepro_fieldtype -- Gets the type of a field

Description

string **filepro_fieldtype** (int field_number)

Returns the edit type of the field corresponding to *field_number*.

filepro_fieldwidth

(PHP 3, PHP 4)

filepro_fieldwidth -- Gets the width of a field

Description

int **filepro_fieldwidth** (int field_number)

Returns the width of the field corresponding to *field_number*.

filepro_retrieve

(PHP 3, PHP 4)

filepro_retrieve -- Retrieves data from a filePro database

Description

string **filepro_retrieve** (int row_number, int field_number)

Returns the data from the specified location in the database.

> **Note:** When safe mode is enabled, PHP checks whether the files or directories you are about to operate on have the same UID (owner) as the script that is being executed.

filepro_rowcount

(PHP 3, PHP 4)

filepro_rowcount -- Find out how many rows are in a filePro database

Description

int **filepro_rowcount** (void)

Returns the number of rows in the opened filePro database.

> **Note:** When safe mode is enabled, PHP checks whether the files or directories you are about to operate on have the same UID (owner) as the script that is being executed.

See also **filepro()**.

filepro

(PHP 3, PHP 4)

filepro -- Read and verify the map file

Description

bool **filepro** (string directory)

This reads and verifies the map file, storing the field count and info.

No locking is done, so you should avoid modifying your filePro database while it may be opened in PHP.

> **Note:** When safe mode is enabled, PHP checks whether the files or directories you are about to operate on have the same UID (owner) as the script that is being executed.

XXX. Filesystem functions

Introduction

Requirements

No external libraries are needed to build this extension.

Installation

There is no installation needed to use these functions; they are part of the PHP core.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. Filesystem and Streams Configuration Options

Name	Default	Changeable
allow_url_fopen	"1"	PHP_INI_ALL
user_agent	NULL	PHP_INI_ALL
default_socket_timeout	"60"	PHP_INI_ALL
from	NULL	??
auto_detect_line_endings	"Off"	PHP_INI_ALL

Here is a short explanation of the configuration directives.

`allow_url_fopen` **boolean**

This option enables the URL-aware fopen wrappers that enable accessing URL object like files. Default wrappers are provided for the access of remote files using the ftp or http protocol, some extensions like zlib may register additional wrappers.

Note: This option was introduced immediately after the release of version 4.0.3. For versions up to and including 4.0.3 you can only disable this feature at compile time by using the configuration switch *`--disable-url-fopen-wrapper`*.

Warning
On Windows versions prior to PHP 4.3, the following functions do not support remote file accesing: **include()**, **include_once()**, **require()**, **require_once()** and the imagecreatefromXXX functions in the Reference XLI, *Image functions* extension.

`user_agent` **string**

Define the user agent for PHP to send.

`default_socket_timeout` **integer**

Default timeout (in seconds) for socket based streams.

Note: This configuration option was introduced in PHP 4.3.

`from="joe@example.com"` **string**

Define the anonymous ftp password (your email address).

`auto_detect_line_endings` **boolean**

When turned on, PHP will examine the data read by **fgets()** and **file()** to see if it is using Unix, MS-Dos or Macintosh line-ending conventions.

This enables PHP to interoperate with Macintosh systems, but defaults to Off, as there is a very small performance penalty when detecting the EOL conventions for the first line, and also because people using carriage-returns as item separators under Unix systems would experience non-backwards-compatible behaviour.

Note: This configuration option was introduced in PHP 4.3.

Resource Types

Predefined Constants

This extension has no constants defined.

See Also

For related functions, see also the Directory and Program Execution sections.

For a list and explanation of the various URL wrappers that can be used as remote files, see also Appendix I.

Table of Contents

ftell -- Tells file pointer read/write position
ftruncate -- Truncates a file to a given length
fwrite -- Binary-safe file write
glob -- Find pathnames matching a pattern
is_dir -- Tells whether the filename is a directory
is_executable -- Tells whether the filename is executable
is_file -- Tells whether the filename is a regular file
is_link -- Tells whether the filename is a symbolic link
is_readable -- Tells whether the filename is readable
is_uploaded_file -- Tells whether the file was uploaded via HTTP POST
is_writable -- Tells whether the filename is writable
is_writeable -- Tells whether the filename is writable
link -- Create a hard link
linkinfo -- Gets information about a link
lstat -- Gives information about a file or symbolic link
mkdir -- Makes directory
move_uploaded_file -- Moves an uploaded file to a new location
parse_ini_file -- Parse a configuration file
pathinfo -- Returns information about a file path
pclose -- Closes process file pointer
popen -- Opens process file pointer
readfile -- Outputs a file
readlink -- Returns the target of a symbolic link
realpath -- Returns canonicalized absolute pathname
rename -- Renames a file
rewind -- Rewind the position of a file pointer
rmdir -- Removes directory
set_file_buffer -- Alias of **stream_set_write_buffer()**
stat -- Gives information about a file
symlink -- Creates a symbolic link
tempnam -- Create file with unique file name
tmpfile -- Creates a temporary file
touch -- Sets access and modification time of file
umask -- Changes the current umask
unlink -- Deletes a file

basename

(PHP 3, PHP 4)

basename -- Returns filename component of path

Description

string **basename** (string path [, string suffix])

Given a string containing a path to a file, this function will return the base name of the file. If the filename ends in *suffix* this will also be cut off.

On Windows, both slash (/) and backslash (\) are used as path separator character. In other environments, it is the forward slash (/).

Example 1. basename() example

```
$path = "/home/httpd/html/index.php";
$file = basename ($path);        // $file is set to "index.php"
$file = basename ($path,".php"); // $file is set to "index"
```

Note: The *suffix* parameter was added in PHP 4.1.0.

See also: **dirname()**

chgrp

(PHP 3, PHP 4)

chgrp -- Changes file group

Description

int **chgrp** (string filename, mixed group)

Attempts to change the group of the file *filename* to *group* (specified by name or number). Only the superuser may change the group of a file arbitrarily; other users may change the group of a file to any group of which that user is a member.

Returns **TRUE** on success or **FALSE** on failure.

See also **chown()** and **chmod()**.

chmod

(PHP 3, PHP 4)

chmod -- Changes file mode

Description

int **chmod** (string filename, int mode)

Attempts to change the mode of the file specified by *filename* to that given in *mode*.

Note that *mode* is not automatically assumed to be an octal value, so strings (such as "g+w") will not work properly. To ensure the expected operation, you need to prefix *mode* with a zero (0):

```
chmod ("/somedir/somefile", 755);   // decimal; probably incorrect
chmod ("/somedir/somefile", "u+rwx,go+rx"); // string; incorrect
chmod ("/somedir/somefile", 0755);  // octal; correct value of mode
```

The *mode* parameter consists of three octal number components specifying access restrictions for the owner, the user group in which the owner is in, and to everybody else in this order. One component can be computed by adding up the needed permissions for that target user base. Number 1 means that you grant execute rights, number 2 means that you make the file writeable, number 4 means that you make the file readable. Add up these numbers to specify needed rights. You can also read more about modes on UNIX systems with 'man 1 chmod' and 'man 2 chmod'.

```
// Read and write for owner, nothing for everybody else
chmod ("/somedir/somefile", 0600);

// Read and write for owner, read for everybody else
chmod ("/somedir/somefile", 0644);

// Everything for owner, read and execute for others
chmod ("/somedir/somefile", 0755);

// Everything for owner, read and execute for owner's group
chmod ("/somedir/somefile", 0750);
```

> **Note:** The current user is the user under which PHP runs. It is probably not the same user you use for normal shell or FTP access.

Returns **TRUE** on success or **FALSE** on failure.

See also **chown()** and **chgrp()**.

chown

(PHP 3, PHP 4)

chown -- Changes file owner

Description

int **chown** (string filename, mixed user)

Attempts to change the owner of the file *filename* to user *user* (specified by name or number). Only the superuser may change the owner of a file.

Returns **TRUE** on success or **FALSE** on failure.

See also **chmod()**.

clearstatcache

(PHP 3, PHP 4)

clearstatcache -- Clears file status cache

Description

void **clearstatcache** (void)

When you use **stat()**, **lstat()**, or any of the other functions listed in the affected functions list (below), PHP caches the information those functions return in order to provide faster performance. However, in certain cases, you may want to clear the cached information. For instance, if the same file is being checked multiple times within a single script, and that file is in danger of being removed or changed during that script's operation, you may elect to clear the status cache. In these cases, you can use the **clearstatcache()** function to clear the information that PHP caches about a file.

> **Note:** This function caches information about specific filenames, so you only need to call **clearstatcache()** if you are performing multiple operations on the same filename and require the information about that particular file to not be cached.

Affected functions include **stat()**, **lstat()**, **file_exists()**, **is_writable()**, **is_readable()**, **is_executable()**, **is_file()**, **is_dir()**, **is_link()**, **filectime()**, **fileatime()**, **filemtime()**, **fileinode()**, **filegroup()**, **fileowner()**, **filesize()**, **filetype()**, and **fileperms()**.

copy

(PHP 3, PHP 4)

copy -- Copies file

Description

int **copy** (string source, string dest)

Makes a copy of a file. Returns **TRUE** if the copy succeeded, **FALSE** otherwise.

Example 1. copy() example

```
if (!copy($file, $file.'.bak')) {
    print ("failed to copy $file...<br>\n");
}
```

> **Note:** As of PHP 4.3.0, both *source* and *dest* may be URLs if the "fopen wrappers" have been enabled. See **fopen()** for more details. If *dest* is an URL, the copy operation may fail if the wrapper does not support overwriting of existing files.

Warning
If the destination file already exists, it will be overwritten.

See also **move_uploaded_file()**, **rename()**, and the section of the manual about handling file uploads.

delete

(no version information, might be only in CVS)

delete -- See **unlink()** or **unset()**

Description

void **delete** (string file)

This is a dummy manual entry to satisfy those people who are looking for **unlink()** or **unset()** in the wrong place.

See also: **unlink()** to delete files, **unset()** to delete variables.

dirname

(PHP 3, PHP 4)

dirname -- Returns directory name component of path

Description

string **dirname** (string path)

Given a string containing a path to a file, this function will return the name of the directory.

On Windows, both slash (/) and backslash (\) are used as path separator character. In other environments, it is the forward slash (/).

Example 1. dirname() example

```
$path = "/etc/passwd";
$file = dirname ($path); // $file is set to "/etc"
```

> **Note:** In PHP 4.0.3, **dirname()** was fixed to be POSIX-compliant. Essentially, this means that if there are no slashes in *path* , a dot ('.') is returned, indicating the current directory. Otherwise, the returned string is *path* with any trailing /component removed. Note that this means that you will often get a slash or a dot back from **dirname()** in situations where the older functionality would have given you the empty string.

See also: **basename()**, **pathinfo()**, and **realpath()**.

disk_free_space

(PHP 4 >= 4.1.0)

disk_free_space -- Returns available space in directory

Description

float **disk_free_space** (string directory)

Given a string containing a directory, this function will return the number of bytes available on the corresponding filesystem or disk partition.

Example 1. disk_free_space() example

```
$df = disk_free_space("/"); // $df contains the number of bytes
                            // available on "/"
```

disk_total_space

(PHP 4 >= 4.1.0)

disk_total_space -- Returns the total size of a directory

Description

float **disk_total_space** (string directory)

Given a string containing a directory, this function will return the total number of bytes on the corresponding filesystem or disk partition.

Example 1. disk_total_space() example

```
$df = disk_total_space("/"); // $df contains the total number of
                             // bytes available on "/"
```

diskfreespace

(PHP 3>= 3.0.7, PHP 4)

diskfreespace -- Alias of **disk_free_space()**

Description

float **diskfreespace** (string directory)

This is a deprecated alias of **disk_free_space()**. Use that function instead.

fclose

(PHP 3, PHP 4)

fclose -- Closes an open file pointer

Description

bool **fclose** (resource handle)

The file pointed to by *handle* is closed.

Returns **TRUE** on success or **FALSE** on failure.

The file pointer must be valid, and must point to a file successfully opened by **fopen()** or **fsockopen()**.

Example 1. A simple fclose() example

```
<?php
  $handle = fopen('somefile.txt', 'r');
  fclose($handle);
?>
```

feof

(PHP 3, PHP 4)

feof -- Tests for end-of-file on a file pointer

Description

bool **feof** (resource handle)

Returns **TRUE** if the file pointer is at EOF or an error occurs; otherwise returns **FALSE**.

The file pointer must be valid, and must point to a file successfully opened by **fopen()**, **popen()**, or **fsockopen()**.

fflush

(PHP 4 >= 4.0.1)

fflush -- Flushes the output to a file

Description

bool **fflush** (resource handle)

This function forces a write of all buffered output to the resource pointed to by the file handle *handle*. Returns **TRUE** if successful, **FALSE** otherwise.

The file pointer must be valid, and must point to a file successfully opened by **fopen()**, **popen()**, or **fsockopen()**.

fgetc

(PHP 3, PHP 4)

fgetc -- Gets character from file pointer

Description

string **fgetc** (resource fp)

Returns a string containing a single character read from the file pointed to by *handle*. Returns **FALSE** on EOF.

The file pointer must be valid, and must point to a file successfully opened by **fopen()**, **popen()**, or **fsockopen()**.

> **Note:** This function is binary safe.

See also **fread()**, **fopen()**, **popen()**, **fsockopen()**, and **fgets()**.

fgetcsv

(PHP 3>= 3.0.8, PHP 4)

fgetcsv -- Gets line from file pointer and parse for CSV fields

Description

array **fgetcsv** (resource handle, int length [, string delimiter [, string enclosure]])

Similar to **fgets()** except that **fgetcsv()** parses the line it reads for fields in CSV format and returns an array containing the fields read. The optional third *delimiter* parameter defaults as a comma. The optional *enclosure* cannot be **null**, and is limited to one character. If *enclosure* is more than one character, only the first character is used.

> **Note:** The *enclosure* parameter was added in PHP 4.3.0.

The *handle* parameter must be a valid file pointer to a file successfully opened by **fopen()**, **popen()**, or **fsockopen()**.

The *length* parameter must be greater than the longest line to be found in the CSV file (allowing for trailing line-end characters).

fgetcsv() returns **FALSE** on error, including end of file.

> **Note:** A blank line in a CSV file will be returned as an array comprising a single **null** field, and will not be treated as an error.

Example 1. Read and print the entire contents of a CSV file

```
<?php
$row = 1;
$handle = fopen ("test.csv","r");
```

```
while ($data = fgetcsv ($handle, 1000, ",")) {
    $num = count ($data);
    print "<p> $num fields in line $row: <br>\n";
    $row++;
    for ($c=0; $c < $num; $c++) {
        print $data[$c] . "<br>\n";
    }
}
fclose ($handle);
?>
```

See also **explode()**, **file()**, and **pack()**

fgets

(PHP 3, PHP 4)

fgets -- Gets line from file pointer

Description

string **fgets** (resource handle [, int length])

Returns a string of up to length - 1 bytes read from the file pointed to by *handle*. Reading ends when length - 1 bytes have been read, on a newline (which is included in the return value), or on EOF (whichever comes first). If no length is specified, the length defaults to 1k, or 1024 bytes.

If an error occurs, returns **FALSE**.

Common Pitfalls:

People used to the 'C' semantics of **fgets()** should note the difference in how EOF is returned.

The file pointer must be valid, and must point to a file successfully opened by **fopen()**, **popen()**, or **fsockopen()**.

A simple example follows:

Example 1. Reading a file line by line

```
$handle = fopen ("/tmp/inputfile.txt", "r");
while (!feof ($handle)) {
    $buffer = fgets($handle, 4096);
    echo $buffer;
}
fclose ($handle);
```

> **Note:** The *length* parameter became optional in PHP 4.2.0, if omitted, it would assume 1024 as the line length. As of PHP 4.3, omitting *length* will keep reading from the stream until it reaches the end of the line. If the majority of the lines in the file are all larger than 8KB, it is more resource efficient for your script to specify the maximum line length.
>
> **Note:** This function is binary safe as of PHP 4.3. Earlier versions were not binary safe.
>
> **Note:** If you are having problems with PHP not recognizing the line endings when reading files either on or created by a Macintosh computer, you might want to enable the auto_detect_line_endings run-time configuration option.

See also **fread()**, **fopen()**, **popen()**, **fgetc()**, **fsockopen()**, and **socket_set_timeout()**.

fgetss

(PHP 3, PHP 4)

fgetss -- Gets line from file pointer and strip HTML tags

Description

string **fgetss** (resource handle, int length [, string allowable_tags])

Identical to **fgets()**, except that fgetss attempts to strip any HTML and PHP tags from the text it reads.

You can use the optional third parameter to specify tags which should not be stripped.

> **Note:** *allowable_tags* was added in PHP 3.0.13, PHP 4.0.0.

> **Note:** If you are having problems with PHP not recognizing the line endings when reading files either on or created by a Macintosh computer, you might want to enable the auto_detect_line_endings run-time configuration option.

See also **fgets()**, **fopen()**, **fsockopen()**, **popen()**, and **strip_tags()**.

file_exists

(PHP 3, PHP 4)

file_exists -- Checks whether a file exists

Description

bool **file_exists** (string filename)

Returns **TRUE** if the file specified by *filename* exists; **FALSE** otherwise.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

The results of this function are cached. See **clearstatcache()** for more details.

> **Using Windows shares:** On windows, use `//computername/share/filename` or `\\\\computername\share\filename` to check files on network shares.

Example 1. Testing whether a file exists

```
<?php
$filename = '/path/to/foo.txt';

if (file_exists($filename)) {
    print "The file $filename exists";
} else {
    print "The file $filename does not exist";
}
?>
```

See also **is_readable()**, **is_writable()**, **is_file()** and **file()**.

file_get_contents

(PHP 4 >= 4.3.0)

file_get_contents -- Reads entire file into a string

Description

string **file_get_contents** (string filename [, int use_include_path])

Identical to **file()**, except that **file_get_contents()** returns the file in a string.

> **Note:** This function is binary-safe.

> **Tip:** You can use a URL as a filename with this function if the fopen wrappers have been enabled. See **fopen()** for more details on how to specify the filename and Appendix I for a list of supported URL protocols.

See also: **fgets()**, **file()**, **fread()**, **include()**, and **readfile()**.

file

(PHP 3, PHP 4)

file -- Reads entire file into an array

Description

array **file** (string filename [, int use_include_path])

Identical to **readfile()**, except that **file()** returns the file in an array. Each element of the array corresponds to a line in the file, with the newline still attached. Upon failure, **file()** returns **FALSE**.

> **Note:** Each line in the resulting array will include the line ending, so you still need to use **trim()** if you do not want the line ending present.
>
> **Note:** If you are having problems with PHP not recognizing the line endings when reading files either on or created by a Macintosh computer, you might want to enable the auto_detect_line_endings run-time configuration option.

You can use the optional *use_include_path* parameter and set it to "1", if you want to search for the file in the include_path, too.

```
<?php
// Get a file into an array.  In this example we'll go through HTTP to get
// the HTML source of a URL.
$lines = file ('http://www.example.com/');

// Loop through our array, show html source as html source; and line numbers too.
foreach ($lines as $line_num => $line) {
    echo "Line #<b>{$line_num}</b> : " . htmlspecialchars($line) . "<br>\n";
}

// Another example, let's get a web page into a string.  See also file_get_contents().
$html = implode ('', file ('http://www.example.com/'));
?>
```

> **Note:** As of PHP 4.3.0 you can use **file_get_contents()** to return the contents of a file as a string.
>
> In PHP 4.3.0 **file()** became binary safe.
>
> **Tip:** You can use a URL as a filename with this function if the fopen wrappers have been enabled. See **fopen()** for more details on how to specify the filename and Appendix I for a list of supported URL protocols.

See also **readfile()**, **fopen()**, **fsockopen()**, **popen()**, **file_get_contents()**, and **include()**.

fileatime

(PHP 3, PHP 4)

fileatime -- Gets last access time of file

Description

int **fileatime** (string filename)

Returns the time the file was last accessed, or **FALSE** in case of an error. The time is returned as a Unix timestamp.

The results of this function are cached. See **clearstatcache()** for more details.

Note: The atime of a file is supposed to change whenever the data blocks of a file are being read. This can be costly performancewise when an application regularly accesses a very large number of files or directories. Some Unix filesystems can be mounted with atime updates disabled to increase the performance of such applications; USENET news spools are a common example. On such filesystems this function will be useless.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

filectime

(PHP 3, PHP 4)

filectime -- Gets inode change time of file

Description

int **filectime** (string filename)

Returns the time the file was last changed, or **FALSE** in case of an error. The time is returned as a Unix timestamp.

The results of this function are cached. See **clearstatcache()** for more details.

Note: In most Unix filesystems, a file is considered changed when its inode data is changed; that is, when the permissions, owner, group, or other metadata from the inode is updated. See also **filemtime()** (which is what you want to use when you want to create "Last Modified" footers on web pages) and **fileatime()**.

Note also that in some Unix texts the ctime of a file is referred to as being the creation time of the file. This is wrong. There is no creation time for Unix files in most Unix filesystems.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

filegroup

(PHP 3, PHP 4)

filegroup -- Gets file group

Description

int **filegroup** (string filename)

Returns the group ID of the file, or **FALSE** in case of an error. The group ID is returned in numerical format, use **posix_getgrgid()** to resolve it to a group name.

The results of this function are cached. See **clearstatcache()** for more details.

> **Note:** This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

fileinode

(PHP 3, PHP 4)

fileinode -- Gets file inode

Description

int **fileinode** (string filename)

Returns the inode number of the file, or **FALSE** in case of an error.

The results of this function are cached. See **clearstatcache()** for more details.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

filemtime

(PHP 3, PHP 4)

filemtime -- Gets file modification time

Description

int **filemtime** (string filename)

Returns the time the file was last modified, or **FALSE** in case of an error. The time is returned as a Unix timestamp, which is suitable for the **date()** function.

The results of this function are cached. See **clearstatcache()** for more details.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

Note: This function returns the time when the data blocks of a file were being written to, that is, the time when the content of the file was changed.

Example 1. filemtime() example

```
<?php
// outputs e.g.  somefile.txt was last modified: December 29 2002 22:16:23.

$filename = 'somefile.txt';
if (file_exists($filename)) {
    echo "$filename was last modified: " . date ("F d Y H:i:s.", filemtime($filename));
}
?>
```

See also **filectime()**, **stat()**, **touch()**, and **getlastmod()**.

fileowner

(PHP 3, PHP 4)

fileowner -- Gets file owner

Description

int **fileowner** (string filename)

Returns the user ID of the owner of the file, or **FALSE** in case of an error. The user ID is returned in numerical format, use **posix_getpwuid()** to resolve it to a username.

The results of this function are cached. See **clearstatcache()** for more details.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

fileperms

(PHP 3, PHP 4)

fileperms -- Gets file permissions

Description

int **fileperms** (string filename)

Returns the permissions on the file, or **FALSE** in case of an error.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

The results of this function are cached. See **clearstatcache()** for more details.

filesize

(PHP 3, PHP 4)

filesize -- Gets file size

Description

int **filesize** (string filename)

Returns the size of the file in bytes, or **FALSE** in case of an error.

The results of this function are cached. See **clearstatcache()** for more details.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

filetype

(PHP 3, PHP 4)

filetype -- Gets file type

Description

string **filetype** (string filename)

Returns the type of the file. Possible values are fifo, char, dir, block, link, file, and unknown.

Returns **FALSE** if an error occurs. **filetype()** will also produce an **E_NOTICE** message if the stat call fails or if the file type is unknown.

The results of this function are cached. See **clearstatcache()** for more details.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

See also: **is_dir()**, **is_file()**, **is_link()**, **file_exists()**, and **stat()**.

flock

(PHP 3>= 3.0.7, PHP 4)

flock -- Portable advisory file locking

Description

bool **flock** (resource handle, int operation [, int &wouldblock])

PHP supports a portable way of locking complete files in an advisory way (which means all accessing programs have to use the same way of locking or it will not work).

flock() operates on *handle* which must be an open file pointer. *operation* is one of the following values:

- To acquire a shared lock (reader), set *operation* to**LOCK_SH** (set to 1 prior to PHP 4.0.1).
- To acquire an exclusive lock (writer), set *operation* to **LOCK_EX** (set to 2 prior to PHP 4.0.1).
- To release a lock (shared or exclusive), set *operation* to **LOCK_UN** (set to 3 prior to PHP 4.0.1).
- If you don't want **flock()** to block while locking, add **LOCK_NB** (4 prior to PHP 4.0.1) to *operation*.

flock() allows you to perform a simple reader/writer model which can be used on virtually every platform (including most Unix derivatives and even Windows). The optional third argument is set to **TRUE** if the lock would block (EWOULDBLOCK errno condition)

Returns **TRUE** on success or **FALSE** on failure.

> **Note:** Because **flock()** requires a file pointer, you may have to use a special lock file to protect access to a file that you intend to truncate by opening it in write mode (with a "w" or "w+" argument to **fopen()**).

Warning
flock() will not work on NFS and many other networked file systems. Check your operating system documentation for more details. On some operating systems **flock()** is implemented at the process level. When using a multithreaded server API like ISAPI you may not be able to rely on **flock()** to protect files against other PHP scripts running in parallel threads of the same server instance! **flock()** is not supported on antiquated filesystems like FAT and its derivates and will therefore always return **FALSE** under this environments (this is especially true for Windows 98 users).

fnmatch

(PHP 4 >= 4.3.0)

fnmatch -- Match filename against a pattern

Description

array **fnmatch** (string pattern, string string [, int flags])

fnmatch() checks if the passed *string* would match the given shell wildcard *pattern*.

This is especialy usefull for filenames, but may also be used on regular strings. The average user may be used to shell patterns or at least in their simplest form to `'?'` and `'*'` wildcards so using **fnmatch()** instead of **ereg()** or **preg_match()** for frontend search expression input may be way more convenient for non-programming users.

Example 1. Checking a color name against a shell wildcard pattern.

```
if(fnmatch("*gr[ae]y", $color)) {
  echo "some form of gray ...";
}
```

See also **glob()**, **ereg()**, **preg_match()** and the unix manpage on `fnmatch(3)` for flag names (as long as they are not documented here).

fopen

(PHP 3, PHP 4)

fopen -- Opens file or URL

Description

resource **fopen** (string filename, string mode [, int use_include_path [, resource zcontext]])

fopen() binds a named resource, specified by *filename*, to a stream. If *filename* is of the form "scheme://...", it is assumed to be a URL and PHP will search for a protocol handler (also known as a wrapper) for that scheme. If no wrappers for that protocol are registered, PHP will emit a notice to help you track potential problems in your script and then continue as though *filename* specifies a regular file.

If PHP has decided that *filename* specifies a local file, then it will try to open a stream on that file. The file must be accessible to PHP, so you need to ensure that the file access permissions allow this access. If you have enabled safe_mode, or open_basedir further restrictions may apply.

If PHP has decided that *filename* specifies a registered protocol, and that protocol is registered as a network URL, PHP will check to make sure that allow_url_fopen is enabled. If it is switched off, PHP will emit a warning and the fopen call will fail.

> **Note:** The list of supported protocols can be found in Appendix I.

mode specifies the type of access you require to the stream. It may be any of the following:

- 'r' - Open for reading only; place the file pointer at the beginning of the file.
- 'r+' - Open for reading and writing; place the file pointer at the beginning of the file.
- 'w' - Open for writing only; place the file pointer at the beginning of the file and truncate the file to zero length. If the file does not exist, attempt to create it.
- 'w+' - Open for reading and writing; place the file pointer at the beginning of the file and truncate the file to zero length. If the file does not exist, attempt to create it.
- 'a' - Open for writing only; place the file pointer at the end of the file. If the file does not exist, attempt to create it.
- 'a+' - Open for reading and writing; place the file pointer at the end of the file. If the file does not exist, attempt to create it.

  Note: The *mode* may contain the letter 'b'. This is useful only on systems which differentiate between binary and text

files (i.e. Windows. It's useless on Unix). If not needed, this will be ignored. You are encouraged to include the 'b' flag in order to make your scripts more portable.

The optional third *use_include_path* parameter can be set to '1' or **TRUE** if you want to search for the file in the include_path, too.

The optional fourth *zcontext* is used for specifying tuning parameters and callbacks.

If the open fails, the function returns **FALSE**.

Example 1. fopen() example

```
<?php
$handle = fopen ("/home/rasmus/file.txt", "r");
$handle = fopen ("/home/rasmus/file.gif", "wb");
$handle = fopen ("http://www.example.com/", "r");
$handle = fopen ("ftp://user:password@example.com/somefile.txt", "w");
?>
```

If you are experiencing problems with reading and writing to files and you're using the server module version of PHP, remember to make sure that the files and directories you're using are accessible to the server process.

On the Windows platform, be careful to escape any backslashes used in the path to the file, or use forward slashes.

```
<?php
$handle = fopen ("c:\\data\\info.txt", "r");
?>
```

See also Appendix I, **fclose()**, **fgets()**, **fsockopen()**, **file()**, **file_exists()**, **is_readable()**, **socket_set_timeout()**, and **popen()**.

fpassthru

(PHP 3, PHP 4)

fpassthru -- Output all remaining data on a file pointer

Description

int **fpassthru** (resource handle)

Reads to EOF on the given file pointer from the current position and writes the results to the output buffer.

If an error occurs, **fpassthru()** returns **FALSE**. Otherwise, **fpassthru()** returns the number of characters read from *handle* and passed through to the output.

The file pointer must be valid, and must point to a file successfully opened by **fopen()**, **popen()**, or **fsockopen()**. You may need to call **rewind()** to reset the file pointer to the beginning of the file if you have already written data to the file. The file is closed when **fpassthru()** is done reading it (leaving *handle* useless).

If you just want to dump the contents of a file to the output buffer, without first modifying it or seeking to a particular offset, you may want to use the **readfile()**, which saves you the **fopen()** call.

Note: When using **fpassthru()** on a binary file on Windows systems, you should make sure to open the file in binary mode by appending a `b` to the mode used in the call to **fopen()**.

You are encouraged to use the `b` flag when dealing with binary files, even if your system does not require it, so that your scripts will be more portable.

See also **readfile()**, **fopen()**, **popen()**, and **fsockopen()**

fputs

(PHP 3, PHP 4)

fputs -- Writes to a file pointer

Description

int **fputs** (resource handle, string str [, int length])

fputs() is an alias to **fwrite()**, and is identical in every way. Note that the *length* parameter is optional and if not specified the entire string will be written.

fread

(PHP 3, PHP 4)

fread -- Binary-safe file read

Description

string **fread** (resource handle, int length)

fread() reads up to *length* bytes from the file pointer referenced by *handle*. Reading stops when *length* bytes have been read or EOF (end of file) reached, whichever comes first.

```
<?php
// get contents of a file into a string
$filename = "/usr/local/something.txt";
$handle = fopen ($filename, "r");
$contents = fread ($handle, filesize ($filename));
fclose ($handle);
?>
```

> **Note:** On systems which differentiate between binary and text files (i.e. Windows) the file must be opened with 'b' included in **fopen()** mode parameter.

```
<?php
$filename = "c:\\files\\somepic.gif";
$handle = fopen ($filename, "rb");
$contents = fread ($handle, filesize ($filename));
fclose ($handle);
?>
```

See also **fwrite()**, **fopen()**, **fsockopen()**, **popen()**, **fgets()**, **fgetss()**, **fscanf()**, **file()**, and **fpassthru()**.

fscanf

(PHP 4 >= 4.0.1)

fscanf -- Parses input from a file according to a format

Description

mixed **fscanf** (resource handle, string format [, string var1])

The function **fscanf()** is similar to **sscanf()**, but it takes its input from a file associated with *handle* and interprets the input according to the specified *format*. If only two parameters were passed to this function, the values parsed will be returned as an array. Otherwise, if optional parameters are passed, the function will return the number of assigned values. The optional parameters must be passed by reference.

Any whitespace in the format string matches any whitespace in the input stream. This means that even a tab `\t` in the format string can match a single space character in the input stream.

Example 1. fscanf() Example

```
$handle = fopen ("users.txt","r");
while ($userinfo = fscanf ($handle, "%s\t%s\t%s\n")) {
    list ($name, $profession, $countrycode) = $userinfo;
    //... do something with the values
}
fclose($handle);
```

Example 2. users.txt

```
javier  argonaut        pe
hiroshi sculptor        jp
```

```
robert  slacker us
luigi   florist it
```

Note: Prior to PHP 4.3.0, the maximum number of characters read from the file was 512 (or up to the first \n, whichever came first). As of PHP 4.3.0 arbitrarily long lines will be read and scanned.

See also **fread()**, **fgets()**, **fgetss()**, **sscanf()**, **printf()**, and **sprintf()**.

fseek

(PHP 3, PHP 4)

fseek -- Seeks on a file pointer

Description

int **fseek** (resource handle, int offset [, int whence])

Sets the file position indicator for the file referenced by *handle*.The new position, measured in bytes from the beginning of the file, is obtained by adding *offset* to the position specified by *whence*, whose values are defined as follows:

SEEK_SET - Set position equal to *offset* bytes.
SEEK_CUR - Set position to current location plus *offset*.
SEEK_END - Set position to end-of-file plus *offset*. (To move to a position before the end-of-file, you need to pass a negative value in *offset*.)

If *whence* is not specified, it is assumed to be **SEEK_SET**.

Upon success, returns 0; otherwise, returns -1. Note that seeking past EOF is not considered an error.

May not be used on file pointers returned by **fopen()** if they use the "http://" or "ftp://" formats.

Note: The *whence* argument was added after PHP 4.0.0.

See also **ftell()** and **rewind()**.

fstat

(PHP 4)

fstat -- Gets information about a file using an open file pointer

Description

array **fstat** (resource handle)

Gathers the statistics of the file opened by the file pointer *handle*. This function is similar to the **stat()** function except that it operates on an open file pointer instead of a filename.

Returns an array with the statistics of the file with the following elements:

1. device
2. inode
3. number of links
4. user id of owner
5. group id owner
6. device type if inode device *
7. size in bytes
8. time of last access

9. time of last modification
10. time of last change
11. blocksize for filesystem I/O *
12. number of blocks allocated

* - only valid on systems supporting the st_blksize type--other systems (i.e. Windows) return -1

The results of this function are cached. See **clearstatcache()** for more details.

ftell

(PHP 3, PHP 4)

ftell -- Tells file pointer read/write position

Description

int **ftell** (resource handle)

Returns the position of the file pointer referenced by *handle*; i.e., its offset into the file stream.

If an error occurs, returns **FALSE**.

The file pointer must be valid, and must point to a file successfully opened by **fopen()** or **popen()**.

See also **fopen()**, **popen()**, **fseek()**, and **rewind()**.

ftruncate

(PHP 4)

ftruncate -- Truncates a file to a given length

Description

bool **ftruncate** (resource handle, int size)

Takes the filepointer, *handle*, and truncates the file to length, *size*. Returns **TRUE** on success or **FALSE** on failure.

fwrite

(PHP 3, PHP 4)

fwrite -- Binary-safe file write

Description

int **fwrite** (resource handle, string string [, int length])

fwrite() writes the contents of *string* to the file stream pointed to by *handle*. If the *length* argument is given, writing will stop after *length* bytes have been written or the end of *string* is reached, whichever comes first.

fwrite() returns the number of bytes written, or **FALSE** on error.

Note that if the *length* argument is given, then the magic_quotes_runtime configuration option will be ignored and no slashes will be stripped from *string*.

> **Note:** On systems which differentiate between binary and text files (i.e. Windows) the file must be opened with 'b' included in **fopen()** mode parameter.

Example 1. A simple fwrite example

```
<?php
$filename = 'test.txt';
$somecontent = "Add this to the file\n";

// Let's make sure the file exists and is writable first.
if (is_writable($filename)) {

    // In our example we're opening $filename in append mode.
    // The file pointer is at the bottom of the file hence
    // that's where $somecontent will go when we fwrite() it.
    if (!$handle = fopen($filename, 'a')) {
         print "Cannot open file ($filename)";
         exit;
    }

    // Write $somecontent to our opened file.
    if (!fwrite($handle, $somecontent)) {
        print "Cannot write to file ($filename)";
        exit;
    }

    print "Success, wrote ($somecontent) to file ($filename)";

    fclose($handle);

} else {
    print "The file $filename is not writable";
}
?>
```

See also **fread()**, **fopen()**, **fsockopen()**, **popen()**, and **fputs()**.

glob

(PHP 4 >= 4.3.0)

glob -- Find pathnames matching a pattern

Description

array **glob** (string pattern [, int flags])

The **glob()** function searches for all the pathnames matching *pattern* according to the rules used by the shell. No tilde expansion or parameter substitution is done.

Returns an array containing the matched files/directories or **FALSE** on error.

Valid flags:

- **GLOB_MARK** - Adds a slash to each item returned
- **GLOB_NOSORT** - Return files as they appear in the directory (no sorting)
- **GLOB_NOCHECK** - Return the search pattern if no files matching it were found
- **GLOB_NOESCAPE** - Backslashes do not quote metacharacters
- **GLOB_BRACE** - Expands {a,b,c} to match 'a', 'b', or 'c'
- **GLOB_ONLYDIR** - Return only directory entries which match the pattern

Example 1. Convenient way how glob() can replace opendir() and friends.

```
<?php
foreach (glob("*.txt") as $filename) {
    echo "$filename size " . filesize($filename) . "\n";
}

/* Output will look something like:

funclist.txt size 44686
funcsummary.txt size 267625
quickref.txt size 137820

*/
?>
```

See also **opendir()**, **readdir()** and **closedir()**, **fnmatch()**.

is_dir

(PHP 3, PHP 4)

is_dir -- Tells whether the filename is a directory

Description

bool **is_dir** (string filename)

Returns **TRUE** if the filename exists and is a directory. If *filename* is a relative filename, it will be checked relative to the current working directory.

The results of this function are cached. See **clearstatcache()** for more details.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

See also **chdir()**, dir, **opendir()**, **is_file()** and **is_link()**.

is_executable

(PHP 3, PHP 4)

is_executable -- Tells whether the filename is executable

Description

bool **is_executable** (string filename)

Returns **TRUE** if the filename exists and is executable.

The results of this function are cached. See **clearstatcache()** for more details.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

Note: This function is **not** available on Win32.

See also **is_file()** and **is_link()**.

is_file

(PHP 3, PHP 4)

is_file -- Tells whether the filename is a regular file

Description

bool **is_file** (string filename)

Returns **TRUE** if the filename exists and is a regular file.

The results of this function are cached. See **clearstatcache()** for more details.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

See also **is_dir()** and **is_link()**.

is_link

(PHP 3, PHP 4)

is_link -- Tells whether the filename is a symbolic link

Description

bool **is_link** (string filename)

Returns **TRUE** if the filename exists and is a symbolic link.

The results of this function are cached. See **clearstatcache()** for more details.

See also **is_dir()**, **is_file()**, and **readlink()**.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

is_readable

(PHP 3, PHP 4)

is_readable -- Tells whether the filename is readable

Description

bool **is_readable** (string filename)

Returns **TRUE** if the filename exists and is readable.

Keep in mind that PHP may be accessing the file as the user id that the web server runs as (often 'nobody'). Safe mode limitations are not taken into account.

The results of this function are cached. See **clearstatcache()** for more details.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

See also **is_writable()**.

is_uploaded_file

(PHP 3>= 3.0.17, PHP 4 >= 4.0.3)

is_uploaded_file -- Tells whether the file was uploaded via HTTP POST

Description

bool **is_uploaded_file** (string filename)

Returns **TRUE** if the file named by `filename` was uploaded via HTTP POST. This is useful to help ensure that a malicious user hasn't tried to trick the script into working on files upon which it should not be working--for instance, `/etc/passwd`.

This sort of check is especially important if there is any chance that anything done with uploaded files could reveal their contents to the user, or even to other users on the same system.

is_uploaded_file() is available only in versions of PHP 3 after PHP 3.0.16, and in versions of PHP 4 after 4.0.2. If you are stuck using an earlier version, you can use the following function to help protect yourself:

> **Note:** The following example will *not* work in versions of PHP 4 after 4.0.2. It depends on internal functionality of PHP which changed after that version.

```
<?php
/* Userland test for uploaded file. */
function is_uploaded_file($filename) {
    if (!$tmp_file = get_cfg_var('upload_tmp_dir')) {
        $tmp_file = dirname(tempnam('', ''));
    }
    $tmp_file .= '/' . basename($filename);
    /* User might have trailing slash in php.ini... */
    return (ereg_replace('/+', '/', $tmp_file) == $filename);
}

/* This is how to use it, since you also don't have
 * move_uploaded_file() in these older versions: */
if (is_uploaded_file($HTTP_POST_FILES['userfile'])) {
```

```
    copy($HTTP_POST_FILES['userfile'], "/place/to/put/uploaded/file");
} else {
    echo "Possible file upload attack: filename '$HTTP_POST_FILES[userfile]'.";
}
?>
```

See also **move_uploaded_file()**, and the section Handling file uploads for a simple usage example.

is_writable

(PHP 4)

is_writable -- Tells whether the filename is writable

Description

bool **is_writable** (string filename)

Returns **TRUE** if the filename exists and is writable. The filename argument may be a directory name allowing you to check if a directory is writeable.

Keep in mind that PHP may be accessing the file as the user id that the web server runs as (often 'nobody'). Safe mode limitations are not taken into account.

The results of this function are cached. See **clearstatcache()** for more details.

This function will not work on remote files; the file to be examined must be accessible via the server's filesystem.

See also **is_readable()**.

is_writeable

(PHP 3, PHP 4)

is_writeable -- Tells whether the filename is writable

Description

bool **is_writeable** (string filename)

This is an alias for **is_writable()**

link

(PHP 3, PHP 4)

link -- Create a hard link

Description

int **link** (string target, string link)

link() creates a hard link.

See also the **symlink()** to create soft links, and **readlink()** along with **linkinfo()**.

linkinfo

(PHP 3, PHP 4)

linkinfo -- Gets information about a link

Description

int **linkinfo** (string path)

linkinfo() returns the st_dev field of the UNIX C stat structure returned by the lstat system call. This function is used to verify if a link (pointed to by *path*) really exists (using the same method as the S_ISLNK macro defined in stat.h). Returns 0 or **FALSE** in case of error.

See also **symlink()**, **link()**, and **readlink()**.

lstat

(PHP 3>= 3.0.4, PHP 4)

lstat -- Gives information about a file or symbolic link

Description

array **lstat** (string filename)

Gathers the statistics of the file or symbolic link named by filename. This function is identical to the **stat()** function except that if the *filename* parameter is a symbolic link, the status of the symbolic link is returned, not the status of the file pointed to by the symbolic link.

Returns an array with the statistics of the file with the following elements:

1. device
2. inode
3. inode protection mode
4. number of links
5. user id of owner
6. group id owner
7. device type if inode device *
8. size in bytes
9. time of last access
10. time of last modification
11. time of last change
12. blocksize for filesystem I/O *
13. number of blocks allocated

* - only valid on systems supporting the st_blksize type--other systems (i.e. Windows) return -1.

lstat() cannot be used on remote files.

The results of this function are cached. See **clearstatcache()** for more details.

mkdir

(PHP 3, PHP 4)

mkdir -- Makes directory

Description

int **mkdir** (string pathname [, int mode])

Attempts to create the directory specified by pathname.

Note that you probably want to specify the mode as an octal number, which means it should have a leading zero. The mode is also modified by the current umask, which you can change using **umask()**.

> **Note:** Mode is ignored on Windows, and became optional in PHP 4.2.0.

The mode is 0777 by default, which means the widest possible access. For more information on modes, read the details on the **chmod()** page.

```
mkdir ("/path/to/my/dir", 0700);
```

Returns **TRUE** on success or **FALSE** on failure.

See also **rmdir()**.

move_uploaded_file

(PHP 4 >= 4.0.3)

move_uploaded_file -- Moves an uploaded file to a new location

Description

bool **move_uploaded_file** (string filename, string destination)

This function checks to ensure that the file designated by *filename* is a valid upload file (meaning that it was uploaded via PHP's HTTP POST upload mechanism). If the file is valid, it will be moved to the filename given by *destination*.

If *filename* is not a valid upload file, then no action will occur, and **move_uploaded_file()** will return **FALSE**.

If *filename* is a valid upload file, but cannot be moved for some reason, no action will occur, and **move_uploaded_file()** will return **FALSE**. Additionally, a warning will be issued.

This sort of check is especially important if there is any chance that anything done with uploaded files could reveal their contents to the user, or even to other users on the same system.

> **Note:** When safe mode is enabled, PHP checks whether the files or directories you are about to operate on have the same UID (owner) as the script that is being executed.
>
> **Note: move_uploaded_file()** is not affected by the normal safe mode UID-restrictions. This is not unsafe because **move_uploaded_file()** only operates on files uploaded via PHP.

Warning
If the destination file already exists, it will be overwritten.

See also **is_uploaded_file()**, and the section Handling file uploads for a simple usage example.

parse_ini_file

(PHP 4)

parse_ini_file -- Parse a configuration file

Description

array **parse_ini_file** (string filename [, bool process_sections])

parse_ini_file() loads in the ini file specified in *filename*, and returns the settings in it in an associative array. By setting the last *process_sections* parameter to **TRUE**, you get a multidimensional array, with the section names and settings included. The default for *process_sections* is **FALSE**

> **Note:** This function has nothing to do with the `php.ini` file. It is already processed, the time you run your script. This function can be used to read in your own application's configuration files.

Note: If a value in the ini file contains any non-alphanumeric characters it needs to be enclosed in double-quotes (").

Note: Since PHP 4.2.1 this function is also affected by safe_mode and open_basedir.

The structure of the ini file is similar to that of the `php.ini`'s.

Constants may also be parsed in the ini file so if you define a constant as an ini value before running **parse_ini_file()**, it will be integrated into the results. Only ini values are evaluated. For example:

Example 1. Contents of `sample.ini`

```
; This is a sample configuration file
; Comments start with ';', as in php.ini

[first_section]
one = 1
five = 5
animal = BIRD

[second_section]
path = /usr/local/bin
URL = "http://www.example.com/~username"
```

Example 2. parse_ini_file() example

```
<?php

define ('BIRD', 'Dodo bird');

// Parse without sections
$ini_array = parse_ini_file("sample.ini");
print_r($ini_array);

// Parse with sections
$ini_array = parse_ini_file("sample.ini", TRUE);
print_r($ini_array);

?>
```

Would produce:

```
Array
(
    [one] => 1
    [five] => 5
    [animal] => Dodo bird
    [path] => /usr/local/bin
    [URL] => http://www.example.com/~username
)
Array
(
    [first_section] => Array
        (
            [one] => 1
            [five] => 5
            [animal] = Dodo bird
        )

    [second_section] => Array
        (
            [path] => /usr/local/bin
            [URL] => http://www.example.com/~username
        )

)
```

pathinfo

(PHP 4 >= 4.0.3)

pathinfo -- Returns information about a file path

Description

array **pathinfo** (string path)

pathinfo() returns an associative array containing information about *`path`*. The following array elements are returned: *`dirname`*, *`basename`* and *`extension`*.

Example 1. pathinfo() Example

```
<?php

$path_parts = pathinfo("/www/htdocs/index.html");

echo $path_parts["dirname"] . "\n";
echo $path_parts["basename"] . "\n";
echo $path_parts["extension"] . "\n";

?>
```

Would produce:

```
/www/htdocs
index.html
html
```

> **Note:** For information on retrieving the current path info, read the section on predefined reserved variables.

See also **dirname()**, **basename()**, **parse_url()** and **realpath()**.

pclose

(PHP 3, PHP 4)

pclose -- Closes process file pointer

Description

int **pclose** (resource handle)

Closes a file pointer to a pipe opened by **popen()**.

The file pointer must be valid, and must have been returned by a successful call to **popen()**.

Returns the termination status of the process that was run.

See also **popen()**.

popen

(PHP 3, PHP 4)

popen -- Opens process file pointer

Description

resource **popen** (string command, string mode)

Opens a pipe to a process executed by forking the command given by command.

Returns a file pointer identical to that returned by **fopen()**, except that it is unidirectional (may only be used for reading or writing) and must be closed with **pclose()**. This pointer may be used with **fgets()**, **fgetss()**, and **fputs()**.

If an error occurs, returns **FALSE**.

> **Note:** If you're looking for bi-directional support (two-way), use **proc_open()**.

Example 1. popen() examole

```
<?php
$handle = popen ("/bin/ls", "r");
?>
```

> **Note:** If the command to be executed could not be found, a valid resource is returned. This may seem odd, but makes sense; it allows you to access any error message returned by the shell:

```
<?php
error_reporting(E_ALL);

/* Add redirection so we can get stderr. */
$handle = popen('/path/to/spooge 2>&1', 'r');
echo "'$handle'; " . gettype($handle) . "\n";
$read = fread($handle, 2096);
echo $read;
pclose($handle);
?>
```

See also **pclose()**, **fopen()**, and **proc_open()**.

readfile

(PHP 3, PHP 4)

readfile -- Outputs a file

Description

int **readfile** (string filename [, int use_include_path])

Reads a file and writes it to the output buffer.

Returns the number of bytes read from the file. If an error occurs, **FALSE** is returned and unless the function was called as @readfile, an error message is printed.

> **Tip:** You can use a URL as a filename with this function if the fopen wrappers have been enabled. See **fopen()** for more details on how to specify the filename and Appendix I for a list of supported URL protocols.

You can use the optional second parameter and set it to "1", if you want to search for the file in the include_path, too.

See also **fpassthru()**, **file()**, **fopen()**, **include()**, **require()**, **virtual()** and Appendix I.

readlink

(PHP 3, PHP 4)

readlink -- Returns the target of a symbolic link

Description

string **readlink** (string path)

readlink() does the same as the readlink C function and returns the contents of the symbolic link path or o in case of error.

See also **is_link()**, **symlink()**, and **linkinfo()**.

realpath

(PHP 4)

realpath -- Returns canonicalized absolute pathname

Description

string **realpath** (string path)

realpath() expands all symbolic links and resolves references to '/./', '/../' and extra '/' characters in the input *path* and return the canonicalized absolute pathname. The resulting path will have no symbolic link, '/./' or '/../' components.

realpath() returns **FALSE** on failure, e.g. if the file does not exists.

Example 1. realpath() example

```
$real_path = realpath ("../../index.php");
```

See also: **basename()**, **dirname()**, and **pathinfo()**.

rename

(PHP 3, PHP 4)

rename -- Renames a file

Description

bool **rename** (string oldname, string newname)

Attempts to rename *oldname* to *newname*.

Returns **TRUE** on success or **FALSE** on failure.

Example 1. Example with rename()

```
<?php
  rename("/tmp/tmp_file.txt", "/home/user/login/docs/my_file.txt");
?>
```

rewind

(PHP 3, PHP 4)

rewind -- Rewind the position of a file pointer

Description

int **rewind** (resource handle)

Sets the file position indicator for *handle* to the beginning of the file stream.

If an error occurs, returns 0, otherwise it returns 1.

The file pointer must be valid, and must point to a file successfully opened by **fopen()**.

> **Note:** If you have opened the file in append ("a") mode, any data you write to the file will always be appended, regardless of the file position.

See also **fseek()** and **ftell()**.

rmdir

(PHP 3, PHP 4)

rmdir -- Removes directory

Description

int **rmdir** (string dirname)

Attempts to remove the directory named by *dirname*. The directory must be empty, and the relevant permissions must permit this. Returns **TRUE** on success or **FALSE** on failure.

See also **mkdir()** and **unlink()**.

set_file_buffer

set_file_buffer -- Alias of **stream_set_write_buffer()**

Description

This function is an alias of **stream_set_write_buffer()**.

stat

(PHP 3, PHP 4)

stat -- Gives information about a file

Description

array **stat** (string filename)

Gathers the statistics of the file named by filename.

Returns an array with the statistics of the file with the following elements:

1. device
2. inode
3. inode protection mode
4. number of links
5. user id of owner
6. group id owner
7. device type if inode device *
8. size in bytes
9. time of last access
10. time of last modification
11. time of last change
12. blocksize for filesystem I/O *
13. number of blocks allocated

* - only valid on systems supporting the st_blksize type--other systems (i.e. Windows) return -1.

Returns **FALSE** in case of error.

stat() cannot be used on remote files.

The results of this function are cached. See **clearstatcache()** for more details.

symlink

(PHP 3, PHP 4)

symlink -- Creates a symbolic link

Description

int **symlink** (string target, string link)

symlink() creates a symbolic link from the existing *target* with the specified name *link*.

See also **link()** to create hard links, and **readlink()** along with **linkinfo()**.

tempnam

(PHP 3, PHP 4)

tempnam -- Create file with unique file name

Description

string **tempnam** (string dir, string prefix)

Creates a file with a unique filename in the specified directory. If the directory does not exist, **tempnam()** may generate a file in the system's temporary directory, and return the name of that.

Prior to PHP 4.0.6, the behaviour of the **tempnam()** function was system dependent. On Windows the TMP environment variable will override the *dir* parameter, on Linux the TMPDIR environment variable has precedence, while SVR4 will always use your *dir* parameter if the directory it points to exists. Consult your system documentation on the tempnam(3) function if in doubt.

Returns the new temporary filename, or the **FALSE** string on failure.

Example 1. tempnam() example

```
$tmpfname = tempnam ("/tmp", "FOO");

$handle = fopen($tmpfname, "w");
fwrite($handle, "writing to tempfile");
fclose($handle);

// do here something

unlink($tmpfname);
```

> **Note:** This function's behavior changed in 4.0.3. The temporary file is also created to avoid a race condition where the file might appear in the filesystem between the time the string was generated and before the the script gets around to creating the file. Note, that you need to remove the file in case you need it no more, it is not done automatically.

See also **tmpfile()** and **unlink()**.

tmpfile

(PHP 3>= 3.0.13, PHP 4)

tmpfile -- Creates a temporary file

Description

resource **tmpfile** (void)

Creates a temporary file with an unique name in write mode, returning a file handle similar to the one returned by **fopen()**. The file is automatically removed when closed (using **fclose()**), or when the script ends.

For details, consult your system documentation on the `tmpfile(3)` function, as well as the `stdio.h` header file.

Example 1. tmpfile() example

```
$temp = tmpfile();
fwrite($temp, "writing to tempfile");
fclose($temp); // this removes the file
```

See also **tempnam()**.

touch

(PHP 3, PHP 4)

touch -- Sets access and modification time of file

Description

int **touch** (string filename [, int time [, int atime]])

Attempts to set the access and modification time of the file named by filename to the value given by time. If the option *time* is not given, uses the present time. This is equivalent to what utime (sometimes referred to as utimes) does. If the third option *atime* is present, the access time of the given filename is set to the value of *atime*. Note that the access time is always modified, regardless of the number of parameters.

If the file does not exist, it is created.

Returns **TRUE** on success or **FALSE** on failure.

Example 1. touch() example

```
if (touch ($FileName)) {
    print "$FileName modification time has been
           changed to todays date and time";
} else {
    print "Sorry Could Not change modification time of $FileName";
}
```

umask

(PHP 3, PHP 4)

umask -- Changes the current umask

Description

int **umask** ([int mask])

umask() sets PHP's umask to mask & 0777 and returns the old umask. When PHP is being used as a server module, the umask is restored when each request is finished.

umask() without arguments simply returns the current umask.

unlink

(PHP 3, PHP 4)

unlink -- Deletes a file

Description

int **unlink** (string filename)

Deletes *filename*. Similar to the Unix C unlink() function.

Returns **TRUE** on success or **FALSE** on failure.

See also **rmdir()** for removing directories.

XXXI. Forms Data Format functions

Introduction

Forms Data Format (FDF) is a format for handling forms within PDF documents. You should read the documentation at http://partners.adobe.com/asn/developer/acrosdk/forms.html for more information on what FDF is and how it is used in general.

The general idea of FDF is similar to HTML forms. The difference is basically the format how data is transmitted to the server when the submit button is pressed (this is actually the Form Data Format) and the format of the form itself (which is the Portable Document Format, PDF). Processing the FDF data is one of the features provided by the fdf functions. But there is

more. One may as well take an existing PDF form and populated the input fields with data without modifying the form itself. In such a case one would create a FDF document (**fdf_create()**) set the values of each input field (**fdf_set_value()**) and associate it with a PDF form (**fdf_set_file()**). Finally it has to be sent to the browser with MimeType `application/vnd.fdf`. The Acrobat reader plugin of your browser recognizes the MimeType, reads the associated PDF form and fills in the data from the FDF document.

If you look at an FDF-document with a text editor you will find a catalogue object with the name `FDF`. Such an object may contain a number of entries like `Fields`, `F`, `Status` etc.. The most commonly used entries are `Fields` which points to a list of input fields, and `F` which contains the filename of the PDF-document this data belongs to. Those entries are referred to in the FDF documentation as /F-Key or /Status-Key. Modifying this entries is done by functions like **fdf_set_file()** and **fdf_set_status()**. Fields are modified with **fdf_set_value()**, **fdf_set_opt()** etc..

Requirements

You need the FDF toolkit SDK available from http://partners.adobe.com/asn/developer/acrosdk/forms.html. As of PHP 4.3 you need at least SDK version 5.0. The FDF toolkit library is available in binary form only, platforms supported by Adobe are Win32, Linux, Solaris and AIX.

Installation

You must compile PHP with `--with-fdftk[=DIR]`.

> **Note:** If you run into problems configuring PHP with fdftk support, check whether the header file `fdftk.h` and the library `libfdftk.so` are at the right place. The configure script supports both the directory structure of the FDF SDK distribution and the usual `DIR/include` / `DIR/lib` layout, so you can point it either directly to the unpacked distribution directory or put the header file and the appropriate library for your platform into e.g. `/usr/local/include` and `/usr/local/lib` and configure with `--with-fdftk=/usr/local`.
>
> **Note to Win32 Users:** In order to enable this module on a Windows environment, you must copy *fdftk.dll* from the DLL folder of the PHP/Win32 binary package to the SYSTEM32 folder of your windows machine. (Ex: C:\WINNT\SYSTEM32 or C:\WINDOWS\SYSTEM32)

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

fdf

Most fdf functions require a *`fdf`* resource as their first parameter. A *`fdf`* resource is a handle to an opened fdf file. *`fdf`* resources may be obtained using **fdf_create()**, **fdf_open()** and **fdf_open_string()**.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

`FDFValue` (integer)

`FDFStatus` (integer)

`FDFFile` (integer)

`FDFID` (integer)

FDFFf (integer)

FDFSetFf (integer)

FDFClearFf (integer)

FDFFlags (integer)

FDFSetF (integer)

FDFClrF (integer)

FDFAP (integer)

FDFAS (integer)

FDFAction (integer)

FDFAA (integer)

FDFAPRef (integer)

FDFIF (integer)

FDFEnter (integer)

FDFExit (integer)

FDFDown (integer)

FDFUp (integer)

FDFFormat (integer)

FDFValidate (integer)

FDFKeystroke (integer)

FDFCalculate (integer)

FDFNormalAP (integer)

FDFRolloverAP (integer)

FDFDownAP (integer)

Examples

The following examples shows just the evaluation of form data.

Example 1. Evaluating a FDF document

```
<?php
// Open fdf from input string provided by the extension
// The pdf form contained several input text fields with the names
// volume, date, comment, publisher, preparer, and two checkboxes
// show_publisher and show_preparer.
$fdf = fdf_open_string($HTTP_FDF_DATA);
$volume = fdf_get_value($fdf, "volume");
echo "The volume field has the value '<B>$volume</B>'<BR>";

$date = fdf_get_value($fdf, "date");
echo "The date field has the value '<B>$date</B>'<BR>";

$comment = fdf_get_value($fdf, "comment");
echo "The comment field has the value '<B>$comment</B>'<BR>";

if(fdf_get_value($fdf, "show_publisher") == "On") {
  $publisher = fdf_get_value($fdf, "publisher");
  echo "The publisher field has the value '<B>$publisher</B>'<BR>";
} else
  echo "Publisher shall not be shown.<BR>";

if(fdf_get_value($fdf, "show_preparer") == "On") {
  $preparer = fdf_get_value($fdf, "preparer");
  echo "The preparer field has the value '<B>$preparer</B>'<BR>";
} else
```

```
  echo "Preparer shall not be shown.<BR>";
fdf_close($fdf);
?>
```

Table of Contents

fdf_add_doc_javascript

(PHP 4 >= 4.3.0)

fdf_add_doc_javascript -- Adds javascript code to the FDF document

Description

bool **fdf_add_doc_javascript** (resource fdfdoc, string script_name, string script_code)

Adds a script to the FDF, which Acrobat then adds to the doc-level scripts of a document, once the FDF is imported into it. It is strongly suggested to use '\r' for linebreaks within *script_code*.

Example 1. Adding JavaScript code to a FDF

```
<?php
$fdf = fdf_create();
fdf_add_doc_javascript($fdf, "PlusOne", "function PlusOne(x)\r{\r  return x+1;\r}\r");
fdf_save($fdf);
?>
```

will create a FDF like this:

```
%FDF-1.2
%âãÏÓ
1 0 obj
<<
/FDF << /JavaScript << /Doc [ (PlusOne)(function PlusOne\(x\)\r{\r  return x+1;\r}\r)] >> >>
>>
endobj
trailer
<<
/Root 1 0 R

>>
```

```
%%EOF
```

fdf_add_template

(PHP 3>= 3.0.13, PHP 4)

fdf_add_template -- Adds a template into the FDF document

Description

bool **fdf_add_template** (int fdfdoc, int newpage, string filename, string template, int rename)

Warning
This function is currently not documented; only the argument list is available.

fdf_close

(PHP 3>= 3.0.6, PHP 4)

fdf_close -- Close an FDF document

Description

bool **fdf_close** (resource fdf_document)

The **fdf_close()** function closes the FDF document.

See also fdf_open().

fdf_create

(PHP 3>= 3.0.6, PHP 4)

fdf_create -- Create a new FDF document

Description

resource **fdf_create** (void)

The **fdf_create()** creates a new FDF document. This function is needed if one would like to populate input fields in a PDF document with data.

Example 1. Populating a PDF document

```
<?php
$outfdf = fdf_create();
fdf_set_value($outfdf, "volume", $volume, 0);

fdf_set_file($outfdf, "http:/testfdf/resultlabel.pdf");
fdf_save($outfdf, "outtest.fdf");
fdf_close($outfdf);
Header("Content-type: application/vnd.fdf");
$fp = fopen("outtest.fdf", "r");
fpassthru($fp);
unlink("outtest.fdf");
?>
```

See also fdf_close(), fdf_save(), fdf_open().

fdf_errno

(PHP 4 >= 4.3.0)

fdf_errno -- Return error code for last fdf operation

Description

int **fdf_errno** (void)

fdf_errno() returns the error code set by the last **fdf_...()** function call. This is zero for a successfull operation or a non-zero error code on failure. A textual description may be obtained using the **fdf_error()** function.

See also **fdf_error()**.

fdf_error

(PHP 4 >= 4.3.0)

fdf_error -- Return error description for fdf error code

Description

string **fdf_error** ([int error_code])

fdf_error() returns a textual description for the fdf error code given in *error_code*. The function uses the internal error code set by the last operation if no *error_code* is given, so `fdf_error()` is a convenient shortcut for `fdf_error(fdf_errno())`.

See also **fdf_errno()**.

fdf_get_ap

(PHP 4 >= 4.3.0)

fdf_get_ap -- Get the appearance of a field

Description

bool **fdf_get_ap** (resource fdf_document, string field, int face, string filename)

The **fdf_get_ap()** function gets the appearance of a *field* (i.e. the value of the /AP key) and stores it in a file. The possible values of *face* are **FDFNormalAP**, **FDFRolloverAP** and **FDFDownAP**. The appearance is stored in *filename*.

fdf_get_attachment

(PHP 4 >= 4.3.0)

fdf_get_attachment -- Extracts uploaded file embedded in the FDF

Description

array **fdf_get_attachment** (resource fdf_document, string fieldname, string savepath)

Extracts a file uploaded by means of the "file selection" field *fieldname* and stores it under *savepath*. *savepath* may be the name of a plain file or an existing directory in which the file is to be created under its original name. Any existing file under the same name will be overwritten.

> **Note:** There seems to be no other way to find out the original filename but to store the file using a directory as *savepath* and check for the basename it was stored under.

The returned array contains the following fields:

- *path* - path were the file got stored

size - size of the stored file in bytes

type - mimetype if given in the FDF

Example 1. Storing an uploaded file

```
<?php
  $fdf = fdf_open_string($HTTP_FDF_DATA);
  $data = fdf_get_attachment($fdf, "filename", "/tmpdir");
  echo "The uploaded file is stored in $data[path]";
?>
```

fdf_get_encoding

(PHP 4 >= 4.3.0)

fdf_get_encoding -- Get the value of the /Encoding key

Description

string **fdf_get_encoding** (resource fdf_document)

The **fdf_get_encoding()** returns the value of the /Encoding key. An empty string is returned if the default PDFDocEncoding/Unicode scheme is used.

See also **fdf_set_encoding()**.

fdf_get_file

(PHP 3>= 3.0.6, PHP 4)

fdf_get_file -- Get the value of the /F key

Description

string **fdf_get_file** (resource fdf_document)

The **fdf_set_file()** returns the value of the /F key.

See also **fdf_set_file()**.

fdf_get_status

(PHP 3>= 3.0.6, PHP 4)

fdf_get_status -- Get the value of the /STATUS key

Description

string **fdf_get_status** (resource fdf_document)

The **fdf_get_status()** returns the value of the /STATUS key.

See also **fdf_set_status()**.

fdf_get_value

(PHP 3>= 3.0.6, PHP 4)

fdf_get_value -- Get the value of a field

Description

string **fdf_get_value** (resource fdf_document, string fieldname [, int which])

The **fdf_get_value()** function returns the value for the requested *fieldname*.

Elements of an array field can be retrieved by passing the optional *which*, starting at zero. For non-array fields the optional parameter *which* will be ignored.

> **Note:** Array support and optional *which* parameter were added in PHP 4.3.

See also **fdf_set_value()**.

fdf_get_version

(PHP 4 >= 4.3.0)

fdf_get_version -- Gets version number for FDF api or file

Description

string **fdf_get_version** ([resource fdf_document])

This function will return the fdf version for the given *fdf_document*, or the toolkit api version number if no parameter is given.

For the current FDF toolkit 5.0 the api version number is '5.0' and the document version number is either '1.2', '1.3' or '1.4'.

See also **fdf_set_version()**.

fdf_header

(PHP 4 >= 4.3.0)

fdf_header -- Sets FDF-specific output headers

Description

bool **fdf_header** (void)

This is a convenience function to set appropriate HTTP headers for FDF output. It sets the `Content-type:` to `application/vnd.fdf`.

fdf_next_field_name

(PHP 3>= 3.0.6, PHP 4)

fdf_next_field_name -- Get the next field name

Description

string **fdf_next_field_name** (resource fdf_document [, string fieldname])

The **fdf_next_field_name()** function returns the name of the field after the field in *fieldname* or the field name of the first field if the second parameter is **NULL**.

Example 1. Detecting all fieldnames in a FDF

```
<?php
  $fdf = fdf_open($HTTP_FDF_DATA);
  for($field = fdf_next_field_name($fdf);
     $field != "";
     $field = fdf_next_field_name($fdf, $field)) {
    echo "field: $field\n";
  }
```

```
?>
```

See also **fdf_enum_fields()** and fdf_get_value().

fdf_open_string

(PHP 4 >= 4.3.0)

fdf_open_string -- Read a FDF document from a string

Description

resource **fdf_open_string** (string fdf_data)

The **fdf_open_string()** function reads form data from a string. *fdf_data* must contain the data as returned from a PDF form or created using fdf_create() and fdf_save_string().

You can **fdf_open_string()** together with $HTTP_FDF_DATA to process fdf form input from a remote client.

Example 1. Accessing the form data

```
<?php
$fdf = fdf_open_string($HTTP_FDF_DATA);
...
fdf_close($fdf);
?>
```

See also fdf_open(), fdf_close(), fdf_create() and fdf_save_string().

fdf_open

(PHP 3>= 3.0.6, PHP 4)

fdf_open -- Open a FDF document

Description

resource **fdf_open** (string filename)

The **fdf_open()** function opens a file with form data. This file must contain the data as returned from a PDF form or created using fdf_create() and fdf_save().

You can process the results of a PDF form POST request by writing the data recieved in $HTTP_FDF_DATA to a file and open it using **fdf_open()**. Starting with PHP 4.3 you can also use fdf_open_string() which handles temporary file creation and cleanup for you.

Example 1. Accessing the form data

```
<?php
// Save the FDF data into a temp file
$fdffp = fopen("test.fdf", "w");
fwrite($fdffp, $HTTP_FDF_DATA, strlen($HTTP_FDF_DATA));
fclose($fdffp);

// Open temp file and evaluate data
$fdf = fdf_open("test.fdf");
...
fdf_close($fdf);
?>
```

See also fdf_open_string(), fdf_close(), fdf_create() and fdf_save().

fdf_save_string

(PHP 4 >= 4.3.0)

fdf_save_string -- Returns the FDF document as a string

Description

string **fdf_save_string** (resource fdf_document)

The **fdf_save_string()** function returns the FDF document as a string.

Example 1. Retrieving FDF as a string

```
<?php
$fdf = fdf_create();
fdf_set_value($fdf, "foo", "bar");
$str = fdf_save_string($fdf);
fdf_close($fdf);
echo $str;
?>
```

will output something like

```
%FDF-1.2
%âãÏÓ
1 0 obj
<<
/FDF << /Fields 2 0 R >>
>>
endobj
2 0 obj
[
<< /T (foo)/V (bar)>>
]
endobj
trailer
<<
/Root 1 0 R

>>
%%EOF
```

See also **fdf_save()**, **fdf_open_string()**, **fdf_create()** and **fdf_close()**.

fdf_save

(PHP 3>= 3.0.6, PHP 4)

fdf_save -- Save a FDF document

Description

bool **fdf_save** (resource fdf_document [, string filename])

The **fdf_save()** function saves a FDF document. The resulting FDF will be written to *filename*. Without a *filename* **fdf_save()** will write the FDF to the default PHP output stream.

See also **fdf_save_string()**, **fdf_create()** and **fdf_close()**.

fdf_set_ap

(PHP 3>= 3.0.6, PHP 4)

fdf_set_ap -- Set the appearance of a field

Description

bool **fdf_set_ap** (resource fdf_document, string field_name, int face, string filename, int page_number)

The **fdf_set_ap()** function sets the appearance of a field (i.e. the value of the /AP key). The possible values of *face* are **FDFNormalAP**, **FDFRolloverAP** and **FDFDownAP**.

fdf_set_encoding

(PHP 4 >= 4.1.0)

fdf_set_encoding -- Sets FDF character encoding

Description

bool **fdf_set_encoding** (resource fdf_document, string encoding)

fdf_set_encoding() sets the character encoding in FDF document *`fdf_document`*. *`encoding`* should be the valid encoding name. Currently the following values are supported: `"Shift-JIS"`, `"UHC"`, `"GBK"`,`"BigFive"`. An empty string resets the encoding to the default PDFDocEncoding/Unicode scheme.

fdf_set_file

(PHP 3>= 3.0.6, PHP 4)

fdf_set_file -- Set PDF document to display FDF data in

Description

bool **fdf_set_file** (resource fdf_document, string url [, string target_frame])

The **fdf_set_file()** selects a different PDF document to display the form results in then the form it originated from. The *`url`* needs to be given as an absolute URL.

The frame to display the document in may be selected using the optional parameter *`target_frame`* or the function **fdf_set_target_frame()**.

Example 1. Passing FDF data to a second form

```
<?php
  /* set content type for Adobe FDF */
  fdf_header();

  /* start new fdf */
  $fdf = fdf_create();

  /* set field "foo" to value "bar" */
  $fdf_set_value($fdf, "foo", "bar");

  /* tell client to display FDF data using "fdf_form.pdf" */
  fdf_set_file($fdf, "http://www.example.com/fdf_form.pdf");

  /* output fdf */
  fdf_save();

  /* clean up */
  fdf_close();
?>
```

See also **fdf_get_file()** and **fdf_set_target_frame()**.

fdf_set_flags

(PHP 4 >= 4.0.2)

fdf_set_flags -- Sets a flag of a field

Description

bool **fdf_set_flags** (resource fdf_document, string fieldname, int whichFlags, int newFlags)

The **fdf_set_flags()** sets certain flags of the given field *`fieldname`*.

See also **fdf_set_opt()**.

fdf_set_javascript_action

(PHP 4 >= 4.0.2)

fdf_set_javascript_action -- Sets an javascript action of a field

Description

bool **fdf_set_javascript_action** (resource fdf_document, string fieldname, int trigger, string script)

fdf_set_javascript_action() sets a javascript action for the given field *fieldname*.

See also **fdf_set_submit_form_action()**.

fdf_set_opt

(PHP 4 >= 4.0.2)

fdf_set_opt -- Sets an option of a field

Description

bool **fdf_set_opt** (resource fdf_document, string fieldname, int element, string str1, string str2)

The **fdf_set_opt()** sets options of the given field *fieldname*.

See also **fdf_set_flags()**.

fdf_set_status

(PHP 3>= 3.0.6, PHP 4)

fdf_set_status -- Set the value of the /STATUS key

Description

bool **fdf_set_status** (resource fdf_document, string status)

The **fdf_set_status()** sets the value of the /STATUS key. When a client recieves a FDF with a status set it will present the value in an alert box.

See also **fdf_get_status()**.

fdf_set_submit_form_action

(PHP 4 >= 4.0.2)

fdf_set_submit_form_action -- Sets a submit form action of a field

Description

bool **fdf_set_submit_form_action** (resource fdf_document, string fieldname, int trigger, string script, int flags)

The **fdf_set_submit_form_action()** sets a submit form action for the given field *fieldname*.

See also **fdf_set_javascript_action()**.

fdf_set_target_frame

(PHP 4 >= 4.3.0)

fdf_set_target_frame -- Set target frame for form display

Description

bool **fdf_set_target_frame** (resource fdf_document, string frame_name)

Sets the target frame to display a result PDF defined with **fdf_save_file()** in.

See also **fdf_save_file()**.

fdf_set_value

(PHP 3>= 3.0.6, PHP 4)

fdf_set_value -- Set the value of a field

Description

bool **fdf_set_value** (resource fdf_document, string fieldname, mixed value [, int isName])

The **fdf_set_value()** function sets the *value* for a field named *fieldname*. The *value* will be stored as a string unless it is an array. In this case all array elements will be stored as a value array.

> **Note:** In older versions of the fdf toolkit last parameter determined if the field value was to be converted to a PDF Name (*isName* = 1) or set to a PDF String (*isName* = 0). The value is no longer used in the current toolkit version 5.0. For compatibility reasons it is still supported as an optional parameter beginning with PHP 4.3, but ignored internaly.
>
> Support for *value* arrays was added in PHP 4.3.

See also **fdf_get_value()** and **fdf_remove_item()**.

fdf_set_version

(PHP 4 >= 4.3.0)

fdf_set_version -- Sets version number for a FDF file

Description

string **fdf_set_version** (resource fdf_document, string version)

This function will set the fdf *version* for the given *fdf_document*. Some features supported by this extension are only available in newer fdf versions. For the current FDF toolkit 5.0 *version* may be either '1.2', '1.3' or '1.4'.

See also **fdf_get_version()**.

XXXII. FriBiDi functions

Introduction

FriBiDi is a free implementation of the Unicode Bidirectional Algorithm.

Requirements

You must download and install the FriBiDi package.

Installation

To enable FriBiDi support in PHP you must compile `--with-fribidi[=DIR]` where DIR is the FriBiDi install directory.

Runtime Configuration

Resource Types

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

FRIBIDI_CHARSET_UTF8 (integer)

FRIBIDI_CHARSET_8859_6 (integer)

FRIBIDI_CHARSET_8859_8 (integer)

FRIBIDI_CHARSET_CP1255 (integer)

FRIBIDI_CHARSET_CP1256 (integer)

FRIBIDI_CHARSET_ISIRI_3342 (integer)

Table of Contents

fribidi_log2vis

(PHP 4 >= 4.0.4)

fribidi_log2vis -- Convert a logical string to a visual one

Description

string **fribidi_log2vis** (string str, string direction, int charset)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

XXXIII. FTP functions

Introduction

The functions in this extension implement client access to file servers speaking the File Transfer Protocol (FTP) as defined in http://www.faqs.org/rfcs/rfc959.html.

Requirements

No external libraries are needed to build this extension.

Installation

In order to use FTP functions with your PHP configuration, you should add the `--enable-ftp` option when installing PHP 4 or `--with-ftp` when using PHP 3.

The windows version of PHP has built in support for this extension. You do not need to load any additional extension in order to use these functions.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension uses one resource type, which is the link identifier of the FTP connection, returned by **ftp_connect()**.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

FTP_ASCII (integer)

FTP_TEXT (integer)

FTP_BINARY (integer)

FTP_IMAGE (integer)

FTP_TIMEOUT_SEC (integer)

See **ftp_set_option()** for information.

The following constants were introduced in PHP 4.3.0.

FTP_AUTOSEEK (integer)

See **ftp_set_option()** for information.

FTP_AUTORESUME (integer)

Automatically determine resume position and start position for GET and PUT requests (only works if FTP_AUTOSEEK is enabled)

FTP_FAILED (integer)

Asynchronous transfer has failed

FTP_FINISHED (integer)

Asynchronous transfer has finished

FTP_MOREDATA (integer)

Asynchronous transfer is still active

Examples

Example 1. FTP example

```
<?php
// set up basic connection
$conn_id = ftp_connect($ftp_server);

// login with username and password
$login_result = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

// check connection
if ((!$conn_id) || (!$login_result)) {
        echo "FTP connection has failed!";
        echo "Attempted to connect to $ftp_server for user $ftp_user_name";
        exit;
    } else {
        echo "Connected to $ftp_server, for user $ftp_user_name";
    }

// upload the file
$upload = ftp_put($conn_id, $destination_file, $source_file, FTP_BINARY);

// check upload status
if (!$upload) {
        echo "FTP upload has failed!";
    } else {
        echo "Uploaded $source_file to $ftp_server as $destination_file";
    }

// close the FTP stream
ftp_close($conn_id);
?>
```

Table of Contents

ftp_cdup

(PHP 3>= 3.0.13, PHP 4)

ftp_cdup -- Changes to the parent directory

Description

bool **ftp_cdup** (resource ftp_stream)

Changes to the parent directory.

Returns **TRUE** on success or **FALSE** on failure.

ftp_chdir

(PHP 3>= 3.0.13, PHP 4)

ftp_chdir -- Changes directories on a FTP server

Description

bool **ftp_chdir** (resource ftp_stream, string directory)

Changes to the specified *directory*.

Returns **TRUE** on success or **FALSE** on error.

ftp_close

(PHP 4 >= 4.2.0)

ftp_close -- Closes an FTP connection

Description

void **ftp_close** (resource ftp_stream)

ftp_close() closes *ftp_stream* and releases the **resource**. After calling this function, you can no longer use the FTP connection and must create a new one with **ftp_connect()**.

See also **ftp_connect()**

ftp_connect

(PHP 3>= 3.0.13, PHP 4)

ftp_connect -- Opens an FTP connection

Description

resource **ftp_connect** (string host [, int port [, int timeout]])

Returns a FTP stream on success or **FALSE** on error.

ftp_connect() opens an FTP connection to the specified *host*. The *port* parameter specifies an alternate port to connect to. If it is omitted or set to zero, then the default FTP port, 21, will be used.

The *timeout* parameter specifies the timeout for all subsequent network operations. If omitted, the default value is 90 seconds. The timeout can be changed and queried at any time with **ftp_set_option()** and **ftp_get_option()**.

> **Note:** The *timeout* parameter became available in PHP 4.2.0.

ftp_delete

(PHP 3>= 3.0.13, PHP 4)

ftp_delete -- Deletes a file on the FTP server

Description

bool **ftp_delete** (resource ftp_stream, string path)

ftp_delete() deletes the file specified by *path* from the FTP server.

Returns **TRUE** on success or **FALSE** on failure.

ftp_exec

(PHP 4 >= 4.0.3)

ftp_exec -- Requests execution of a program on the FTP server

Description

bool **ftp_exec** (resource ftp_stream, string command)

Sends a SITE EXEC *command* request to the FTP server. Returns the output of the command if successful; otherwise returns **FALSE**.

ftp_fget

(PHP 3>= 3.0.13, PHP 4)

ftp_fget -- Downloads a file from the FTP server and saves to an open file

Description

bool **ftp_fget** (resource ftp_stream, resource fp, string remote_file, int mode [, int resumepos])

ftp_fget() retrieves *remote_file* from the FTP server, and writes it to the given file pointer, *fp*. The transfer *mode* specified must be either **FTP_ASCII** or **FTP_BINARY**.

> **Note:** The *resumepos* parameter was added in PHP 4.3.0.

Returns **TRUE** on success or **FALSE** on failure.

See also **ftp_get()**.

ftp_fput

(PHP 3>= 3.0.13, PHP 4)

ftp_fput -- Uploads from an open file to the FTP server

Description

bool **ftp_fput** (resource ftp_stream, string remote_file, resource fp, int mode [, int startpos])

ftp_fput() uploads the data from the file pointer *fp* until the end of the file is reached. The results are stored in *remote_file* on the FTP server. The transfer *mode* specified must be either **FTP_ASCII** or **FTP_BINARY**.

> **Note:** The *startpos* parameter was added in PHP 4.3.0.

Returns **TRUE** on success or **FALSE** on failure.

ftp_get_option

(PHP 4 >= 4.2.0)

ftp_get_option -- Retrieves various runtime behaviours of the current FTP stream

Description

mixed **ftp_get_option** (resource ftp_stream, int option)

Note: This function is only available in CVS.

Returns the value on success or **FALSE** if the given *option* is not supported. In the latter case, a warning message is also thrown.

This function returns the value for the requested *option* from the specified *ftp_stream* . Currently, the following options are supported:

Table 1. Supported runtime FTP options

FTP_TIMEOUT_SEC	Returns the current timeout used for network related operations.

Example 1. ftp_get_option() example

```
// Get the timeout of the given FTP stream
$timeout = ftp_get_option($conn_id, FTP_TIMEOUT_SEC);
```

ftp_get

(PHP 3>= 3.0.13, PHP 4)

ftp_get -- Downloads a file from the FTP server

Description

bool **ftp_get** (resource ftp_stream, string local_file, string remote_file, int mode [, int resumepos])

ftp_get() retrieves *remote_file* from the FTP server, and saves it to *local_file* locally. The transfer *mode* specified must be either **FTP_ASCII** or **FTP_BINARY**.

Note: The *resumepos* parameter was added in PHP 4.3.0.

Returns **TRUE** on success or **FALSE** on failure.

See also **ftp_fget()** and **ftp_async_get()**.

ftp_login

(PHP 3>= 3.0.13, PHP 4)

ftp_login -- Logs in to an FTP connection

Description

bool **ftp_login** (resource ftp_stream, string username, string password)

Logs in to the given FTP stream.

Returns **TRUE** on success or **FALSE** on failure.

ftp_mdtm

(PHP 3>= 3.0.13, PHP 4)

ftp_mdtm -- Returns the last modified time of the given file

Description

int **ftp_mdtm** (resource ftp_stream, string remote_file)

ftp_mdtm() checks the last modified time for a file, and returns it as a UNIX timestamp. If an error occurs, or the file does not exist, -1 is returned.

Returns a UNIX timestamp on success, or -1 on error.

> **Note:** Not all servers support this feature!
>
> **Note: ftp_mdtm()** does not work with directories.

ftp_mkdir

(PHP 3>= 3.0.13, PHP 4)

ftp_mkdir -- Creates a directory

Description

string **ftp_mkdir** (resource ftp_stream, string directory)

Creates the specified *directory*.

Returns the newly created directory name on success or **FALSE** on error.

ftp_nb_continue

(PHP 4 >= 4.3.0)

ftp_nb_continue -- Continues retrieving/sending a file (non-blocking)

Description

bool **ftp_nb_continue** (resource ftp_stream)

Continues retrieving/sending a file nbronously

Returns **FTP_FAILED** or **FTP_FINISHED** or **FTP_MOREDATA**.

ftp_nb_fget

(PHP 4 >= 4.3.0)

ftp_nb_fget -- Retrieves a file from the FTP server and writes it to an open file (non-blocking)

Description

bool **ftp_nb_fget** (resource ftp_stream, resource fp, string remote_file, int mode [, int resumepos])

ftp_nb_fget() retrieves *remote_file* from the FTP server, and writes it to the given file pointer, *fp*. The transfer *mode* specified must be either **FTP_ASCII** or **FTP_BINARY**. The difference between this function and the **ftp_fget()** is that this function retrieves the file asynchronously, so your program can perform other operations while the file is being downloaded.

Returns **TRUE** on success or **FALSE** on failure.

See also **ftp_nb_get()**.

ftp_nb_fput

(PHP 4 >= 4.3.0)

ftp_nb_fput -- Stores a file from an open file to the FTP server (non-blocking)

Description

bool **ftp_nb_fput** (resource ftp_stream, string remote_file, resource fp, int mode [, int startpos])

ftp_nb_fput() uploads the data from the file pointer *fp* until it reaches the end of the file. The results are stored in *remote_file* on the FTP server. The transfer *mode* specified must be either **FTP_ASCII** or **FTP_BINARY**. The difference between this function and the **ftp_fput()** is that this function uploads the file asynchronously, so your program can perform other operations while the file is being downloaded.

Returns **TRUE** on success or **FALSE** on failure.

See also **ftp_nb_put()**, **ftp_nb_continue()**, **ftp_put()** and **ftp_fput()**.

ftp_nb_get

(PHP 4 >= 4.3.0)

ftp_nb_get -- Retrieves a file from the FTP server and writes it to a local file (non-blocking)

Description

bool **ftp_nb_get** (resource ftp_stream, string local_file, string remote_file, int mode [, int resumepos])

ftp_nb_get() retrieves *remote_file* from the FTP server, and saves it to *local_file* locally. The transfer *mode* specified must be either **FTP_ASCII** or **FTP_BINARY**. The difference between this function and the **ftp_get()** is that this function retrieves the file asynchronously, so your program can perform other operations while the file is being downloaded.

Returns **TRUE** on success or **FALSE** on failure.

Example 1. ftp_nb_get() example

```
// Initate the download
$ret = ftp_nb_get($my_connection, "test", "README", FTP_BINARY);
while ($ret == FTP_MOREDATA) {

   // Do whatever you want
   echo ".";

   // Continue downloading...
   $ret = ftp_nb_continue ($my_connection);
}
if ($ret != FTP_FINISHED) {
   echo "There was an error downloading the file...";
   exit(1);
}
```

Example 2. Resuming a download with ftp_nb_get()

```
// Initate
$ret = ftp_nb_get ($my_connection, "test", "README", FTP_BINARY,
                   filesize("test"));
// OR: $ret = ftp_nb_get ($my_connection, "test", "README",
//                        FTP_BINARY, FTP_AUTORESUME);
while ($ret == FTP_MOREDATA) {

   // Do whatever you want
   echo ".";

   // Continue downloading...
   $ret = ftp_nb_continue ($my_connection);
}
if ($ret != FTP_FINISHED) {
   echo "There was an error downloading the file...";
   exit(1);
}
```

Example 3. Resuming a download at position 100 to a new file with ftp_nb_get()

```
// Disable Autoseek
ftp_set_option ($my_connection, FTP_AUTOSEEK, FALSE);
```

```
// Initiate
$ret = ftp_nb_get ($my_connection, "newfile", "README", FTP_BINARY, 100);
while ($ret == FTP_MOREDATA) {

   ...

   // Continue downloading...
   $ret = ftp_nb_continue ($my_connection);
}
```

In the example above, `"newfile"` is 100 bytes smaller than `"README"` on the FTP server because we started reading at offset 100. If we have not have disabled **FTP_AUTOSEEK**, the first 100 bytes of newfile will be `'\0'`.

See also ftp_nb_fget(), ftp_nb_continue(), ftp_get() and ftp_fget().

ftp_nb_put

(PHP 4 >= 4.3.0)

ftp_nb_put -- Stores a file on the FTP server (non-blocking)

Description

bool **ftp_nb_put** (resource ftp_stream, string remote_file, string local_file, int mode [, int startpos])

ftp_nb_put() stores *local_file* on the FTP server, as *remote_file*. The transfer *mode* specified must be either **FTP_ASCII** or **FTP_BINARY**. The difference between this function and the ftp_put() is that this function uploads the file asynchronously, so your program can perform other operations while the file is being downloaded.

Returns **TRUE** on success or **FALSE** on failure.

Example 1. ftp_nb_put() example

```
// Initiate the Upload
$ret = ftp_nb_put($my_connection, "test.remote", "test.local", FTP_BINARY);
while ($ret == FTP_MOREDATA) {

   // Do whatever you want
   echo ".";

   // Continue uploading...
   $ret = ftp_nb_continue ($my_connection);
}
if ($ret != FTP_FINISHED) {
   echo "There was an error uploading the file...";
   exit(1);
}
```

Example 2. Resuming an upload with ftp_nb_put()

```
// Initiate
$ret = ftp_nb_put ($my_connection, "test.remote", "test.local",
                   FTP_BINARY, ftp_size("test.remote"));
// OR: $ret = ftp_nb_put ($my_connection, "test.remote", "test.local",
//                        FTP_BINARY, FTP_AUTORESUME);

while ($ret == FTP_MOREDATA) {

   // Do whatever you want
   echo ".";

   // Continue uploading...
   $ret = ftp_nb_continue ($my_connection);
}
if ($ret != FTP_FINISHED) {
   echo "There was an error uploading the file...";
   exit(1);
}
```

See also ftp_nb_fput(), ftp_nb_continue(), ftp_put() and ftp_fput().

ftp_nlist

(PHP 3>= 3.0.13, PHP 4)

ftp_nlist -- Returns a list of files in the given directory

Description

array **ftp_nlist** (resource ftp_stream, string directory)

Returns an array of filenames from the specified directory on success or **FALSE** on error.

ftp_pasv

(PHP 3>= 3.0.13, PHP 4)

ftp_pasv -- Turns passive mode on or off

Description

bool **ftp_pasv** (resource ftp_stream, bool pasv)

ftp_pasv() turns on passive mode if the *pasv* parameter is **TRUE**. It turns off passive mode if *pasv* is **FALSE**. In passive mode, data connections are initiated by the client, rather than by the server.

Returns **TRUE** on success or **FALSE** on failure.

ftp_put

(PHP 3>= 3.0.13, PHP 4)

ftp_put -- Uploads a file to the FTP server

Description

bool **ftp_put** (resource ftp_stream, string remote_file, string local_file, int mode [, int startpos])

ftp_put() stores *local_file* on the FTP server, as *remote_file*. The transfer *mode* specified must be either **FTP_ASCII** or **FTP_BINARY**.

Note: The *startpos* parameter was added in PHP 4.3.0.

Returns **TRUE** on success or **FALSE** on failure.

Example 1. ftp_put() example

```
$upload = ftp_put($conn_id, $destination_file, $source_file, FTP_ASCII);
```

ftp_pwd

(PHP 3>= 3.0.13, PHP 4)

ftp_pwd -- Returns the current directory name

Description

string **ftp_pwd** (resource ftp_stream)

Returns the current directory or **FALSE** on error.

ftp_quit

(PHP 3>= 3.0.13, PHP 4)

ftp_quit -- Closes an FTP connection

Description

void **ftp_quit** (resource ftp_stream)

ftp_quit() is an alias for **ftp_close()**.

ftp_rawlist

(PHP 3>= 3.0.13, PHP 4)

ftp_rawlist -- Returns a detailed list of files in the given directory

Description

array **ftp_rawlist** (resource ftp_stream, string directory)

ftp_rawlist() executes the FTP LIST command, and returns the result as an array. Each array element corresponds to one line of text. The output is not parsed in any way. The system type identifier returned by **ftp_systype()** can be used to determine how the results should be interpreted.

ftp_rename

(PHP 3>= 3.0.13, PHP 4)

ftp_rename -- Renames a file on the FTP server

Description

bool **ftp_rename** (resource ftp_stream, string from, string to)

ftp_rename() renames the file or directory that is currently named *`from`* to the new name *`to`*, using the FTP stream *`ftp_stream`*.

Returns **TRUE** on success or **FALSE** on failure.

ftp_rmdir

(PHP 3>= 3.0.13, PHP 4)

ftp_rmdir -- Removes a directory

Description

bool **ftp_rmdir** (resource ftp_stream, string directory)

Removes the specified *`directory`*.

Returns **TRUE** on success or **FALSE** on failure.

ftp_set_option

(PHP 4 >= 4.2.0)

ftp_set_option -- Set miscellaneous runtime FTP options

Description

bool **ftp_set_option** (resource ftp_stream, int option, mixed value)

Note: This function is only available in CVS.

Returns **TRUE** if the option could be set; **FALSE** if not. A warning message will be thrown if the *option* is not supported or the passed *value* doesn't match the expected value for the given *option*.

This function controls various runtime options for the specified FTP stream. The *value* parameter depends on which *option* parameter is chosen to be altered. Currently, the following options are supported:

Table 1. Supported runtime FTP options

FTP_TIMEOUT_SEC	Changes the timeout in seconds used for all network related functions. *value* must be an integer that is greater than 0. The default timeout is 90 seconds.
FTP_AUTOSEEK	When enabled, GET or PUT requests with a *resumepos* or *startpos* parameter will first seek to the requested position within the file. This is enabled by default.

Example 1. ftp_set_option() example

```
// Set the network timeout to 10 seconds
ftp_set_option($conn_id, FTP_TIMEOUT_SEC, 10);
```

ftp_site

(PHP 3>= 3.0.15, PHP 4)

ftp_site -- Sends a SITE command to the server

Description

bool **ftp_site** (resource ftp_stream, string cmd)

ftp_site() sends the command specified by *cmd* to the FTP server. SITE commands are not standardized, and vary from server to server. They are useful for handling such things as file permissions and group membership.

Returns **TRUE** on success or **FALSE** on failure.

ftp_size

(PHP 3>= 3.0.13, PHP 4)

ftp_size -- Returns the size of the given file

Description

int **ftp_size** (resource ftp_stream, string remote_file)

ftp_size() returns the size of a file in bytes. If an error occurs, of if the file does not exist, -1 is returned. Not all servers support this feature.

Returns the file size on success, or -1 on error.

ftp_ssl_connect

(PHP 4 >= 4.3.0)

ftp_ssl_connect -- Opens an Secure SSL-FTP connection

Description

resource **ftp_ssl_connect** (string host [, int port [, int timeout]])

Returns a SSL-FTP stream on success or **FALSE** on error.

ftp_ssl_connect() opens a SSL-FTP connection to the specified *host*. The *port* parameter specifies an alternate port to connect to. If it's omitted or set to zero then the default FTP port 21 will be used.

The *timeout* parameter specifies the timeout for all subsequent network operations. If omitted, the default value is 90 seconds. The timeout can be changed and queried at any time with **ftp_set_option()** and **ftp_get_option()**.

> **Why this function may not exist: ftp_ssl_connect()** is only available if OpenSSL support is enabled into your version of PHP. If it's undefined and you've compiled FTP support then this is why.

See also **ftp_connect()**

ftp_systype

(PHP 3>= 3.0.13, PHP 4)

ftp_systype -- Returns the system type identifier of the remote FTP server

Description

string **ftp_systype** (resource ftp_stream)

Returns the remote system type, or **FALSE** on error.

XXXIV. Function Handling functions

Introduction

These functions all handle various operations involved in working with functions.

Requirements

No external libraries are needed to build this extension.

Installation

There is no installation needed to use these functions; they are part of the PHP core.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Predefined Constants

This extension has no constants defined.

Table of Contents

get_defined_functions -- Returns an array of all defined functions
register_shutdown_function -- Register a function for execution on shutdown
register_tick_function -- Register a function for execution on each tick
unregister_tick_function -- De-register a function for execution on each tick

call_user_func_array

(PHP 4 >= 4.0.4)

call_user_func_array -- Call a user function given with an array of parameters

Description

mixed **call_user_func_array** (callback function [, array paramarr])

Call a user defined function given by *function*, with the parameters in *paramarr*. For example:

```
function debug($var, $val)
    echo "***DEBUGGING\nVARIABLE: $var\nVALUE:";
    if (is_array($val) || is_object($val) || is_resource($val))
        print_r($val);
    else
        echo "\n$val\n";
    echo "***\n";
}

$c = mysql_connect();
$host = $_SERVER["SERVER_NAME"];

call_user_func_array ('debug', array("host", $host));
call_user_func_array ('debug', array("c", $c));
call_user_func_array ('debug', array("_POST", $_POST));
```

See also: **call_user_func()**, **call_user_method()**, **call_user_method_array()**.

call_user_func

(PHP 3>= 3.0.3, PHP 4)

call_user_func -- Call a user function given by the first parameter

Description

mixed **call_user_func** (callback function [, mixed parameter [, mixed ...]])

Call a user defined function given by the *function* parameter. Take the following:

```
function barber ($type) {
    print "You wanted a $type haircut, no problem";
}
call_user_func ('barber', "mushroom");
call_user_func ('barber', "shave");
```

See also: **call_user_func_array()**, **call_user_method()**, **call_user_method_array()**.

create_function

(PHP 4 >= 4.0.1)

create_function -- Create an anonymous (lambda-style) function

Description

string **create_function** (string args, string code)

Creates an anonymous function from the parameters passed, and returns a unique name for it. Usually the *args* will be passed as a single quote delimited string, and this is also recommended for the *code*. The reason for using single quoted strings, is to

protect the variable names from parsing, otherwise, if you use double quotes there will be a need to escape the variable names, e.g. `\$avar`.

You can use this function, to (for example) create a function from information gathered at run time:

Example 1. Creating an anonymous function with create_function()

```
$newfunc = create_function('$a,$b','return "ln($a) + ln($b) = ".log($a * $b);');
echo "New anonymous function: $newfunc\n";
echo $newfunc(2,M_E)."\n";
// outputs
// New anonymous function: lambda_1
// ln(2) + ln(2.718281828459) = 1.6931471805599
```

Or, perhaps to have general handler function that can apply a set of operations to a list of parameters:

Example 2. Making a general processing function with create_function()

```
function process($var1, $var2, $farr) {
    for ($f=0; $f < count($farr); $f++)
    echo $farr[$f]($var1,$var2)."\n";
}

// create a bunch of math functions
$f1 = 'if ($a >=0) {return "b*a^2 = ".$b*sqrt($a);} else {return false;}';
$f2 = "return \"min(b^2+a, a^2,b) = \".min(\$a*\$a+\$b,\$b*\$b+\$a);";
$f3 = 'if ($a > 0 && $b != 0) {return "ln(a)/b = ".log($a)/$b; } else { return false; }';
$farr = array(
    create_function('$x,$y', 'return "some trig: ".(sin($x) + $x*cos($y));'),
    create_function('$x,$y', 'return "a hypotenuse: ".sqrt($x*$x + $y*$y);'),
    create_function('$a,$b', $f1),
    create_function('$a,$b', $f2),
    create_function('$a,$b', $f3)
    );

echo "\nUsing the first array of anonymous functions\n";
echo "parameters: 2.3445, M_PI\n";
process(2.3445, M_PI, $farr);

// now make a bunch of string processing functions
$garr = array(
    create_function('$b,$a','if (strncmp($a,$b,3) == 0) return "** \"$a\" '.
    'and \"$b\"\n** Look the same to me! (looking at the first 3 chars)";'),
    create_function('$a,$b',"; return "CRCs: ".crc32($a)." , ".crc32(b);'),
    create_function('$a,$b',"; return "similar(a,b) = ".similar_text($a,$b,&$p)."($p%)";')
    );
echo "\nUsing the second array of anonymous functions\n";
process("Twas brilling and the slithy toves", "Twas the night", $garr);
```

and when you run the code above, the output will be:

```
Using the first array of anonymous functions
parameters: 2.3445, M_PI
some trig: -1.6291725057799
a hypotenuse: 3.9199852871011
b*a^2 = 4.8103313314525
min(b^2+a, a^2,b) = 8.6382729035898
ln(a/b) = 0.27122299212594

Using the second array of anonymous functions
** "Twas the night" and "Twas brilling and the slithy toves"
** Look the same to me! (looking at the first 3 chars)
CRCs: -725381282 , 1908338681
similar(a,b) = 11(45.833333333333%)
```

But perhaps the most common use for of lambda-style (anonymous) functions is to create callback functions, for example when using **array_walk()** or **usort()**

Example 3. Using anonymous functions as callback functions

```
$av = array("the ","a ","that ","this ");
array_walk($av, create_function('&$v,$k','$v = $v."mango";'));
print_r($av);  // for PHP 3 use var_dump()
// outputs:
// Array
// (
//    [0] => the mango
//    [1] => a mango
//    [2] => that mango
//    [3] => this mango
// )

// an array of strings ordered from shorter to longer
$sv = array("small","larger","a big string","it is a string thing");
print_r($sv);
// outputs:
// Array
// (
//    [0] => small
//    [1] => larger
```

```
//   [2] => a big string
//   [3] => it is a string thing
// )

// sort it from longer to shorter
usort($sv, create_function('$a,$b','return strlen($b) - strlen($a);'));
print_r($sv);
// outputs:
// Array
// (
//   [0] => it is a string thing
//   [1] => a big string
//   [2] => larger
//   [3] => small
// )
```

func_get_arg

(PHP 4)

func_get_arg -- Return an item from the argument list

Description

mixed **func_get_arg** (int arg_num)

Returns the argument which is at the *arg_num*'th offset into a user-defined function's argument list. Function arguments are counted starting from zero. **func_get_arg()** will generate a warning if called from outside of a function definition.

If *arg_num* is greater than the number of arguments actually passed, a warning will be generated and **func_get_arg()** will return **FALSE**.

```
<?php
function foo() {
     $numargs = func_num_args();
     echo "Number of arguments: $numargs<br>\n";
     if ($numargs >= 2) {
     echo "Second argument is: " . func_get_arg (1) . "<br>\n";
     }
}

foo (1, 2, 3);
?>
```

func_get_arg() may be used in conjunction with **func_num_args()** and **func_get_args()** to allow user-defined functions to accept variable-length argument lists.

> **Note:** This function was added in PHP 4.

func_get_args

(PHP 4)

func_get_args -- Returns an array comprising a function's argument list

Description

array **func_get_args** (void)

Returns an array in which each element is the corresponding member of the current user-defined function's argument list. **func_get_args()** will generate a warning if called from outside of a function definition.

```
<?php
function foo() {
    $numargs = func_num_args();
    echo "Number of arguments: $numargs<br>\n";
    if ($numargs >= 2) {
    echo "Second argument is: " . func_get_arg (1) . "<br>\n";
    }
    $arg_list = func_get_args();
    for ($i = 0; $i < $numargs; $i++) {
    echo "Argument $i is: " . $arg_list[$i] . "<br>\n";
    }
}
```

```
foo (1, 2, 3);
?>
```

func_get_args() may be used in conjunction with **func_num_args()** and **func_get_arg()** to allow user-defined functions to accept variable-length argument lists.

Note: This function was added in PHP 4.

func_num_args

(PHP 4)

func_num_args -- Returns the number of arguments passed to the function

Description

int **func_num_args** (void)

Returns the number of arguments passed into the current user-defined function. **func_num_args()** will generate a warning if called from outside of a user-defined function.

```
<?php
function foo() {
    $numargs = func_num_args();
    echo "Number of arguments: $numargs\n";
}

foo (1, 2, 3);    // Prints 'Number of arguments: 3'
?>
```

func_num_args() may be used in conjunction with **func_get_arg()** and **func_get_args()** to allow user-defined functions to accept variable-length argument lists.

function_exists

(PHP 3>= 3.0.7, PHP 4)

function_exists -- Return **TRUE** if the given function has been defined

Description

bool **function_exists** (string function_name)

Checks the list of defined functions, both built-in (internal) and user-defined, for *function_name*. Returns **TRUE** on success or **FALSE** on failure.

```
if (function_exists('imap_open')) {
    echo "IMAP functions are available.<br>\n";
} else {
    echo "IMAP functions are not available.<br>\n";
}
```

Note that a function name may exist even if the function itself is unusable due to configuration or compiling options (with the image functions being an example). Also note that **function_exists()** will return **FALSE** for constructs, such as **include_once()** and **echo()**.

See also **method_exists()** and **get_defined_functions()**.

get_defined_functions

(PHP 4 >= 4.0.4)

get_defined_functions -- Returns an array of all defined functions

Description

array **get_defined_functions** (void)

This function returns an multidimensional array containing a list of all defined functions, both built-in (internal) and user-defined. The internal functions will be accessible via `$arr["internal"]`, and the user defined ones using `$arr["user"]` (see example below).

```
function myrow($id, $data) {
    return "<tr><th>$id</th><td>$data</td></tr>\n";
}

$arr = get_defined_functions();

print_r($arr);
```

Will output something along the lines of:

```
Array
(
    [internal] => Array
        (
            [0] => zend_version
            [1] => func_num_args
            [2] => func_get_arg
            [3] => func_get_args
            [4] => strlen
            [5] => strcmp
            [6] => strncmp
            ...
            [750] => bcscale
            [751] => bccomp
        )

    [user] => Array
        (
            [0] => myrow
        )

)
```

See also **get_defined_vars()** and **get_defined_constants()**.

register_shutdown_function

(PHP 3>= 3.0.4, PHP 4)

register_shutdown_function -- Register a function for execution on shutdown

Description

int **register_shutdown_function** (callback function)

Registers the function named by *function* to be executed when script processing is complete.

Multiple calls to **register_shutdown_function()** can be made, and each will be called in the same order as they were registered. If you call **exit()** within one registered shutdown function, processing will stop completely and no other registered shutdown functions will be called.

The registered shutdown functions are called after the request has been completed (including sending any output buffers), so it is not possible to send output to the browser using **echo()** or **print()**, or retrieve the contents of any output buffers using **ob_get_contents()**.

register_tick_function

(PHP 4 >= 4.0.3)

register_tick_function -- Register a function for execution on each tick

Description

void **register_tick_function** (callback function [, mixed arg])

Registers the function named by *func* to be executed when a tick is called.

unregister_tick_function

(PHP 4 >= 4.0.3)

unregister_tick_function -- De-register a function for execution on each tick

Description

void **unregister_tick_function** (string function_name)

De-registers the function named by *function_name* so it is no longer executed when a tick is called.

XXXV. Gettext

Introduction

The gettext functions implement an NLS (Native Language Support) API which can be used to internationalize your PHP applications. Please see the gettext documentation for your system for a thorough explanation of these functions or view the docs at http://www.gnu.org/manual/gettext/index.html.

Requirements

To use these functions you must download and install the GNU gettext package from
http://www.gnu.org/software/gettext/gettext.html

Installation

To include GNU gettext support in your PHP build you must add the option `--with-gettext[=DIR]` where DIR is the gettext install directory, defaults to `/usr/local`.

> **Note to Win32 Users:** In order to enable this module on a Windows environment, you must copy *gnu_gettext.dll* from the DLL folder of the PHP/Win32 binary package to the SYSTEM32 folder of your windows machine. (Ex: C:\WINNT\SYSTEM32 or C:\WINDOWS\SYSTEM32). Starting with PHP 4.2.3 the name changed to *libintl-1.dll*

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Table of Contents

dcgettext -- Overrides the domain for a single lookup
dcngettext -- Plural version of dcgettext
dgettext -- Override the current domain
dngettext -- Plural version of dgettext
gettext -- Lookup a message in the current domain
ngettext -- Plural version of gettext
textdomain -- Sets the default domain

bind_textdomain_codeset

(PHP 4 >= 4.2.0)

bind_textdomain_codeset -- Specify the character encoding in which the messages from the DOMAIN message catalog will be returned

Description

string **bind_textdomain_codeset** (string domain, string codeset)

Warning
This function is currently not documented; only the argument list is available.

bindtextdomain

(PHP 3>= 3.0.7, PHP 4)

bindtextdomain -- Sets the path for a domain

Description

string **bindtextdomain** (string domain, string directory)

The **bindtextdomain()** function sets the path for a domain.

dcgettext

(PHP 3>= 3.0.7, PHP 4)

dcgettext -- Overrides the domain for a single lookup

Description

string **dcgettext** (string domain, string message, int category)

This function allows you to override the current domain for a single message lookup. It also allows you to specify a *category*.

dcngettext

(PHP 4 >= 4.2.0)

dcngettext -- Plural version of dcgettext

Description

string **dcngettext** (string domain, string msgid1, string msgid2, int n, int category)

Warning
This function is currently not documented; only the argument list is available.

dgettext

(PHP 3>= 3.0.7, PHP 4)

dgettext -- Override the current domain

Description

string **dgettext** (string domain, string message)

The **dgettext()** function allows you to override the current domain for a single message lookup.

dngettext

(PHP 4 >= 4.2.0)

dngettext -- Plural version of dgettext

Description

string **dngettext** (string domain, string msgid1, string msgid2, int n)

Warning
This function is currently not documented; only the argument list is available.

gettext

(PHP 3>= 3.0.7, PHP 4)

gettext -- Lookup a message in the current domain

Description

string **gettext** (string message)

This function returns a translated string if one is found in the translation table, or the submitted message if not found. You may use the underscore character '_' as an alias to this function.

Example 1. gettext()-check

```
<?php
// Set language to German
setlocale(LC_ALL, 'de');

// Specify location of translation tables
bindtextdomain("myPHPApp", "./locale");

// Choose domain
textdomain("myPHPApp");

// Print a test message
print gettext("Welcome to My PHP Application");

// Or use the alias _() for gettext()
print _("Have a nice day");
?>
```

ngettext

(PHP 4 >= 4.2.0)

ngettext -- Plural version of gettext

Description

string **ngettext** (string msgid1, string msgid2, int n)

Warning
This function is currently not documented; only the argument list is available.

textdomain

(PHP 3>= 3.0.7, PHP 4)

textdomain -- Sets the default domain

Description

string **textdomain** (string text_domain)

This function sets the domain to search within when calls are made to **gettext()**, usually the named after an application. The previous default domain is returned. Call it with `NULL` as parameter to get the current setting without changing it.

XXXVI. GMP functions

Introduction

These functions allow you to work with arbitrary-length integers using the GNU MP library.

These functions have been added in PHP 4.0.4.

> **Note:** Most GMP functions accept GMP number arguments, defined as `resource` below. However, most of these functions will also accept numeric and string arguments, given that it is possible to convert the latter to a number. Also, if there is a faster function that can operate on integer arguments, it would be used instead of the slower function when the supplied arguments are integers. This is done transparently, so the bottom line is that you can use integers in every function that expects GMP number. See also the **gmp_init()** function.

Warning
If you want to explicitly specify a large integer, specify it as a string. If you don't do that, PHP will interpret the integer-literal first, possibly resulting in loss of precision, even before `GMP` comes into play.

> **Note:** This extension is not available on Windows platforms.

Requirements

You can download the GMP library from http://www.swox.com/gmp/. This site also has the GMP manual available.

You will need GMP version 2 or better to use these functions. Some functions may require more recent version of the GMP library.

Installation

In order to have these functions available, you must compile PHP with GMP support by using the `--with-gmp` option.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

`GMP_ROUND_ZERO` (integer)

`GMP_ROUND_PLUSINF` (integer)

`GMP_ROUND_MINUSINF` (integer)

Examples

Example 1. Factorial function using GMP

```
<?php
function fact ($x) {
    if ($x <= 1)
        return 1;
    else
        return gmp_mul ($x, fact ($x-1));
}

print gmp_strval (fact (1000)) . "\n";
?>
```

This will calculate factorial of 1000 (pretty big number) very fast.

See Also

More mathmatical functions can be found in the sections BCMath Arbitrary Precision Mathematics Functions and Mathematical Functions.

Table of Contents

gmp_abs

(PHP 4 >= 4.0.4)

gmp_abs -- Absolute value

Description

resource **gmp_abs** (resource a)

Returns absolute value of *a*.

gmp_add

(PHP 4 >= 4.0.4)

gmp_add -- Add numbers

Description

resource **gmp_add** (resource a, resource b)

Add two GMP numbers. The result will be a GMP number representing the sum of the arguments.

gmp_and

(PHP 4 >= 4.0.4)

gmp_and -- Logical AND

Description

resource **gmp_and** (resource a, resource b)

Calculates logical AND of two GMP numbers.

gmp_clrbit

(PHP 4 >= 4.0.4)

gmp_clrbit -- Clear bit

Description

resource **gmp_clrbit** (resource &a, int index)

Clears (sets to 0) bit *index* in *a*.

gmp_cmp

(PHP 4 >= 4.0.4)

gmp_cmp -- Compare numbers

Description

int **gmp_cmp** (resource a, resource b)

Returns a positive value if `a > b`, zero if `a = b` and negative value if `a < b`.

gmp_com

(PHP 4 >= 4.0.4)

gmp_com -- Calculates one's complement of a

Description

resource **gmp_com** (resource a)

Warning
This function is currently not documented; only the argument list is available.

gmp_div_q

(PHP 4 >= 4.0.4)

gmp_div_q -- Divide numbers

Description

resource **gmp_div_q** (resource a, resource b [, int round])

Divides *a* by *b* and returns the integer result. The result rounding is defined by the *round*, which can have the following values:

- *GMP_ROUND_ZERO*: The result is truncated towards 0.
- *GMP_ROUND_PLUSINF*: The result is rounded towards `+infinity`.
- *GMP_ROUND_MINUSINF*: The result is rounded towards `-infinity`.

This function can also be called as **gmp_div()**.

See also **gmp_div_r()**, **gmp_div_qr()**

gmp_div_qr

(PHP 4 >= 4.0.4)

gmp_div_qr -- Divide numbers and get quotient and remainder

Description

array **gmp_div_qr** (resource n, resource d [, int round])

The function divides *n* by *d* and returns array, with the first element being `[n/d]` (the integer result of the division) and the second being `(n - [n/d] * d)` (the remainder of the division).

See the **gmp_div_q()** function for description of the *round* argument.

Example 1. Division of GMP numbers

```
<?php
    $a = gmp_init ("0x41682179fbf5");
    $res = gmp_div_qr ($a, "0xDEFE75");
    printf("Result is: q - %s, r - %s",
            gmp_strval ($res[0]), gmp_strval ($res[1]));
?>
```

See also **gmp_div_q()**, **gmp_div_r()**.

gmp_div_r

(PHP 4 >= 4.0.4)

gmp_div_r -- Remainder of the division of numbers

Description

resource **gmp_div_r** (resource n, resource d [, int round])

Calculates remainder of the integer division of *n* by *d*. The remainder has the sign of the *n* argument, if not zero.

See the **gmp_div_q()** function for description of the *round* argument.

See also **gmp_div_q()**, **gmp_div_qr()**

gmp_div

(PHP 4 >= 4.0.4)

gmp_div -- Divide numbers

Description

resource **gmp_div** (resource a, resource b)

This function is an alias to **gmp_div_q()**.

gmp_divexact

(PHP 4 >= 4.0.4)

gmp_divexact -- Exact division of numbers

Description

resource **gmp_divexact** (resource n, resource d)

Divides *n* by *d*, using fast "exact division" algorithm. This function produces correct results only when it is known in advance that *d* divides *n*.

gmp_fact

(PHP 4 >= 4.0.4)

gmp_fact -- Factorial

Description

resource **gmp_fact** (int a)

Calculates factorial (`a!`) of *a*.

gmp_gcd

(PHP 4 >= 4.0.4)

gmp_gcd -- Calculate GCD

Description

resource **gmp_gcd** (resource a, resource b)

Calculate greatest common divisor of *a* and *b*. The result is always positive even if either of, or both, input operands are negative.

gmp_gcdext

(PHP 4 >= 4.0.4)

gmp_gcdext -- Calculate GCD and multipliers

Description

array **gmp_gcdext** (resource a, resource b)

Calculates g, s, and t, such that `a*s + b*t = g = gcd(a,b)`, where gcd is the greatest common divisor. Returns an array with respective elements g, s and t.

gmp_hamdist

(PHP 4 >= 4.0.4)

gmp_hamdist -- Hamming distance

Description

int **gmp_hamdist** (resource a, resource b)

Returns the hamming distance between *a* and *b*. Both operands should be non-negative.

gmp_init

(PHP 4 >= 4.0.4)

gmp_init -- Create GMP number

Description

resource **gmp_init** (mixed number)

Creates a GMP number from an integer or string. String representation can be decimal or hexadecimal. In the latter case, the string should start with `0x`.

Example 1. Creating GMP number

```
<?php
    $a = gmp_init (123456);
    $b = gmp_init ("0xFFFFDEBACDFEDF7200");
?>
```

> **Note:** It is not necessary to call this function if you want to use integer or string in place of GMP number in GMP functions, like **gmp_add()**. Function arguments are automatically converted to GMP numbers, if such conversion is possible and needed, using the same rules as **gmp_init()**.

gmp_intval

(PHP 4 >= 4.0.4)

gmp_intval -- Convert GMP number to integer

Description

int **gmp_intval** (resource gmpnumber)

This function allows to convert GMP number to integer.

Warning
This function returns a useful result only if the number actually fits the PHP integer (i.e., signed long type). If you want just to print the GMP number, use **gmp_strval()**.

gmp_invert

(PHP 4 >= 4.0.4)

gmp_invert -- Inverse by modulo

Description

resource **gmp_invert** (resource a, resource b)

Computes the inverse of *a* modulo *b*. Returns **FALSE** if an inverse does not exist.

gmp_jacobi

(PHP 4 >= 4.0.4)

gmp_jacobi -- Jacobi symbol

Description

int **gmp_jacobi** (resource a, resource p)

Computes Jacobi symbol of *a* and *p*. *p* should be odd and must be positive.

gmp_legendre

(PHP 4 >= 4.0.4)

gmp_legendre -- Legendre symbol

Description

int **gmp_legendre** (resource a, resource p)

Compute the Legendre symbol of *a* and *p*. *p* should be odd and must be positive.

gmp_mod

(PHP 4 >= 4.0.4)

gmp_mod -- Modulo operation

Description

resource **gmp_mod** (resource n, resource d)

Calculates *n* modulo *d*. The result is always non-negative, the sign of *d* is ignored.

gmp_mul

(PHP 4 >= 4.0.4)

gmp_mul -- Multiply numbers

Description

resource **gmp_mul** (resource a, resource b)

Multiplies *a* by *b* and returns the result.

gmp_neg

(PHP 4 >= 4.0.4)

gmp_neg -- Negate number

Description

resource **gmp_neg** (resource a)

Returns -*a*.

gmp_or

(PHP 4 >= 4.0.4)

gmp_or -- Logical OR

Description

resource **gmp_or** (resource a, resource b)

Calculates logical inclusive OR of two GMP numbers.

gmp_perfect_square

(PHP 4 >= 4.0.4)

gmp_perfect_square -- Perfect square check

Description

bool **gmp_perfect_square** (resource a)

Returns **TRUE** if *a* is a perfect square, **FALSE** otherwise.

See also: **gmp_sqrt()**, **gmp_sqrtrm()**.

gmp_popcount

(PHP 4 >= 4.0.4)

gmp_popcount -- Population count

Description

int **gmp_popcount** (resource a)

Return the population count of *a*.

gmp_pow

(PHP 4 >= 4.0.4)

gmp_pow -- Raise number into power

Description

resource **gmp_pow** (resource base, int exp)

Raise *base* into power *exp*. The case of 0^0 yields 1. *exp* cannot be negative.

gmp_powm

(PHP 4 >= 4.0.4)

gmp_powm -- Raise number into power with modulo

Description

resource **gmp_powm** (resource base, resource exp, resource mod)

Calculate (*base* raised into power *exp*) modulo *mod*. If *exp* is negative, result is undefined.

gmp_prob_prime

(PHP 4 >= 4.0.4)

gmp_prob_prime -- Check if number is "probably prime"

Description

int **gmp_prob_prime** (resource a [, int reps])

If this function returns 0, *a* is definitely not prime. If it returns 1, then *a* is "probably" prime. If it returns 2, then *a* is surely prime.

Reasonable values of *reps* vary from 5 to 10 (default being 10); a higher value lowers the probability for a non-prime to pass as a "probable" prime.

The function uses Miller-Rabin's probabilistic test.

gmp_random

(PHP 4 >= 4.0.4)

gmp_random -- Random number

Description

resource **gmp_random** (int limiter)

Generate a random number. The number will be between *limiter* and zero in value. If *limiter* is negative, negative numbers are generated.

gmp_scan0

(PHP 4 >= 4.0.4)

gmp_scan0 -- Scan for 0

Description

int **gmp_scan0** (resource a, int start)

Scans *a*, starting with bit *start*, towards more significant bits, until the first clear bit is found. Returns the index of the found bit.

gmp_scan1

(PHP 4 >= 4.0.4)

gmp_scan1 -- Scan for 1

Description

int **gmp_scan1** (resource a, int start)

Scans *a*, starting with bit *start*, towards more significant bits, until the first set bit is found. Returns the index of the found bit.

gmp_setbit

(PHP 4 >= 4.0.4)

gmp_setbit -- Set bit

Description

resource **gmp_setbit** (resource &a, int index [, bool set_clear])

Sets bit *index* in *a*. *set_clear* defines if the bit is set to 0 or 1. By default the bit is set to 1.

gmp_sign

(PHP 4 >= 4.0.4)

gmp_sign -- Sign of number

Description

int **gmp_sign** (resource a)

Return sign of *a* - 1 if *a* is positive and -1 if it's negative.

gmp_sqrt

(PHP 4 >= 4.0.4)

gmp_sqrt -- Square root

Description

resource **gmp_sqrt** (resource a)

Calculates square root of *a*.

gmp_sqrtrm

(no version information, might be only in CVS)

gmp_sqrtrm -- Square root with remainder

Description

array **gmp_sqrtrm** (resource a)

Returns array where first element is the integer square root of *a* (see also **gmp_sqrt()**), and the second is the remainder (i.e., the difference between *a* and the first element squared).

gmp_strval

(PHP 4 >= 4.0.4)

gmp_strval -- Convert GMP number to string

Description

string **gmp_strval** (resource gmpnumber [, int base])

Convert GMP number to string representation in base *base*. The default base is 10. Allowed values for the base are from 2 to 36.

Example 1. Converting a GMP number to a string

```
<?php
    $a = gmp_init("0x41682179fbf5");
    printf ("Decimal: %s, 36-based: %s", gmp_strval($a), gmp_strval($a,36));
?>
```

gmp_sub

(PHP 4 >= 4.0.4)

gmp_sub -- Subtract numbers

Description

resource **gmp_sub** (resource a, resource b)

Subtracts *b* from *a* and returns the result.

gmp_xor

(PHP 4 >= 4.0.4)

gmp_xor -- Logical XOR

Description

resource **gmp_xor** (resource a, resource b)

Calculates logical exclusive OR (XOR) of two GMP numbers.

XXXVII. HTTP functions

Introduction

These functions let you manipulate the output sent back to the remote browser right down to the HTTP protocol level.

Requirements

No external libraries are needed to build this extension.

Installation

There is no installation needed to use these functions; they are part of the PHP core.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Table of Contents

header

(PHP 3, PHP 4)

header -- Send a raw HTTP header

Description

int **header** (string string [, bool replace [, int http_reponse_code]])

header() is used to send raw HTTP headers. See the HTTP/1.1 specification for more information on HTTP headers.

The optional *replace* parameter indicates whether the header should replace a previous similar header, or add a second header of the same type. By default it will replace, but if you pass in **FALSE** as the second argument you can force multiple headers of the same type. For example:

```
<?php
header('WWW-Authenticate: Negotiate');
header('WWW-Authenticate: NTLM', FALSE);
?>
```

The second optional *http_response_code* force the HTTP response code to the specified value. (This parameter is available in PHP 4.3.0 and higher.)

There are two special-case header calls. The first is a header that starts with the string "HTTP/" (case is not significant), which will be used to figure out the HTTP status code to send. For example, if you have configured Apache to use a PHP script to handle requests for missing files (using the ErrorDocument directive), you may want to make sure that your script generates the proper status code.

```
<?php
  header("HTTP/1.0 404 Not Found");
?>
```

> **Note:** The HTTP status header line will always be the first sent to the client, regardless of the actual **header()** call beeing the first or not. The status may be overridden by calling **header()** with a new status line at any time unless the HTTP headers have already been sent.
>
> **Note:** In PHP 3, this only works when PHP is compiled as an Apache module. You can achieve the same effect using the Status header.
>
> ```
> <?php
> header("Status: 404 Not Found");
> ?>
> ```

The second special case is the "Location:" header. Not only does it send this header back to the browser, but it also returns a REDIRECT (302) status code to the browser unless some 3xx status code has already been set.

```
<?php
header("Location: http://www.example.com/"); /* Redirect browser */
exit;                /* Make sure that code below does
                        not get executed when we redirect. */
?>
```

> **Note:** HTTP/1.1 requires an absolute URI as argument to Location: including the scheme, hostname and absolute path, but some clients accept relative URIs. You can usually use $_SERVER['HTTP_HOST'], $_SERVER['PHP_SELF'] and **dirname()** to make an absolute URI from a relative one yourself:
>
> ```
> <?php
> header("Location: http://".$_SERVER['HTTP_HOST']
>                     .dirname($_SERVER['PHP_SELF'])
>                     ."/".$relative_url);
> ?>
> ```

PHP scripts often generate dynamic content that must not be cached by the client browser or any proxy caches between the server and the client browser. Many proxies and clients can be forced to disable caching with

```
<?php
header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");    // Date in the past
header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");
                                                     // always modified
header("Cache-Control: no-store, no-cache, must-revalidate");  // HTTP/1.1
header("Cache-Control: post-check=0, pre-check=0", false);
header("Pragma: no-cache");                          // HTTP/1.0
```

```
?>
```

> **Note:** You may find that your pages aren't cached even if you don't output all of the headers above. There are a number of options that users may be able to set for their browser that change its default caching behavior. By sending the headers above, you should override any settings that may otherwise cause the output of your script to be cached.
>
> Additionally, **session_cache_limiter()** and the `session.cache_limiter` configuration setting can be used to automatically generate the correct caching-related headers when sessions are being used.

Remember that **header()** must be called before any actual output is sent, either by normal HTML tags, blank lines in a file, or from PHP. It is a very common error to read code with **include()**, or **require()**, functions, or another file access function, and have spaces or empty lines that are output before **header()** is called. The same problem exists when using a single PHP/HTML file.

```
<html>
<?php
// This will give an error. Note the output
// above, which is before the header() call
header('Location: http://www.example.com/');
?>
```

> **Note:** In PHP 4, you can use output buffering to get around this problem, with the overhead of all of your output to the browser being buffered in the server until you send it. You can do this by calling **ob_start()** and **ob_end_flush()** in your script, or setting the `output_buffering` configuration directive on in your `php.ini` or server configuration files.

If you want the user to be prompted to save the data you are sending, such as a generated PDF file, you can use the Content-Disposition header to supply a recommended filename and force the browser to display the save dialog.

```
<?php
// We'll be outputting a PDF
header("Content-type: application/pdf");

// It will be called downloaded.pdf
header("Content-Disposition: attachment; filename=downloaded.pdf");

// The PDF source is in original.pdf
readfile('original.pdf');
?>
```

> **Note:** There is a bug in Microsoft Internet Explorer 4.01 that prevents this from working. There is no workaround. There is also a bug in Microsoft Internet Explorer 5.5 that interferes with this, which can be resolved by upgrading to Service Pack 2 or later.
>
> **Note:** If safe mode is enabled the uid of the script is added to the `realm` part of the `WWW-Authenticate` header if you set this header (used for HTTP Authentication).

See also **headers_sent()**, **setcookie()**, and the section on HTTP authentication.

headers_sent

(PHP 3>= 3.0.8, PHP 4)

headers_sent -- Checks if or where headers have been sent

Description

boolean **headers_sent** ([string &file [, int &line]])

headers_sent() will return **FALSE** if no HTTP headers have already been sent or **TRUE** otherwise. If the optional *file* and *line* parameters are set, **headers_sent()** will put the php source file name and line number where output started in the *file* and *line* variables.

You can't add any more header lines using the **header()** function once the header block has already been sent. Using this function you can at least prevent getting HTTP header related error messages. Another option is to use Output Buffering.

> **New parameters:** The optional *file* and *line* parameters where added in PHP 4.3.0.

Example 1. Examples using headers_sent()

```
<?php

// If no headers are sent, send one
```

```
if (!headers_sent()) {
    header ('Location: http://www.example.com/');
    exit;
}

// An example using the optional file and line parameters, as of PHP 4.3.0
// Note that $filename and $linenum are passed in for later use.
// Do not assign them values beforehand.
if (!headers_sent($filename, $linenum)) {
    header ('Location: http://www.example.com/');
    exit;

// You would most likely trigger an error here.
} else {

    print "Headers already sent in $filename on line $linenum\n" .
          "Cannot redirect, for now please click this <a " .
          "href=\"http://www.example.com\">link</a> instead\n";
    exit;
}

?>
```

See also **ob_start()**, **trigger_error()**, and **header()** for a more detailed discussion of the matters involved.

setcookie

(PHP 3, PHP 4)

setcookie -- Send a cookie

Description

boolean **setcookie** (string name [, string value [, int expire [, string path [, string domain [, int secure]]]]])

setcookie() defines a cookie to be sent along with the rest of the HTTP headers. Like other headers, cookies must be sent *before* any output from your script (this is a protocol restriction). This requires that you place calls to this function prior to any output, including <html> and <head> tags as well as any whitespace. If output exists prior to calling this function, **setcookie()** will fail and return **FALSE**. If **setcookie()** successfully runs, it will return **TRUE**. This does not indicate whether the user accepted the cookie.

All the arguments except the *name* argument are optional. If only the name argument is present, the cookie by that name will be deleted from the remote client. You may also replace an argument with an empty string ("") in order to skip that argument. Because the *expire* and *secure* arguments are integers, they cannot be skipped with an empty string, use a zero (*0*) instead. The following table explains each parameter of the **setcookie()** function, be sure to read the Netscape cookie specification for specifics.

Table 1. setcookie() parameters explained

Parameter	Description	Examples
name	The name of the cookie.	'cookiename' is called as `$_COOKIE['cookiename']`
value	The value of the cookie. This value is stored on the clients computer; do not store sensitive information.	Assuming the *name* is 'cookiename', this value is retrieved through `$_COOKIE['cookiename']`
expire	The time the cookie expires. This is a unix timestamp so is in number of seconds since the epoch. In otherwords, you'll most likely set this with the **time()** function plus the number of seconds before you want it to expire. Or you might use **mktime()**.	`time()+60*60*24*30` will set the cookie to expire in 30 days. If not set, the cookie will expire at the end of the session (when the browser closes).
path	The path on the server in which the cookie will be available on.	If set to `'/'`, the cookie will be available within the entire *domain*. If set to `'/foo/'`, the cookie will only be available within the `/foo/` directory and all sub-directories such as `/foo/bar/` of *domain*. The default value is the current directory that the cookie is being set in.
domain	The domain that the cookie is available.	To make the cookie available on all subdomains of example.com then you'd set it to `'.example.com'`. The `.` is not required but makes it compatible with more browsers. Setting it to `www.example.com` will make the cookie only available in the `www` subdomain. Refer to tail matching in the spec for details.

Parameter	Description	Examples
secure	Indicates that the cookie should only be transmitted over a secure HTTPS connection. When set to 1, the cookie will only be set if a secure connection exists. The default is 0.	0 or 1

Once the cookies have been set, they can be accessed on the next page load with the $_COOKIE or $HTTP_COOKIE_VARS arrays. Note, autoglobals such as $_COOKIE became available in PHP 4.1.0. $HTTP_COOKIE_VARS has existed since PHP 3. Cookie values also exist in $_REQUEST.

> **Note:** If the PHP directive register_globals is set to on then cookie values will also be made into variables. In our examples below, $TextCookie will exist. It's recommended to use $_COOKIE.

Common Pitfalls:

- Cookies will not become visible until the next loading of a page that the cookie should be visible for. To test if a cookie was successfully set, check for the cookie on a next loading page before the cookie expires. Expire time is set via the *expire* parameter. A nice way to debug the existence of cookies is by simply calling `print_r($_COOKIE);`.
- Cookies must be deleted with the same parameters as they were set with.
- Cookies names can be set as array names and will be available to your PHP scripts as arrays but seperate cookies are stored on the users system. Consider **explode()** or **serialize()** to set one cookie with multiple names and values.

In PHP 3, multiple calls to **setcookie()** in the same script will be performed in reverse order. If you are trying to delete one cookie before inserting another you should put the insert before the delete. In PHP 4, multiple calls to **setcookie()** are performed in the order called.

Some examples follow how to send cookies:

Example 1. setcookie() send examples

```
<?php
$value = 'something from somewhere';

setcookie ("TestCookie", $value);
setcookie ("TestCookie", $value,time()+3600);  /* expire in 1 hour */
setcookie ("TestCookie", $value,time()+3600, "/~rasmus/", ".example.com", 1);
?>
```

Note that the value portion of the cookie will automatically be urlencoded when you send the cookie, and when it is received, it is automatically decoded and assigned to a variable by the same name as the cookie name. To see the contents of our test cookie in a script, simply use one of the following examples:

```
<?php
// Print an individual cookie
echo $_COOKIE["TestCookie"];
echo $HTTP_COOKIE_VARS["TestCookie"];

// Another way to debug/test is to view all cookies
print_r($_COOKIE);
?>
```

When deleting a cookie you should assure that the expiration date is in the past, to trigger the removal mechanism in your browser. Examples follow how to delete cookies sent in previous example:

Example 2. setcookie() delete examples

```
<?php
// set the expiration date to one hour ago
setcookie ("TestCookie", "", time() - 3600);
setcookie ("TestCookie", "", time() - 3600, "/~rasmus/", ".example.com", 1);
?>
```

You may also set array cookies by using array notation in the cookie name. This has the effect of setting as many cookies as you have array elements, but when the cookie is received by your script, the values are all placed in an array with the cookie's name:

```
<?php
// set the cookies
setcookie ("cookie[three]", "cookiethree");
setcookie ("cookie[two]", "cookietwo");
setcookie ("cookie[one]", "cookieone");

// after the page reloads, print them out
if (isset($_COOKIE['cookie'])) {
    foreach ($_COOKIE['cookie'] as $name => $value) {
        echo "$name : $value <br />\n";
    }
```

```
}

/* which prints

three : cookiethree
two : cookietwo
one : cookieone

*/
?>
```

For more information on cookies, see Netscape's cookie specification at http://www.netscape.com/newsref/std/cookie_spec.html.

Microsoft Internet Explorer 4 with Service Pack 1 applied does not correctly deal with cookies that have their path parameter set.

Netscape Communicator 4.05 and Microsoft Internet Explorer 3.x appear to handle cookies incorrectly when the path and time are not set.

XXXVIII. Hyperwave functions

Introduction

Hyperwave has been developed at IICM in Graz. It started with the name Hyper-G and changed to Hyperwave when it was commercialised (in 1996).

Hyperwave is not free software. The current version, 5.5 is available at http://www.hyperwave.com/. A time limited version can be ordered for free (30 days).

See also the Hyperwave API module.

Hyperwave is an information system similar to a database (HIS, Hyperwave Information Server). Its focus is the storage and management of documents. A document can be any possible piece of data that may as well be stored in file. Each document is accompanied by its object record. The object record contains meta data for the document. The meta data is a list of attributes which can be extended by the user. Certain attributes are always set by the Hyperwave server, other may be modified by the user. An attribute is a name/value pair of the form name=value. The complete object record contains as many of those pairs as the user likes. The name of an attribute does not have to be unique, e.g. a title may appear several times within an object record. This makes sense if you want to specify a title in several languages. In such a case there is a convention, that each title value is preceded by the two letter language abbreviation followed by a colon, e.g. `'en:Title in English'` or `'ge:Titel in deutsch'`. Other attributes like a description or keywords are potential candidates. You may also replace the language abbreviation by any other string as long as it separated by colon from the rest of the attribute value.

Each object record has native a string representation with each name/value pair separated by a newline. The Hyperwave extension also knows a second representation which is an associated array with the attribute name being the key. Multilingual attribute values itself form another associated array with the key being the language abbreviation. Actually any multiple attribute forms an associated array with the string left to the colon in the attribute value being the key. (This is not fully implemented. Only the attributes Title, Description and Keyword are treated properly yet.)

Besides the documents, all hyper links contained in a document are stored as object records as well. Hyper links which are in a document will be removed from it and stored as individual objects, when the document is inserted into the database. The object record of the link contains information about where it starts and where it ends. In order to gain the original document you will have to retrieve the plain document without the links and the list of links and reinsert them. The functions **hw_pipedocument()** and **hw_gettext()** do this for you. The advantage of separating links from the document is obvious. Once a document to which a link is pointing to changes its name, the link can easily be modified accordingly. The document containing the link is not affected at all. You may even add a link to a document without modifying the document itself.

Saying that **hw_pipedocument()** and **hw_gettext()** do the link insertion automatically is not as simple as it sounds. Inserting links implies a certain hierarchy of the documents. On a web server this is given by the file system, but Hyperwave has its own hierarchy and names do not reflect the position of an object in that hierarchy. Therefore creation of links first of all requires a mapping from the Hyperwave hierarchy and namespace into a web hierarchy respective web namespace. The fundamental difference between Hyperwave and the web is the clear distinction between names and hierarchy in Hyperwave. The name does not contain any information about the objects position in the hierarchy. In the web the name also contains the information on where the object is located in the hierarchy. This leads to two possibles ways of mapping. Either the Hyperwave hierarchy and name of the Hyperwave object is reflected in the URL or the name only. To make things simple the second approach is used. Hyperwave object with name `my_object` is mapped to `http://host/my_object` disregarding where it resides in the Hyperwave hierarchy. An object with name `parent/my_object` could be the child of `my_object` in the Hyperwave hierarchy, though in a web namespace it appears to be just the opposite and the user might get confused. This can only be prevented by selecting reasonable object names.

Having made this decision a second problem arises. How do you involve PHP? The URL `http://host/my_object` will not call any PHP script unless you tell your web server to rewrite it to e.g. `http://host/php_script/my_object` and the script `php_script`

evaluates the `$PATH_INFO` variable and retrieves the object with name `my_object` from the Hyperwave server. Their is just one little drawback which can be fixed easily. Rewriting any URL would not allow any access to other document on the web server. A PHP script for searching in the Hyperwave server would be impossible. Therefore you will need at least a second rewriting rule to exclude certain URLs like all e.g. starting with `http://host/Hyperwave` This is basically sharing of a namespace by the web and Hyperwave server.

Based on the above mechanism links are insert into documents.

It gets more complicated if PHP is not run as a server module or CGI script but as a standalone application e.g. to dump the content of the Hyperwave server on a CD-ROM. In such a case it makes sense to retain the Hyperwave hierarchy and map in onto the file system. This conflicts with the object names if they reflect its own hierarchy (e.g. by choosing names including '/'). Therefore '/' has to be replaced by another character, e.g. '_'.

The network protocol to communicate with the Hyperwave server is called HG-CSP (Hyper-G Client/Server Protocol). It is based on messages to initiate certain actions, e.g. get object record. In early versions of the Hyperwave Server two native clients (Harmony, Amadeus) were provided for communication with the server. Those two disappeared when Hyperwave was commercialised. As a replacement a so called wavemaster was provided. The wavemaster is like a protocol converter from HTTP to HG-CSP. The idea is to do all the administration of the database and visualisation of documents by a web interface. The wavemaster implements a set of placeholders for certain actions to customise the interface. This set of placeholders is called the PLACE Language. PLACE lacks a lot of features of a real programming language and any extension to it only enlarges the list of placeholders. This has led to the use of JavaScript which IMO does not make life easier.

Adding Hyperwave support to PHP should fill in the gap of a missing programming language for interface customisation. It implements all the messages as defined by the HG-CSP but also provides more powerful commands to e.g. retrieve complete documents.

Hyperwave has its own terminology to name certain pieces of information. This has widely been taken over and extended. Almost all functions operate on one of the following data types.

- object ID: An unique integer value for each object in the Hyperwave server. It is also one of the attributes of the object record (ObjectID). Object ids are often used as an input parameter to specify an object.
- object record: A string with attribute-value pairs of the form attribute=value. The pairs are separated by a carriage return from each other. An object record can easily be converted into an object array with **hw_objectarray()**. Several functions return object records. The names of those functions end with obj.
- object array: An associative array with all attributes of an object. The keys are the attribute names. If an attribute occurs more than once in an object record it will result in another indexed or associative array. Attributes which are language depended (like the title, keyword, description) will form an associative array with the keys set to the language abbreviations. All other multiple attributes will form an indexed array. PHP functions never return object arrays.
- hw_document: This is a complete new data type which holds the actual document, e.g. HTML, PDF etc. It is somewhat optimized for HTML documents but may be used for any format.

Several functions which return an array of object records do also return an associative array with statistical information about them. The array is the last element of the object record array. The statistical array contains the following entries:

Hidden

Number of object records with attribute PresentationHints set to Hidden.

CollectionHead

Number of object records with attribute PresentationHints set to CollectionHead.

FullCollectionHead

Number of object records with attribute PresentationHints set to FullCollectionHead.

CollectionHeadNr

Index in array of object records with attribute PresentationHints set to CollectionHead.

FullCollectionHeadNr

Index in array of object records with attribute PresentationHints set to FullCollectionHead.

Total

Total: Number of object records.

Requirements

This extension needs a Hyperwave server downloadable from http://www.hyperwave.com/.

Installation

To enable Hyperwave support compile PHP `--with-hyperwave`.

Integration with Apache

The Hyperwave extension is best used when PHP is compiled as an Apache module. In such a case the underlying Hyperwave server can be hidden from users almost completely if Apache uses its rewriting engine. The following instructions will explain this.

Since PHP with Hyperwave support built into Apache is intended to replace the native Hyperwave solution based on Wavemaster, we will assume that the Apache server will only serve as a Hyperwave web interface for these examples. This is not necessary but it simplifies the configuration. The concept is quite simple. First of all you need a PHP script which evaluates the `$_ENV['PATH_INFO']` variable and treats its value as the name of a Hyperwave object. Let's call this script `'Hyperwave'`. The URL `http://your.hostname/Hyperwave/name_of_object` would than return the Hyperwave object with the name `'name_of_object'`. Depending on the type of the object the script has to react accordingly. If it is a collection, it will probably return a list of children. If it is a document it will return the mime type and the content. A slight improvement can be achieved if the Apache rewriting engine is used. From the users point of view it would be more straight forward if the URL `http://your.hostname/name_of_object` would return the object. The rewriting rule is quite easy:

```
RewriteRule ^/(.*) /usr/local/apache/htdocs/HyperWave/$1 [L]
```

Now every URL relates to an object in the Hyperwave server. This causes a simple to solve problem. There is no way to execute a different script, e.g. for searching, than the 'Hyperwave' script. This can be fixed with another rewriting rule like the following:

```
RewriteRule ^/hw/(.*) /usr/local/apache/htdocs/hw/$1 [L]
```

This will reserve the directory `/usr/local/apache/htdocs/hw` for additional scripts and other files. Just make sure this rule is evaluated before the one above. There is just a little drawback: all Hyperwave objects whose name starts with `'hw/'` will be shadowed. So, make sure you don't use such names. If you need more directories, e.g. for images just add more rules or place them all in one directory. Before you put those instructions, don't forget to turn on the rewriting engine with

```
RewriteEngine on
```

You will need scripts:

- to return the object itself
- to allow searching
- to identify yourself
- to set your profile
- one for each additional function like to show the object attributes, to show information about users, to show the status of the server, etc.

As an alternative to the Rewrite Engine, you can also consider using the Apache `ErrorDocument` directive, but be aware, that `ErrorDocument` redirected pages cannot receive POST data.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. Hyperwave configuration options

Name	Default	Changeable
hyerwave.allow_persistent	"0"	PHP_INI_SYSTEM
hyperwave.default_port	"418"	PHP_INI_ALL

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

HW_ATTR_LANG (integer)

HW_ATTR_NR (integer)

HW_ATTR_NONE (integer)

Todo

There are still some things to do:

- The hw_InsertDocument has to be split into **hw_insertobject()** and **hw_putdocument()**.
- The names of several functions are not fixed, yet.
- Most functions require the current connection as its first parameter. This leads to a lot of typing, which is quite often not necessary if there is just one open connection. A default connection will improve this.
- Conversion form object record into object array needs to handle any multiple attribute.

Table of Contents

hw_Array2Objrec

(PHP 3>= 3.0.4, PHP 4)

hw_Array2Objrec -- convert attributes from object array to object record

Description

strin **hw_array2objrec** (array object_array)

Converts an *object_array* into an object record. Multiple attributes like 'Title' in different languages are treated properly.

See also **hw_objrec2array()**.

hw_changeobject

(PHP 3>= 3.0.3, PHP 4)

hw_changeobject -- Changes attributes of an object (obsolete)

Description

void **hw_changeobject** (int link, int objid, array attributes)

Warning
This function is currently not documented; only the argument list is available.

hw_Children

(PHP 3>= 3.0.3, PHP 4)

hw_Children -- object ids of children

Description

array **hw_children** (int connection, int objectID)

Returns an array of object ids. Each id belongs to a child of the collection with ID *objectID*. The array contains all children both documents and collections.

hw_ChildrenObj

(PHP 3>= 3.0.3, PHP 4)

hw_ChildrenObj -- object records of children

Description

array **hw_childrenobj** (int connection, int objectID)

Returns an array of object records. Each object record belongs to a child of the collection with ID *objectID*. The array contains all children both documents and collections.

hw_Close

(PHP 3>= 3.0.3, PHP 4)

hw_Close -- closes the Hyperwave connection

Description

int **hw_close** (int connection)

Returns **FALSE** if connection is not a valid connection index, otherwise **TRUE**. Closes down the connection to a Hyperwave server with the given connection index.

hw_Connect

(PHP 3>= 3.0.3, PHP 4)

hw_Connect -- opens a connection

Description

int **hw_connect** (string host, int port, string username, string password)

Opens a connection to a Hyperwave server and returns a connection index on success, or **FALSE** if the connection could not be made. Each of the arguments should be a quoted string, except for the port number. The *username* and *password* arguments are optional and can be left out. In such a case no identification with the server will be done. It is similar to identify as user anonymous. This function returns a connection index that is needed by other Hyperwave functions. You can have multiple connections open at once. Keep in mind, that the password is not encrypted.

See also **hw_pconnect()**.

hw_connection_info

(PHP 3>= 3.0.3, PHP 4)

hw_connection_info -- Prints information about the connection to Hyperwave server

Description

void **hw_connection_info** (int link)

Warning
This function is currently not documented; only the argument list is available.

hw_Cp

(PHP 3>= 3.0.3, PHP 4)

hw_Cp -- copies objects

Description

int **hw_cp** (int connection, array object_id_array, int destination id)

Copies the objects with object ids as specified in the second parameter to the collection with the id *destination id*.

The value return is the number of copied objects.

See also **hw_mv()**.

hw_Deleteobject

(PHP 3>= 3.0.3, PHP 4)

hw_Deleteobject -- deletes object

Description

int **hw_deleteobject** (int connection, int object_to_delete)

Deletes the object with the given object id in the second parameter. It will delete all instances of the object.

Returns **TRUE** if no error occurs otherwise **FALSE**.

See also **hw_mv()**.

hw_DocByAnchor

(PHP 3>= 3.0.3, PHP 4)

hw_DocByAnchor -- object id object belonging to anchor

Description

int **hw_docbyanchor** (int connection, int anchorID)

Returns an th object id of the document to which *anchorID* belongs.

hw_DocByAnchorObj

(PHP 3>= 3.0.3, PHP 4)

hw_DocByAnchorObj -- object record object belonging to anchor

Description

string **hw_docbyanchorobj** (int connection, int anchorID)

Returns an th object record of the document to which *anchorID* belongs.

hw_Document_Attributes

(PHP 3>= 3.0.3, PHP 4)

hw_Document_Attributes -- object record of hw_document

Description

string **hw_document_attributes** (int hw_document)

Returns the object record of the document.

For backward compatibility, **hw_documentattributes()** is also accepted. This is deprecated, however.

See also **hw_document_bodytag()**, and **hw_document_size()**.

hw_Document_BodyTag

(PHP 3>= 3.0.3, PHP 4)

hw_Document_BodyTag -- body tag of hw_document

Description

string **hw_document_bodytag** (int hw_document)

Returns the BODY tag of the document. If the document is an HTML document the BODY tag should be printed before the document.

See also **hw_document_attributes()**, and **hw_document_size()**.

For backward compatibility, **hw_documentbodytag()** is also accepted. This is deprecated, however.

hw_Document_Content

(PHP 3>= 3.0.3, PHP 4)

hw_Document_Content -- returns content of hw_document

Description

string **hw_document_content** (int hw_document)

Returns the content of the document. If the document is an HTML document the content is everything after the BODY tag. Information from the HEAD and BODY tag is in the stored in the object record.

See also **hw_document_attributes()**, **hw_document_size()**, and **hw_document_setcontent()**.

hw_Document_SetContent

(PHP 4)

hw_Document_SetContent -- sets/replaces content of hw_document

Description

string **hw_document_setcontent** (int hw_document, string content)

Sets or replaces the content of the document. If the document is an HTML document the content is everything after the BODY tag. Information from the HEAD and BODY tag is in the stored in the object record. If you provide this information in the content of the document too, the Hyperwave server will change the object record accordingly when the document is inserted. Probably not a very good idea. If this functions fails the document will retain its old content.

See also **hw_document_attributes()**, **hw_document_size()**, and **hw_document_content()**.

hw_Document_Size

(PHP 3>= 3.0.3, PHP 4)

hw_Document_Size -- size of hw_document

Description

int **hw_document_size** (int hw_document)

Returns the size in bytes of the document.

See also **hw_document_bodytag()**, and **hw_document_attributes()**.

For backward compatibility, **hw_documentsize()** is also accepted. This is deprecated, however.

hw_dummy

(PHP 3>= 3.0.3, PHP 4)

hw_dummy -- Hyperwave dummy function

Description

string **hw_dummy** (int link, int id, int msgid)

Warning
This function is currently not documented; only the argument list is available.

hw_EditText

(PHP 3>= 3.0.3, PHP 4)

hw_EditText -- retrieve text document

Description

int **hw_edittext** (int connection, int hw_document)

Uploads the text document to the server. The object record of the document may not be modified while the document is edited. This function will only works for pure text documents. It will not open a special data connection and therefore blocks the control connection during the transfer.

See also **hw_pipedocument()**, **hw_free_document()**, **hw_document_bodytag()**, **hw_document_size()**, **hw_output_document()**, **hw_gettext()**.

hw_Error

(PHP 3>= 3.0.3, PHP 4)

hw_Error -- error number

Description

int **hw_error** (int connection)

Returns the last error number. If the return value is o no error has occurred. The error relates to the last command.

hw_ErrorMsg

(PHP 3>= 3.0.3, PHP 4)

hw_ErrorMsg -- returns error message

Description

string **hw_errormsg** (int connection)

Returns a string containing the last error message or 'No Error'. If **FALSE** is returned, this function failed. The message relates to the last command.

hw_Free_Document

(PHP 3>= 3.0.3, PHP 4)

hw_Free_Document -- frees hw_document

Description

int **hw_free_document** (int hw_document)

Frees the memory occupied by the Hyperwave document.

hw_GetAnchors

(PHP 3>= 3.0.3, PHP 4)

hw_GetAnchors -- object ids of anchors of document

Description

array **hw_getanchors** (int connection, int objectID)

Returns an array of object ids with anchors of the document with object ID *objectID*.

hw_GetAnchorsObj

(PHP 3>= 3.0.3, PHP 4)

hw_GetAnchorsObj -- object records of anchors of document

Description

array **hw_getanchorsobj** (int connection, int objectID)

Returns an array of object records with anchors of the document with object ID *objectID*.

hw_GetAndLock

(PHP 3>= 3.0.3, PHP 4)

hw_GetAndLock -- return bject record and lock object

Description

string **hw_getandlock** (int connection, int objectID)

Returns the object record for the object with ID *objectID*. It will also lock the object, so other users cannot access it until it is unlocked.

See also **hw_unlock()**, and **hw_getobject()**.

hw_GetChildColl

(PHP 3>= 3.0.3, PHP 4)

hw_GetChildColl -- object ids of child collections

Description

array **hw_getchildcoll** (int connection, int objectID)

Returns an array of object ids. Each object ID belongs to a child collection of the collection with ID *objectID*. The function will not return child documents.

See also **hw_children()**, and **hw_getchilddoccoll()**.

hw_GetChildCollObj

(PHP 3>= 3.0.3, PHP 4)

hw_GetChildCollObj -- object records of child collections

Description

array **hw_getchildcollobj** (int connection, int objectID)

Returns an array of object records. Each object records belongs to a child collection of the collection with ID *objectID*. The function will not return child documents.

See also **hw_childrenobj()**, and **hw_getchilddoccollobj()**.

hw_GetChildDocColl

(PHP 3>= 3.0.3, PHP 4)

hw_GetChildDocColl -- object ids of child documents of collection

Description

array **hw_getchilddoccoll** (int connection, int objectID)

Returns array of object ids for child documents of a collection.

See also **hw_children()**, and **hw_getchildcoll()**.

hw_GetChildDocCollObj

(PHP 3>= 3.0.3, PHP 4)

hw_GetChildDocCollObj -- object records of child documents of collection

Description

array **hw_getchilddoccollobj** (int connection, int objectID)

Returns an array of object records for child documents of a collection.

See also **hw_childrenobj()**, and **hw_getchildcollobj()**.

hw_GetObject

(PHP 3>= 3.0.3, PHP 4)

hw_GetObject -- object record

Description

array **hw_getobject** (int connection, [int|array] objectID, string query)

Returns the object record for the object with ID *objectID* if the second parameter is an integer. If the second parameter is an array of integer the function will return an array of object records. In such a case the last parameter is also evaluated which is a query string.

The query string has the following syntax:

<expr> ::= "(" <expr> ")" |

"!" <expr> | /* NOT */

<expr> "||" <expr> | /* OR */

<expr> "&&" <expr> | /* AND */

<attribute> <operator> <value>

<attribute> ::= /* any attribute name (Title, Author, DocumentType ...) */

<operator> ::= "=" | /* equal */

"<" | /* less than (string compare) */

">" | /* greater than (string compare) */

"~" /* regular expression matching */

The query allows to further select certain objects from the list of given objects. Unlike the other query functions, this query may use not indexed attributes. How many object records are returned depends on the query and if access to the object is allowed.

See also **hw_getandlock()**, and **hw_getobjectbyquery()**.

hw_GetObjectByQuery

(PHP 3>= 3.0.3, PHP 4)

hw_GetObjectByQuery -- search object

Description

array **hw_getobjectbyquery** (int connection, string query, int max_hits)

Searches for objects on the whole server and returns an array of object ids. The maximum number of matches is limited to *max_hits*. If *max_hits* is set to -1 the maximum number of matches is unlimited.

The query will only work with indexed attributes.

See also **hw_getobjectbyqueryobj()**.

hw_GetObjectByQueryColl

(PHP 3>= 3.0.3, PHP 4)

hw_GetObjectByQueryColl -- search object in collection

Description

array **hw_getobjectbyquerycoll** (int connection, int objectID, string query, int max_hits)

Searches for objects in collection with ID *objectID* and returns an array of object ids. The maximum number of matches is limited to *max_hits*. If *max_hits* is set to -1 the maximum number of matches is unlimited.

The query will only work with indexed attributes.

See also **hw_getobjectbyquerycollobj()**.

hw_GetObjectByQueryCollObj

(PHP 3>= 3.0.3, PHP 4)

hw_GetObjectByQueryCollObj -- search object in collection

Description

array **hw_getobjectbyquerycollobj** (int connection, int objectID, string query, int max_hits)

Searches for objects in collection with ID *objectID* and returns an array of object records. The maximum number of matches is limited to *max_hits*. If *max_hits* is set to -1 the maximum number of matches is unlimited.

The query will only work with indexed attributes.

See also **hw_getobjectbyquerycoll()**.

hw_GetObjectByQueryObj

(PHP 3>= 3.0.3, PHP 4)

hw_GetObjectByQueryObj -- search object

Description

array **hw_getobjectbyqueryobj** (int connection, string query, int max_hits)

Searches for objects on the whole server and returns an array of object records. The maximum number of matches is limited to *max_hits*. If *max_hits* is set to -1 the maximum number of matches is unlimited.

The query will only work with indexed attributes.

See also **hw_getobjectbyquery()**.

hw_GetParents

(PHP 3>= 3.0.3, PHP 4)

hw_GetParents -- object ids of parents

Description

array **hw_getparents** (int connection, int objectID)

Returns an indexed array of object ids. Each object id belongs to a parent of the object with ID *objectID*.

hw_GetParentsObj

(PHP 3>= 3.0.3, PHP 4)

hw_GetParentsObj -- object records of parents

Description

array **hw_getparentsobj** (int connection, int objectID)

Returns an indexed array of object records plus an associated array with statistical information about the object records. The associated array is the last entry of the returned array. Each object record belongs to a parent of the object with ID *objectID*.

hw_getrellink

(PHP 3>= 3.0.3, PHP 4)

hw_getrellink -- Get link from source to dest relative to rootid

Description

string **hw_getrellink** (int link, int rootid, int sourceid, int destid)

Warning
This function is currently not documented; only the argument list is available.

hw_GetRemote

(PHP 3>= 3.0.3, PHP 4)

hw_GetRemote -- Gets a remote document

Description

int **hw_getremote** (int connection, int objectID)

Returns a remote document. Remote documents in Hyperwave notation are documents retrieved from an external source. Common remote documents are for example external web pages or queries in a database. In order to be able to access external sources throught remote documents Hyperwave introduces the HGI (Hyperwave Gateway Interface) which is similar to the CGI. Currently, only ftp, http-servers and some databases can be accessed by the HGI. Calling **hw_getremote()** returns the document from the external source. If you want to use this function you should be very familiar with HGIs. You should also consider to use PHP instead of Hyperwave to access external sources. Adding database support by a Hyperwave gateway should be more difficult than doing it in PHP.

See also **hw_getremotechildren()**.

hw_GetRemoteChildren

(PHP 3>= 3.0.3, PHP 4)

hw_GetRemoteChildren -- Gets children of remote document

Description

int **hw_getremotechildren** (int connection, string object record)

Returns the children of a remote document. Children of a remote document are remote documents itself. This makes sense if a database query has to be narrowed and is explained in Hyperwave Programmers' Guide. If the number of children is 1 the function will return the document itself formated by the Hyperwave Gateway Interface (HGI). If the number of children is greater than 1 it will return an array of object record with each maybe the input value for another call to **hw_getremotechildren()**. Those object records are virtual and do not exist in the Hyperwave server, therefore they do not have a valid object ID. How exactly such an object record looks like is up to the HGI. If you want to use this function you should be very familiar with HGIs. You should also consider to use PHP instead of Hyperwave to access external sources. Adding database support by a Hyperwave gateway should be more difficult than doing it in PHP.

See also **hw_getremote()**.

hw_GetSrcByDestObj

(PHP 3>= 3.0.3, PHP 4)

hw_GetSrcByDestObj -- Returns anchors pointing at object

Description

array **hw_getsrcbydestobj** (int connection, int objectID)

Returns the object records of all anchors pointing to the object with ID *objectID*. The object can either be a document or an anchor of type destination.

See also **hw_getanchors()**.

hw_GetText

(PHP 3>= 3.0.3, PHP 4)

hw_GetText -- retrieve text document

Description

int **hw_gettext** (int connection, int objectID [, mixed rootID/prefix])

Returns the document with object ID *objectID*. If the document has anchors which can be inserted, they will be inserted already. The optional parameter *rootID/prefix* can be a string or an integer. If it is an integer it determines how links are inserted into the document. The default is 0 and will result in links that are constructed from the name of the link's destination object. This is useful for web applications. If a link points to an object with name 'internet_movie' the HTML link will be <A HREF="/internet_movie">. The actual location of the source and destination object in the document hierachy is disregarded. You will have to set up your web browser, to rewrite that URL to for example '/my_script.php3/internet_movie'. 'my_script.php3' will have to evaluate $PATH_INFO and retrieve the document. All links will have the prefix '/my_script.php3/'. If you do not want this you can set the optional parameter *rootID/prefix* to any prefix which is used instead. Is this case it has to be a string.

If *rootID/prefix* is an integer and unequal to 0 the link is constructed from all the names starting at the object with the id *rootID/prefix* separated by a slash relative to the current object. If for example the above document 'internet_movie' is located at 'a-b-c-internet_movie' with '-' being the seperator between hierachy levels on the Hyperwave server and the source document is located at 'a-b-d-source' the resulting HTML link would be: <A HREF="../c/internet_movie">. This is useful if you want to download the whole server content onto disk and map the document hierachy onto the file system.

This function will only work for pure text documents. It will not open a special data connection and therefore blocks the control connection during the transfer.

See also **hw_pipedocument()**, **hw_free_document()**, **hw_document_bodytag()**, **hw_document_size()**, and **hw_output_document()**.

hw_getusername

(PHP 3>= 3.0.3, PHP 4)

hw_getusername -- name of currently logged in user

Description

string **hw_getusername** (int connection)

Returns the username of the connection.

hw_Identify

(PHP 3>= 3.0.3, PHP 4)

hw_Identify -- identifies as user

Description

int **hw_identify** (string username, string password)

Identifies as user with *username* and *password*. Identification is only valid for the current session. I do not thing this function will be needed very often. In most cases it will be easier to identify with the opening of the connection.

See also **hw_connect()**.

hw_InCollections

(PHP 3>= 3.0.3, PHP 4)

hw_InCollections -- check if object ids in collections

Description

array **hw_incollections** (int connection, array object_id_array, array collection_id_array, int return_collections)

Checks whether a set of objects (documents or collections) specified by the *object_id_array* is part of the collections listed in *collection_id_array*. When the fourth parameter *return_collections* is 0, the subset of object ids that is part of the collections (i.e., the documents or collections that are children of one or more collections of collection ids or their subcollections, recursively) is returned as an array. When the fourth parameter is 1, however, the set of collections that have one or more objects of this subset as children are returned as an array. This option allows a client to, e.g., highlight the part of the collection hierarchy that contains the matches of a previous query, in a graphical overview.

hw_Info

(PHP 3>= 3.0.3, PHP 4)

hw_Info -- info about connection

Description

string **hw_info** (int connection)

Returns information about the current connection. The returned string has the following format: <Serverstring>, <Host>, <Port>, <Username>, <Port of Client>, <Byte swapping>

hw_InsColl

(PHP 3>= 3.0.3, PHP 4)

hw_InsColl -- insert collection

Description

int **hw_inscoll** (int connection, int objectID, array object_array)

Inserts a new collection with attributes as in *object_array* into collection with object ID *objectID*.

hw_InsDoc

(PHP 3>= 3.0.3, PHP 4)

hw_InsDoc -- insert document

Description

int **hw_insdoc** (int connection, int parentID, string object_record, string text)

Inserts a new document with attributes as in *object_record* into collection with object ID *parentID*. This function inserts either an object record only or an object record and a pure ascii text in *text* if *text* is given. If you want to insert a general document of any kind use **hw_insertdocument()** instead.

See also **hw_insertdocument()**, and **hw_inscoll()**.

hw_insertanchors

(PHP 4 >= 4.0.4)

hw_insertanchors -- Inserts only anchors into text

Description

string **hw_insertanchors** (int hwdoc, array anchorecs, array dest [, array urlprefixes])

Warning
This function is currently not documented; only the argument list is available.

hw_InsertDocument

(PHP 3>= 3.0.3, PHP 4)

hw_InsertDocument -- upload any document

Description

int **hw_insertdocument** (int connection, int parent_id, int hw_document)

Uploads a document into the collection with *parent_id*. The document has to be created before with **hw_new_document()**. Make sure that the object record of the new document contains at least the attributes: Type, DocumentType, Title and Name. Possibly you also want to set the MimeType. The functions returns the object id of the new document or **FALSE**.

See also **hw_pipedocument()**.

hw_InsertObject

(PHP 3>= 3.0.3, PHP 4)

hw_InsertObject -- inserts an object record

Description

int **hw_insertobject** (int connection, string object rec, string parameter)

Inserts an object into the server. The object can be any valid hyperwave object. See the HG-CSP documentation for a detailed information on how the parameters have to be.

Note: If you want to insert an Anchor, the attribute Position has always been set either to a start/end value or to 'invisible'. Invisible positions are needed if the annotation has no correspondig link in the annotation text.

See also **hw_pipedocument()**, **hw_insertdocument()**, **hw_insdoc()**, and **hw_inscoll()**.

hw_mapid

(PHP 3>= 3.0.13, PHP 4)

hw_mapid -- Maps global id on virtual local id

Description

int **hw_mapid** (int connection, int server id, int object id)

Maps a global object id on any hyperwave server, even those you did not connect to with **hw_connect()**, onto a virtual object id. This virtual object id can then be used as any other object id, e.g. to obtain the object record with **hw_getobject()**. The server id is the first part of the global object id (GOid) of the object which is actually the IP number as an integer.

Note: In order to use this function you will have to set the F_DISTRIBUTED flag, which can currently only be set at compile time in hg_comm.c. It is not set by default. Read the comment at the beginning of hg_comm.c

hw_Modifyobject

(PHP 3>= 3.0.7, PHP 4)

hw_Modifyobject -- modifies object record

Description

int **hw_modifyobject** (int connection, int object_to_change, array remove, array add, int mode)

This command allows to remove, add, or modify individual attributes of an object record. The object is specified by the Object ID *object_to_change*. The first array *remove* is a list of attributes to remove. The second array *add* is a list of attributes to add. In order to modify an attribute one will have to remove the old one and add a new one. **hw_modifyobject()** will always remove the attributes before it adds attributes unless the value of the attribute to remove is not a string or array.

The last parameter determines if the modification is performed recursively. 1 means recurive modification. If some of the objects cannot be modified they will be skiped without notice. **hw_error()** may not indicate an error though some of the objects could not be modified.

The keys of both arrays are the attributes name. The value of each array element can either be an array, a string or anything else. If it is an array each attribute value is constructed by the key of each element plus a colon and the value of each element. If it is a string it is taken as the attribute value. An empty string will result in a complete removal of that attribute. If the value is neither a string nor an array but something else, e.g. an integer, no operation at all will be performed on the attribute. This is neccessary if you want to to add a completely new attribute not just a new value for an existing attribute. If the remove array contained an empty string for that attribute, the attribute would be tried to be removed which would fail since it doesn't exist. The following addition of a new value for that attribute would also fail. Setting the value for that attribute to e.g. 0 would not even try to remove it and the addition will work.

If you would like to change the attribute 'Name' with the current value 'books' into 'articles' you will have to create two arrays and call **hw_modifyobject()**.

Example 1. modifying an attribute

```
// $connect is an existing connection to the Hyperwave server
       // $objid is the ID of the object to modify
       $remarr = array("Name" => "books");
       $addarr = array("Name" => "articles");
       $hw_modifyobject($connect, $objid, $remarr, $addarr);
```

In order to delete/add a name=value pair from/to the object record just pass the remove/add array and set the last/third

parameter to an empty array. If the attribute is the first one with that name to add, set attribute value in the remove array to an integer.

Example 2. adding a completely new attribute

```
// $connect is an existing connection to the Hyperwave server
       // $objid is the ID of the object to modify
       $remarr = array("Name" => 0);
       $addarr = array("Name" => "articles");
       $hw_modifyobject($connect, $objid, $remarr, $addarr);
```

> **Note:** Multilingual attributes, e.g. 'Title', can be modified in two ways. Either by providing the attributes value in its native form 'language':'title' or by providing an array with elements for each language as described above. The above example would than be:

Example 3. modifying Title attribute

```
$remarr = array("Title" => "en:Books");
       $addarr = array("Title" => "en:Articles");
       $hw_modifyobject($connect, $objid, $remarr, $addarr);
```

or

Example 4. modifying Title attribute

```
$remarr = array("Title" => array("en" => "Books"));
       $addarr = array("Title" => array("en" => "Articles", "ge"=>"Artikel"));
       $hw_modifyobject($connect, $objid, $remarr, $addarr);
```

This removes the english title 'Books' and adds the english title 'Articles' and the german title 'Artikel'.

Example 5. removing attribute

```
$remarr = array("Title" => "");
       $addarr = array("Title" => "en:Articles");
       $hw_modifyobject($connect, $objid, $remarr, $addarr);
```

> **Note:** This will remove all attributes with the name 'Title' and adds a new 'Title' attribute. This comes in handy if you want to remove attributes recursively.
>
> **Note:** If you need to delete all attributes with a certain name you will have to pass an empty string as the attribute value.
>
> **Note:** Only the attributes 'Title', 'Description' and 'Keyword' will properly handle the language prefix. If those attributes don't carry a language prefix, the prefix 'xx' will be assigned.
>
> **Note:** The 'Name' attribute is somewhat special. In some cases it cannot be complete removed. You will get an error message 'Change of base attribute' (not clear when this happens). Therefore you will always have to add a new Name first and than remove the old one.
>
> **Note:** You may not suround this function by calls to **hw_getandlock()** and **hw_unlock()**. **hw_modifyobject()** does this internally.

Returns **TRUE** if no error occurs otherwise **FALSE**.

hw_Mv

(PHP 3>= 3.0.3, PHP 4)

hw_Mv -- moves objects

Description

int **hw_mv** (int connection, array object id array, int source id, int destination id)

Moves the objects with object ids as specified in the second parameter from the collection with id *source id* to the collection with the id *destination id*. If the destination id is o the objects will be unlinked from the source collection. If this is the last instance of that object it will be deleted. If you want to delete all instances at once, use **hw_deleteobject()**.

The value return is the number of moved objects.

See also **hw_cp()**, and **hw_deleteobject()**.

hw_New_Document

(PHP 3>= 3.0.3, PHP 4)

hw_New_Document -- create new document

Description

int **hw_new_document** (string object_record, string document_data, int document_size)

Returns a new Hyperwave document with document data set to *document_data* and object record set to *object_record*. The length of the *document_data* has to passed in *document_size*This function does not insert the document into the Hyperwave server.

See also **hw_free_document()**, **hw_document_size()**, **hw_document_bodytag()**, **hw_output_document()**, and **hw_insertdocument()**.

hw_Objrec2Array

(PHP 3>= 3.0.3, PHP 4)

hw_Objrec2Array -- convert attributes from object record to object array

Description

array **hw_objrec2array** (string object_record [, array format])

Converts an *object_record* into an object array. The keys of the resulting array are the attributes names. Multi-value attributes like 'Title' in different languages form its own array. The keys of this array are the left part to the colon of the attribute value. This left part must be two characters long. Other multi-value attributes without a prefix form an indexed array. If the optional parameter is missing the attributes 'Title', 'Description' and 'Keyword' are treated as language attributes and the attributes 'Group', 'Parent' and 'HtmlAttr' as non-prefixed multi-value attributes. By passing an array holding the type for each attribute you can alter this behaviour. The array is an associated array with the attribute name as its key and the value being one of HW_ATTR_LANG or HW_ATTR_NONE

See also **hw_array2objrec()**.

hw_Output_Document

(PHP 3>= 3.0.3, PHP 4)

hw_Output_Document -- prints hw_document

Description

int **hw_output_document** (int hw_document)

Prints the document without the BODY tag.

For backward compatibility, **hw_outputdocument()** is also accepted. This is deprecated, however.

hw_pConnect

(PHP 3>= 3.0.3, PHP 4)

hw_pConnect -- make a persistent database connection

Description

int **hw_pconnect** (string host, int port, string username, string password)

Returns a connection index on success, or **FALSE** if the connection could not be made. Opens a persistent connection to a Hyperwave server. Each of the arguments should be a quoted string, except for the port number. The *username* and *password* arguments are optional and can be left out. In such a case no identification with the server will be done. It is similar to identify as user anonymous. This function returns a connection index that is needed by other Hyperwave functions. You can have multiple persistent connections open at once.

See also **hw_connect()**.

hw_PipeDocument

(PHP 3>= 3.0.3, PHP 4)

hw_PipeDocument -- retrieve any document

Description

int **hw_pipedocument** (int connection, int objectID)

Returns the Hyperwave document with object ID *objectID*. If the document has anchors which can be inserted, they will have been inserted already. The document will be transfered via a special data connection which does not block the control connection.

See also **hw_gettext()** for more on link insertion, **hw_free_document()**, **hw_document_size()**, **hw_document_bodytag()**, and **hw_output_document()**.

hw_Root

(PHP 3>= 3.0.3, PHP 4)

hw_Root -- root object id

Description

int **hw_root** ( )

Returns the object ID of the hyperroot collection. Currently this is always 0. The child collection of the hyperroot is the root collection of the connected server.

hw_setlinkroot

(PHP 3>= 3.0.3, PHP 4)

hw_setlinkroot -- Set the id to which links are calculated

Description

void **hw_setlinkroot** (int link, int rootid)

Warning
This function is currently not documented; only the argument list is available.

hw_stat

(PHP 3>= 3.0.3, PHP 4)

hw_stat -- Returns status string

Description

string **hw_stat** (int link)

Warning
This function is currently not documented; only the argument list is available.

hw_Unlock

(PHP 3>= 3.0.3, PHP 4)

hw_Unlock -- unlock object

Description

int **hw_unlock** (int connection, int objectID)

Unlocks a document, so other users regain access.

See also **hw_getandlock()**.

hw_Who

(PHP 3>= 3.0.3, PHP 4)

hw_Who -- List of currently logged in users

Description

int **hw_who** (int connection)

Returns an array of users currently logged into the Hyperwave server. Each entry in this array is an array itself containing the elements id, name, system, onSinceDate, onSinceTime, TotalTime and self. 'self' is 1 if this entry belongs to the user who initianted the request.

XXXIX. Hyperwave API functions

Introduction

Hyperwave has been developed at IICM in Graz. It started with the name Hyper-G and changed to Hyperwave when it was commercialised (in 1996).

Hyperwave is not free software. The current version, 5.5, is available at http://www.hyperwave.com/. A time limited version can be ordered for free (30 days).

See also the Hyperwave module.

Hyperwave is an information system similar to a database (HIS, Hyperwave Information Server). Its focus is the storage and management of documents. A document can be any possible piece of data that may as well be stored in file. Each document is accompanied by its object record. The object record contains meta data for the document. The meta data is a list of attributes which can be extended by the user. Certain attributes are always set by the Hyperwave server, other may be modified by the user.

Requirements

Since 2001 there is a Hyperwave SDK available. It supports Java, JavaScript and C++. This PHP Extension is based on the C++ interface. In order to activate the hwapi support in PHP you will have to install the Hyperwave SDK first.

Installation

After installing the Hyperwave SDK, configure PHP with `--with-hwapi[=DIR]`.

Integration with Apache

The integration with Apache and possible other servers is already described in the Hyperwave module which has been the first extension to connect a Hyperwave Server.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. Hyperwave API configuration options

Name	Default	Changeable
hwapi.allow_persistent	"0"	PHP_INI_SYSTEM

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Classes

The API provided by the HW_API extension is fully object oriented. It is very similar to the C++ interface of the Hyperwave SDK. It consist of the following classes.

- **HW_API**
- **HW_API_Object**
- **HW_API_Attribute**
- **HW_API_Error**
- **HW_API_Content**
- **HW_API_Reason**

Some basic classes like **HW_API_String**, **HW_API_String_Array**, etc., which exist in the Hyperwave SDK have not been implemented since PHP has powerful replacements for them.

Each class has certain method, whose names are identical to its counterparts in the Hyperwave SDK. Passing arguments to this function differs from all the other PHP extensions but is close to the C++ API of the HW SDK. Instead of passing serval parameters they are all put into an associated array and passed as one paramter. The names of the keys are identical to those documented in the HW SDK. The common parameters are listed below. If other parameters are required they will be documented if needed.

- **objectIdentifier** The name or id of an object, e.g. "rootcollection", "0x873A8768 0x00000002".

- **parentIdentifier** The name or id of an object which is considered to be a parent.
- **object** An instance of class HW_API_Object.
- **parameters** An instance of class HW_API_Object.
- **version** The version of an object.
- **mode** An integer value determine the way an operation is executed.
- **attributeSelector** Any array of strings, each containing a name of an attribute. This is used if you retrieve the object record and want to include certain attributes.
- **objectQuery** A query to select certain object out of a list of objects. This is used to reduce the number of objects which was delivered by a function like **hw_api->children()** or **hw_api->find()**.

Table of Contents

hw_api_attribute->key

(no version information, might be only in CVS)

hw_api_attribute->key -- Returns key of the attribute

Description

string **key** (void)

Returns the name of the attribute.

See also **hwapi_attribute_value()**.

hw_api_attribute->langdepvalue

(no version information, might be only in CVS)

hw_api_attribute->langdepvalue -- Returns value for a given language

Description

string **langdepvalue** (string language)

Returns the value in the given language of the attribute.

See also **hwapi_attribute_value()**.

hw_api_attribute->value

(no version information, might be only in CVS)

hw_api_attribute->value -- Returns value of the attribute

Description

string **value** (void)

Returns the value of the attribute.

See also **hwapi_attribute_key()**, **hwapi_attribute_values()**.

hw_api_attribute->values

(no version information, might be only in CVS)

hw_api_attribute->values -- Returns all values of the attribute

Description

array **values** (void)

Returns all values of the attribute as an array of strings.

See also **hwapi_attribute_value()**.

hw_api_attribute

(no version information, might be only in CVS)

hw_api_attribute -- Creates instance of class hw_api_attribute

Description

object **attribute** ([string name [, string value]])

Creates a new instance of hw_api_attribute with the given name and value.

hw_api->checkin

(no version information, might be only in CVS)

hw_api->checkin -- Checks in an object

Description

object **checkin** (array parameter)

This function checks in an object or a whole hiearchie of objects. The parameters array contains the required element 'objectIdentifier' and the optional element 'version', 'comment', 'mode' and 'objectQuery'. 'version' sets the version of the object. It consists of the major and minor version separated by a period. If the version is not set, the minor version is incremented. 'mode' can be one of the following values:

HW_API_CHECKIN_NORMAL

Checks in and commits the object. The object must be a document.

HW_API_CHECKIN_RECURSIVE

If the object to check in is a collection, all children will be checked in recursively if they are documents. Trying to check in a collection would result in an error.

HW_API_CHECKIN_FORCE_VERSION_CONTROL

Checks in an object even if it is not under version control.

HW_API_CHECKIN_REVERT_IF_NOT_CHANGED

Check if the new version is different from the last version. Unless this is the case the object will be checked in.

HW_API_CHECKIN_KEEP_TIME_MODIFIED

Keeps the time modified from the most recent object.

HW_API_CHECKIN_NO_AUTO_COMMIT

The object is not automatically commited on checkin.

See also **hwapi_checkout()**.

hw_api->checkout

(no version information, might be only in CVS)

hw_api->checkout -- Checks out an object

Description

object **checkout** (array parameter)

This function checks out an object or a whole hiearchie of objects. The parameters array contains the required element 'objectIdentifier' and the optional element 'version', 'mode' and 'objectQuery'. 'mode' can be one of the following values:

HW_API_CHECKIN_NORMAL

Checks out an object. The object must be a document.

HW_API_CHECKIN_RECURSIVE

If the object to check out is a collection, all children will be checked out recursively if they are documents. Trying to check out a collection would result in an error.

See also **hwapi_checkin()**.

hw_api->children

(no version information, might be only in CVS)

hw_api->children -- Returns children of an object

Description

array **children** (array parameter)

Retrieves the children of a collection or the attributes of a document. The children can be further filtered by specifying an object query. The parameter array contains the required elements 'objectIdentifier' and the optional elements 'attributeSelector' and 'objectQuery'.

The return value is an array of objects of type **HW_API_Object** or **HW_API_Error**.

See also **hwapi_parents()**.

hw_api_content->mimetype

(no version information, might be only in CVS)

hw_api_content->mimetype -- Returns mimetype

Description

string **mimetype** (void)

Returns the mimetype of the content.

hw_api_content->read

(no version information, might be only in CVS)

hw_api_content->read -- Read content

Description

string **read** (string buffer, integer len)

Reads *len* bytes from the content into the given buffer.

hw_api->content

(no version information, might be only in CVS)

hw_api->content -- Returns content of an object

Description

object **content** (array parameter)

This function returns the content of a document as an object of type **hw_api_content**. The parameter array contains the required elements 'objectidentifier' and the optional element 'mode'. The mode can be one of the constants

HW_API_CONTENT_ALLLINKS, HW_API_CONTENT_REACHABLELINKS or HW_API_CONTENT_PLAIN. HW_API_CONTENT_ALLLINKS means to insert all anchors even if the destination is not reachable. HW_API_CONTENT_REACHABLELINKS tells **hw_api_content()** to insert only reachable links and HW_API_CONTENT_PLAIN will lead to document without any links.

hw_api->copy

(no version information, might be only in CVS)

hw_api->copy -- Copies physically

Description

object **copy** (array parameter)

This function will make a physical copy including the content if it exists and returns the new object or an error object. The parameter array contains the required elements 'objectIdentifier' and 'destinationParentIdentifier'. The optional parameter is 'attributeSelector'`

See also **hwapi_move()**, **hwapi_link()**.

hw_api->dbstat

(no version information, might be only in CVS)

hw_api->dbstat -- Returns statistics about database server

Description

object **dbstat** (array parameter)

See also **hwapi_dcstat()**, **hwapi_hwstat()**, **hwapi_ftstat()**.

hw_api->dcstat

(no version information, might be only in CVS)

hw_api->dcstat -- Returns statistics about document cache server

Description

object **dcstat** (array parameter)

See also **hwapi_hwstat()**, **hwapi_dbstat()**, **hwapi_ftstat()**.

hw_api->dstanchors

(no version information, might be only in CVS)

hw_api->dstanchors -- Returns a list of all destination anchors

Description

object **dstanchors** (array parameter)

Retrieves all destination anchors of an object. The parameter array contains the required element 'objectIdentifier' and the optional elements 'attributeSelector' and 'objectQuery'.

See also **hwapi_srcanchors()**.

hw_api->dstofsrcanchors

(no version information, might be only in CVS)

hw_api->dstofsrcanchors -- Returns destination of a source anchor

Description

object **dstofsrcanchors** (array parameter)

Retrieves the destination object pointed by the specified source anchors. The destination object can either be a destination anchor or a whole document. The parameters array contains the required element 'objectIdentifier' and the optional element 'attributeSelector'.

See also **hwapi_srcanchors()**, **hwapi_dstanchors()**, **hwapi_objectbyanchor()**.

hw_api_error->count

(no version information, might be only in CVS)

hw_api_error->count -- Returns number of reasons

Description

int **count** (void)

Returns the number of error reasons.

See also **hwapi_error_reason()**.

hw_api_error->reason

(no version information, might be only in CVS)

hw_api_error->reason -- Returns reason of error

Description

object **reason** (void)

Returns the first error reason.

See also **hwapi_error_count()**.

hw_api->find

(no version information, might be only in CVS)

hw_api->find -- Search for objects

Description

array **find** (array parameter)

This functions searches for objects either by executing a key or/and full text query. The found objects can further be filtered by an optional object query. They are sorted by their importance. The second search operation is relatively slow and its result can be limited to a certain number of hits. This allows to perform an incremental search, each returning just a subset of all found documents, starting at a given index. The parameter array contains the 'keyquery' or/and 'fulltextquery' depending on who you would like to search. Optional parameters are 'objectquery', 'scope', 'lanugages' and 'attributeselector'. In case of an incremental search the optional parameters 'startIndex', numberOfObjectsToGet' and 'exactMatchUnit' can be passed.

hw_api->ftstat

(no version information, might be only in CVS)

hw_api->ftstat -- Returns statistics about fulltext server

Description

object **ftstat** (array parameter)

See also **hwapi_dcstat()**, **hwapi_dbstat()**, **hwapi_hwstat()**.

hwapi_hgcsp

(no version information, might be only in CVS)

hwapi_hgcsp -- Returns object of class hw_api

Description

object **hwapi_hgcsp** (string hostname [, int port])

Opens a connection to the Hyperwave server on host *hostname*. The protocol used is HGCSP. If you do not pass a port number, 418 is used.

See also **hwapi_hwtp()**.

hw_api->hwstat

(no version information, might be only in CVS)

hw_api->hwstat -- Returns statistics about Hyperwave server

Description

object **hwstat** (array parameter)

See also **hwapi_dcstat()**, **hwapi_dbstat()**, **hwapi_ftstat()**.

hw_api->identify

(no version information, might be only in CVS)

hw_api->identify -- Log into Hyperwave Server

Description

object **identify** (array parameter)

Logs into the Hyperwave Server. The parameter array must contain the elements 'username' und 'password'.

The return value will be an object of type **HW_API_Error** if identification failed or TRUE if it was successful.

hw_api->info

(no version information, might be only in CVS)

hw_api->info -- Returns information about server configuration

Description

object **info** (array parameter)

See also **hwapi_dcstat()**, **hwapi_dbstat()**, **hwapi_ftstat()**, **hwapi_hwstat()**.

hw_api->insert

(no version information, might be only in CVS)

hw_api->insert -- Inserts a new object

Description

object **insert** (array parameter)

Insert a new object. The object type can be user, group, document or anchor. Depending on the type other object attributes has to be set. The parameter array contains the required elements 'object' and 'content' (if the object is a document) and the optional parameters 'parameters', 'mode' and 'attributeSelector'. The 'object' must contain all attributes of the object. 'parameters' is an object as well holding futher attributes like the destination (attribute key is 'Parent'). 'content' is the content of the document. 'mode' can be a combination of the following flags:

HW_API_INSERT_NORMAL

The object in inserted into the server.

HW_API_INSERT_FORCE-VERSION-CONTROL

HW_API_INSERT_AUTOMATIC-CHECKOUT

HW_API_INSERT_PLAIN

HW_API_INSERT_KEEP_TIME_MODIFIED

HW_API_INSERT_DELAY_INDEXING

See also **hwapi_replace()**.

hw_api->insertanchor

(no version information, might be only in CVS)

hw_api->insertanchor -- Inserts a new object of type anchor

Description

object **insertanchor** (array parameter)

This function is a shortcut for **hwapi_insert()**. It inserts an object of type anchor and sets some of the attributes required for an anchor. The parameter array contains the required elements 'object' and 'documentIdentifier' and the optional elements 'destinationIdentifier', 'parameter', 'hint' and 'attributeSelector'. The 'documentIdentifier' specifies the document where the anchor shall be inserted. The target of the anchor is set in 'destinationIdentifier' if it already exists. If the target does not exists the element 'hint' has to be set to the name of object which is supposed to be inserted later. Once it is inserted the anchor target is resolved automatically.

See also **hwapi_insertdocument()**, **hwapi_insertcollection()**, **hwapi_insert()**.

hw_api->insertcollection

(no version information, might be only in CVS)

hw_api->insertcollection -- Inserts a new object of type collection

Description

object **insertcollection** (array parameter)

This function is a shortcut for **hwapi_insert()**. It inserts an object of type collection and sets some of the attributes required for a collection. The parameter array contains the required elements 'object' and 'parentIdentifier' and the optional elements 'parameter' and 'attributeSelector'. See **hwapi_insert()** for the meaning of each element.

See also **hwapi_insertdocument()**, **hwapi_insertanchor()**, **hwapi_insert()**.

hw_api->insertdocument

(no version information, might be only in CVS)

hw_api->insertdocument -- Inserts a new object of type document

Description

object **insertdocument** (array parameter)

This function is a shortcut for **hwapi_insert()**. It inserts an object with content and sets some of the attributes required for a document. The parameter array contains the required elements 'object', 'parentIdentifier' and 'content' and the optional elements 'mode', 'parameter' and 'attributeSelector'. See **hwapi_insert()** for the meaning of each element.

See also **hwapi_insert()** **hwapi_insertanchor()**, **hwapi_insertcollection()**.

hw_api->link

(no version information, might be only in CVS)

hw_api->link -- Creates a link to an object

Description

object **link** (array parameter)

Creates a link to an object. Accessing this link is like accessing the object to links points to. The parameter array contains the required elements 'objectIdentifier' and 'destinationParentIdentifier'. 'destinationParentIdentifier' is the target collection.

The function returns **TRUE** on success or an error object.

See also **hwapi_copy()**.

hw_api->lock

(no version information, might be only in CVS)

hw_api->lock -- Locks an object

Description

object **lock** (array parameter)

Locks an object for exclusive editing by the user calling this function. The object can be only unlocked by this user or the sytem user. The parameter array contains the required element 'objectIdentifier' and the optional parameters 'mode' and 'objectquery'. 'mode' determines how an object is locked. HW_API_LOCK_NORMAL means, an object is locked until it is unlocked. HW_API_LOCK_RECURSIVE is only valid for collection and locks all objects within th collection und possible subcollections. HW_API_LOCK_SESSION means, an object is locked only as long as the session is valid.

See also **hwapi_unlock()**.

hw_api->move

(no version information, might be only in CVS)

hw_api->move -- Moves object between collections

Description

object **move** (array parameter)

See also **hw_objrec2array()**.

hw_api_content

(no version information, might be only in CVS)

hw_api_content -- Create new instance of class hw_api_content

Description

string **content** (string content, string mimetype)

Creates a new content object from the string *content*. The mimetype is set to *mimetype*.

hw_api_object->assign

(no version information, might be only in CVS)

hw_api_object->assign -- Clones object

Description

object **assign** (array parameter)

Clones the attributes of an object.

hw_api_object->attreditable

(no version information, might be only in CVS)

hw_api_object->attreditable -- Checks whether an attribute is editable

Description

bool **attreditable** (array parameter)

hw_api_object->count

(no version information, might be only in CVS)

hw_api_object->count -- Returns number of attributes

Description

int **count** (array parameter)

hw_api_object->insert

(no version information, might be only in CVS)

hw_api_object->insert -- Inserts new attribute

Description

bool **insert** (object attribute)

Adds an attribute to the object. Returns **TRUE** on success and otherwise **FALSE**.

See also **hwapi_object_remove()**.

hw_api_object

(no version information, might be only in CVS)

hw_api_object -- Creates a new instance of class hw_api_object

Description

object **hw_api_object** (array parameter)

See also **hwapi_lock()**.

hw_api_object->remove

(no version information, might be only in CVS)

hw_api_object->remove -- Removes attribute

Description

bool **remove** (string name)

Removes the attribute with the given name. Returns **TRUE** on success and otherwise **FALSE**.

See also **hwapi_object_insert()**.

hw_api_object->title

(no version information, might be only in CVS)

hw_api_object->title -- Returns the title attribute

Description

string **title** (array parameter)

hw_api_object->value

(no version information, might be only in CVS)

hw_api_object->value -- Returns value of attribute

Description

string **value** (string name)

Returns the value of the attribute with the given name or **FALSE** if an error occured.

hw_api->object

(no version information, might be only in CVS)

hw_api->object -- Retrieve attribute information

Description

object **hw_api->object** (array parameter)

This function retrieves the attribute information of an object of any version. It will not return the document content. The parameter array contains the required elements 'objectIdentifier' and the optional elements 'attributeSelector' and 'version'.

The returned object is an instance of class **HW_API_Object** on success or **HW_API_Error** if an error occured.

This simple example retrieves an object and checks for errors.

Example 1. Retrieve an object

```
<?php
function handle_error($error) {
  $reason = $error->reason(0);
  echo "Type: <B>";
  switch($reason->type()) {
    case 0:
      echo "Error";
      break;
    case 1:
      echo "Warning";
      break;
    case 2:
      echo "Message";
      break;
  }
  echo "</B><BR>\n";
  echo "Description: ".$reason->description("en")."<BR>\n";
}

function list_attr($obj) {
  echo "<TABLE>\n";
  $count = $obj->count();
  for($i=0; $i<$count; $i++) {
    $attr = $obj->attribute($i);
    printf("  <TR><TD ALIGN=right bgcolor=#c0c0c0><B>%s</B></TD><TD bgcolor=#F0F0F0>%s</TD>\n",
             $attr->key(), $attr->value());
  }
  echo "</TABLE>\n";
}

$hwapi = hwapi_hgcsp($g_config[HOSTNAME]);
$parms = array("objectIdentifier"=>"rootcollection, "attributeSelector"=>array("Title", "Name", "DocumentType"));
$root = $hwapi->object($parms);
if(get_class($root) == "HW_API_Error") {
  handle_error($root);
  exit;
}
list_attr($root);
?>
```

See also **hwapi_content()**.

hw_api->objectbyanchor

(no version information, might be only in CVS)

hw_api->objectbyanchor -- Returns the object an anchor belongs to

Description

object **objectbyanchor** (array parameter)

This function retrieves an object the specified anchor belongs to. The parameter array contains the required element

'objectIdentifier' and the optional element 'attributeSelector'.

See also **hwapi_dstofsrcanchor()**, **hwapi_srcanchors()**, **hwapi_dstanchors()**.

hw_api->parents

(no version information, might be only in CVS)

hw_api->parents -- Returns parents of an object

Description

array **parents** (array parameter)

Retrieves the parents of an object. The parents can be further filtered by specifying an object query. The parameter array contains the required elements 'objectidentifier' and the optional elements 'attributeselector' and 'objectquery'.

The return value is an array of objects of type **HW_API_Object** or **HW_API_Error**.

See also **hwapi_children()**.

hw_api_reason->description

(no version information, might be only in CVS)

hw_api_reason->description -- Returns description of reason

Description

string **description** (void)

Returns the description of a reason

hw_api_reason->type

(no version information, might be only in CVS)

hw_api_reason->type -- Returns type of reason

Description

object **type** (void)

Returns the type of a reason.

hw_api->remove

(no version information, might be only in CVS)

hw_api->remove -- Delete an object

Description

object **remove** (array parameter)

This function removes an object from the specified parent. Collections will be removed recursively. You can pass an optional object query to remove only those objects which match the query. An object will be deleted physically if it is the last instance. The parameter array contains the required elements 'objectidentifier' and 'parentidentifier'. If you want to remove a user or group 'parentidentifier' can be skiped. The optional parameter 'mode' determines how the deletion is performed. In normal mode the object will not be removed physically until all instances are removed. In physical mode all instances of the object will

be deleted imediately. In removelinks mode all references to and from the objects will be deleted as well. In nonrecursive the deletion is not performed recursive. Removing a collection which is not empty will cause an error.

See also **hwapi_move()**.

hw_api->replace

(no version information, might be only in CVS)

hw_api->replace -- Replaces an object

Description

object **replace** (array parameter)

Replaces the attributes and the content of an object The parameter array contains the required elements 'objectIdentifier' and 'object' and the optional parameters 'content', 'parameters', 'mode' and 'attributeSelector'. 'objectIdentifier' contains the object to be replaced. 'object' contains the new object. 'content' contains the new content. 'parameters' contain extra information for HTML documents. HTML_Language is the letter abbreviation of the language of the title. HTML_Base sets the base attribute of the HTML document. 'mode' can be a combination of the following flags:

HW_API_REPLACE_NORMAL

The object on the server is replace with the object passed.

HW_API_REPLACE_FORCE_VERSION_CONTROL

HW_API_REPLACE_AUTOMATIC_CHECKOUT

HW_API_REPLACE_AUTOMATIC_CHECKIN

HW_API_REPLACE_PLAIN

HW_API_REPLACE_REVERT_IF_NOT_CHANGED

HW_API_REPLACE_KEEP_TIME_MODIFIED

See also **hwapi_insert()**.

hw_api->setcommitedversion

(no version information, might be only in CVS)

hw_api->setcommitedversion -- Commits version other than last version

Description

object **setcommitedversion** (array parameter)

Commits a version of a docuemnt. The commited version is the one which is visible to users with read access. By default the last version is the commited version.

See also **hwapi_checkin()**, **hwapi_checkout()**, **hwapi_revert()**.

hw_api->srcanchors

(no version information, might be only in CVS)

hw_api->srcanchors -- Returns a list of all source anchors

Description

object **srcanchors** (array parameter)

Retrieves all source anchors of an object. The parameter array contains the required element 'objectIdentifier' and the optional elements 'attributeSelector' and 'objectQuery'.

See also **hwapi_dstanchors()**.

hw_api->srcsofdst

(no version information, might be only in CVS)

hw_api->srcsofdst -- Returns source of a destination object

Description

object **srcsofdst** (array parameter)

Retrieves all the source anchors pointing to the specified destination. The destination object can either be a destination anchor or a whole document. The parameters array contains the required element 'objectIdentifier' and the optional element 'attributeSelector' and 'objectQuery'. The function returns an array of objects or an error.

See also **hwapi_dstofsrcanchor()**.

hw_api->unlock

(no version information, might be only in CVS)

hw_api->unlock -- Unlocks a locked object

Description

object **unlock** (array parameter)

Unlocks a locked object. Only the user who has locked the object and the system user may unlock an object. The parameter array contains the required element 'objectIdentifier' and the optional parameters 'mode' and 'objectquery'. The meaning of 'mode' is the same as in function **hwapi_lock()**.

Returns **TRUE** on success or an object of class HW_API_Error.

See also **hwapi_lock()**.

hw_api->user

(no version information, might be only in CVS)

hw_api->user -- Returns the own user object

Description

object **user** (array parameter)

See also **hwapi_userlist()**.

hw_api->userlist

(no version information, might be only in CVS)

hw_api->userlist -- Returns a list of all logged in users

Description

object **userlist** (array parameter)

See also **hwapi_user()**.

XL. iconv functions

Introduction

This module contains an interface to the iconv library functions. The iconv library functions convert strings between various character sets encodings. The supported character sets depend on the iconv() implementation on your system. Note that the iconv() function on some systems may not work as well as you expect. In this case, you should install the libiconv library.

Requirements

Your systems standard C library must provide the iconv() function or you must have libiconv installed on your system. The libiconv library is available from http://www.gnu.org/software/libiconv/.

Installation

To be able to use the functions defined in this module you must compile your PHP interpreter using the configure line `--with-iconv[=DIR]`.

> **Note to Win32 Users:** In order to enable this module on a Windows environment, you must copy *iconv-1.3.dll* from the DLL folder of the PHP/Win32 binary package to the SYSTEM32 folder of your windows machine. (Ex: C:\WINNT\SYSTEM32 or C:\WINDOWS\SYSTEM32). Starting with PHP 4.2.1 the name changed to *iconv.dll*

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. Iconv configuration options

Name	Default	Changeable
iconv.input_encoding	ICONV_INPUT_ENCODING	PHP_INI_ALL
iconv.output_encoding	ICONV_OUTPUT_ENCODING	PHP_INI_ALL
iconv.internal_encoding	ICONV_INTERNAL_ENCODING	PHP_INI_ALL

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Resource Types

This extension has no resource types defined.

Predefined Constants

Since PHP 4.3.0 it is possible to identify at runtime which iconv implementation is adopted by this extension.

Table 2. iconv constants

constant	type	description
ICONV_IMPL	string	The implementation name
ICONV_VERSION	string	The implementation version

Note: Writing implementation-dependent scripts with these constants should be discouraged.

See Also

See also the GNU Recode functions.

Table of Contents

iconv_get_encoding

(PHP 4 >= 4.0.5)

iconv_get_encoding -- Get current setting for character encoding conversion

Description

array **iconv_get_encoding** ([string type])

It returns the current settings of **ob_iconv_handler()** as array or **FALSE** on failure. The value of the optinal *type* can be:

all
input_encoding
output_encoding
internal_encoding

If *type* is omitted or not 'all' **iconv_get_encoding()** returns the current settings of **ob_iconv_handler()** as string.

Example 1. iconv_get_encoding() example:

```
<pre>
<?php
iconv_set_encoding("internal_encoding", "UTF-8");
iconv_set_encoding("output_encoding", "ISO-8859-1");
var_dump(iconv_get_encoding('all'));
?>
</pre>
```

The printout of the above program will be:

```
Array
(
    [input_encoding] => ISO-8859-1
    [output_encoding] => ISO-8859-1
    [internal_encoding] => UTF-8
)
```

See also: **iconv_set_encoding()** and **ob_iconv_handler()**.

iconv_set_encoding

(PHP 4 >= 4.0.5)

iconv_set_encoding -- Set current setting for character encoding conversion

Description

bool **iconv_set_encoding** (string type, string charset)

It changes the value of *type* to *charset*. Returns **TRUE** on success or **FALSE** on failure.

The value of *type* can be:

input_encoding
output_encoding
internal_encoding

Example 1. iconv_set_encoding() example:

```
iconv_set_encoding("internal_encoding", "UTF-8");
iconv_set_encoding("output_encoding", "ISO-8859-1");
```

See also: **iconv_get_encoding()** and **ob_iconv_handler()**.

iconv

(PHP 4 >= 4.0.5)

iconv -- Convert string to requested character encoding

Description

string **iconv** (string in_charset, string out_charset, string str)

It converts the string *str* encoded in *in_charset* to the string encoded in *out_charset*. It returns the converted string or **FALSE**, if it fails.

Example 1. iconv() example:

```
echo iconv("ISO-8859-1","UTF-8","This is a test.");
```

ob_iconv_handler

(PHP 4 >= 4.0.5)

ob_iconv_handler -- Convert character encoding as output buffer handler

Description

array **ob_iconv_handler** (string contents, int status)

It converts the string encoded in *internal_encoding* to *output_encoding*.

internal_encoding and *output_encoding* should be defined by **iconv_set_encoding()** or in the configuration file `php.ini`.

Example 1. ob_iconv_handler() example:

```
ob_start("ob_iconv_handler"); // start output buffering
```

See also: **iconv_get_encoding()**, **iconv_set_encoding()**, and output-control functions.

XLI. Image functions

Introduction

PHP is not limited to creating just HTML output. It can also be used to create and manipulate image files in a variety of different image formats, including gif, png, jpg, wbmp, and xpm. Even more convenient, PHP can output image streams directly to a

browser. You will need to compile PHP with the GD library of image functions for this to work. GD and PHP may also require other libraries, depending on which image formats you want to work with.

You can use the image functions in PHP to get the size of JPEG, GIF, PNG, SWF, TIFF and JPEG2000 images.

> **Note:** Read requirements section about how to expand image capabilities to read, write and modify images and to read meta data of pictures taken by digital cameras.

Requirements

If you have the GD library (available at http://www.boutell.com/gd/) you will also be able to create and manipulate images.

The format of images you are able to manipulate depend on the version of GD you install, and any other libraries GD might need to access those image formats. Versions of GD older than gd-1.6 support GIF format images, and do not support PNG, where versions greater than gd-1.6 support PNG, not GIF.

> **Note:** Since PHP 4.3 there is a bundled version of the GD lib. This bundled version has some additional features like alpha blending, and should be used in preference to the external library since it's codebase is better maintained and more stable.

You may wish to enhance GD to handle more image formats.

Table 1. Supported image formats

Image format	Library to download	Notes
`gif`		Only supported in GD versions older than gd-1.6. *Read-only* GIF support is available with PHP 4.3.0 and the bundled GD-library.
`jpeg-6b`	ftp://ftp.uu.net/graphics/jpeg/	
`png`	http://www.libpng.org/pub/png/libpng.html	Only supported in GD versions greater than gd-1.6.
`xpm`	ftp://metalab.unc.edu/pub/Linux/libs/X/!INDEX.html	It's likely you have this library already available, if your system has an installed X-Environment.

You may wish to enhance GD to deal with different fonts. The following font libraries are supported:

Table 2. Supported font libraries

Font library	Download	Notes
`FreeType 1.x`	http://www.freetype.org/	
`FreeType 2`	http://www.freetype.org/	
`T1lib`	ftp://sunsite.unc.edu/pub/Linux/libs/graphics/)	Support for Type 1 fonts.

If you have PHP compiled with `--enable-exif` you are able to work with information stored in headers of JPEG and TIFF images. This way you can read meta data generated by digital cameras as mentioned above. These functions does not require the GD library.

> **Note:** PHP does not require any additional library for exif the module.

Installation

To enable GD-support configure PHP `--with-gd[=DIR]`, where DIR is the GD base install directory. To use the recommended bundled version of the GD library configure `--with-gd`. In Windows you'll include `php_gd2.dll` as an extension in `php.ini`. There is also `php_gd.dll` for GD1 but it's not preferred.

Enhance the capabilities of GD to handle more image formats by specifying the `--with-XXXX` configure switch to your PHP configure line.

Table 3. Supported image formats

Image Format	Configure Switch
`jpeg-6b`	To enable support for jpeg-6b add `--with-jpeg-dir=DIR`.

Image Format	Configure Switch
png	To enable support for png add `--with-png-dir=DIR`. Note, libpng requires the zlib library, therefore add `--with-zlib-dir[=DIR]` to your configure line.
xpm	To enable support for xpm add `--with-xpm-dir=DIR`. If configure is not able to find the required libraries, you may add the path to your X11 libraries.

Enhance the capabilities of GD to deal with different fonts by specifying the `--with-XXXX` configure switch to your PHP configure line.

Table 4. Supported font libraries

Font library	Configure Switch
`FreeType 1.x`	To enable support for FreeType 1.x add `--with-ttf[=DIR]`.
`FreeType 2`	To enable support for FreeType 2 add `--with-freetype-dir=DIR`.
`T1lib`	To enable support for T1lib (Type 1 fonts) add `--with-t1lib[=DIR]`.
`Native TrueType string function`	To enable support for native TrueType string function add `--enable-gd-native-ttf`.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Exif supports automatically conversion for Unicode and JIS character encodings of user comments when module mbstring is available. This is done by first decoding the comment using the specified characterset. The result is then encoded with another characterset which should match your `HTTP` output.

Table 5. Exif configuration options

Name	Default	Changeable
exif.encode_unicode	"ISO-8859-15"	PHP_INI_ALL
exif.decode_unicode_motorola	"UCS-2BE"	PHP_INI_ALL
exif.decode_unicode_intel	"UCS-2LE"	PHP_INI_ALL
exif.encode_jis	""	PHP_INI_ALL
exif.decode_jis_motorola	"JIS"	PHP_INI_ALL
exif.decode_jis_intel	"JIS"	PHP_INI_ALL

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Here is a short explanation of the configuration directives.

`exif.encode_unicode` **string**

`exif.encode_unicode` defines the characterset UNICODE user comments are handled. This defaults to ISO-8859-15 which should work for most non asian countries. The setting can be empty or must be an encoding supported by mbstring. If it is empty the current internal encoding of mbstring is used.

`exif.decode_unicode_motorola` **string**

`exif.decode_unicode_motorola` defines the image internal characterset for Unicode encoded user comments if image is in motorola byte order (big-endian). This setting cannot be empty but you can specify a list of encodings supported by mbstring. The default is UCS-2BE.

`exif.decode_unicode_intel` **string**

`exif.decode_unicode_intel` defines the image internal characterset for Unicode encoded user comments if image is in intel byte order (little-endian). This setting cannot be empty but you can specify a list of encodings supported by mbstring. The default is UCS-2LE.

`exif.encode_jis` **string**

`exif.encode_jis` defines the characterset JIS user comments are handled. This defaults to an empty value which forces the functions to use the current internal encoding of mbstring.

`exif.decode_jis_motorola` **string**

`exif.decode_jis_motorola` defines the image internal characterset for JIS encoded user comments if image is in motorola byte order (big-endian). This setting cannot be empty but you can specify a list of encodings supported by mbstring. The default is JIS.

`exif.decode_jis_intel` **string**

`exif.decode_jis_intel` defines the image internal characterset for JIS encoded user comments if image is in intel byte order (little-endian). This setting cannot be empty but you can specify a list of encodings supported by mbstring. The default is JIS.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

`IMG_GIF` (integer)

`IMG_JPG` (integer)

`IMG_JPEG` (integer)

`IMG_PNG` (integer)

`IMG_WBMP` (integer)

`IMG_XPM` (integer)

`IMG_COLOR_TILED` (integer)

`IMG_COLOR_STYLED` (integer)

`IMG_COLOR_BRUSHED` (integer)

`IMG_COLOR_STYLEDBRUSHED` (integer)

`IMG_COLOR_TRANSPARENT` (integer)

`IMG_ARC_ROUNDED` (integer)

`IMG_ARC_PIE` (integer)

`IMG_ARC_CHORD` (integer)

`IMG_ARC_NOFILL` (integer)

`IMG_ARC_EDGED` (integer)

`IMAGETYPE_GIF` (integer)

`IMAGETYPE_JPEG` (integer)

`IMAGETYPE_PNG` (integer)

`IMAGETYPE_SWF` (integer)

`IMAGETYPE_PSD` (integer)

`IMAGETYPE_BMP` (integer)

`IMAGETYPE_TIFF_II` (integer)

`IMAGETYPE_TIFF_MM` (integer)

`IMAGETYPE_JPC` (integer)

IMAGETYPE_JP2 (integer)

IMAGETYPE_JPX (integer)

IMAGETYPE_SWC (integer)

Examples

Example 1. PNG creation with PHP

```
<?php
    header("Content-type: image/png");
    $string = $_GET['text'];
    $im     = imagecreatefrompng("images/button1.png");
    $orange = imagecolorallocate($im, 220, 210, 60);
    $px     = (imagesx($im) - 7.5 * strlen($string)) / 2;
    imagestring($im, 3, $px, 9, $string, $orange);
    imagepng($im);
    imagedestroy($im);
?>
```

This example would be called from a page with a tag like: <img src="button.php?text">. The above button.php script then takes this "text" string and overlays it on top of a base image which in this case is "images/button1.png" and outputs the resulting image. This is a very convenient way to avoid having to draw new button images every time you want to change the text of a button. With this method they are dynamically generated.

Table of Contents

imagecreatetruecolor -- Create a new true color image
imagedashedline -- Draw a dashed line
imagedestroy -- Destroy an image
imageellipse -- Draw an ellipse
imagefill -- Flood fill
imagefilledarc -- Draw a partial ellipse and fill it
imagefilledellipse -- Draw a filled ellipse
imagefilledpolygon -- Draw a filled polygon
imagefilledrectangle -- Draw a filled rectangle
imagefilltoborder -- Flood fill to specific color
imagefontheight -- Get font height
imagefontwidth -- Get font width
imageftbbox -- Give the bounding box of a text using fonts via freetype2
imagefttext -- Write text to the image using fonts using FreeType 2
imagegammacorrect -- Apply a gamma correction to a GD image
imagegd2 -- Output GD2 image to browser or file
imagegd -- Output GD image to browser or file
imagegif -- Output image to browser or file
imageinterlace -- Enable or disable interlace
imagejpeg -- Output image to browser or file
imageline -- Draw a line
imageloadfont -- Load a new font
imagepalettecopy -- Copy the palette from one image to another
imagepng -- Output a PNG image to either the browser or a file
imagepolygon -- Draw a polygon
imagepsbbox -- Give the bounding box of a text rectangle using PostScript Type1 fonts
imagepscopyfont -- Make a copy of an already loaded font for further modification
imagepsencodefont -- Change the character encoding vector of a font
imagepsextendfont -- Extend or condense a font
imagepsfreefont -- Free memory used by a PostScript Type 1 font
imagepsloadfont -- Load a PostScript Type 1 font from file
imagepsslantfont -- Slant a font
imagepstext -- To draw a text string over an image using PostScript Type1 fonts
imagerectangle -- Draw a rectangle
imagerotate -- Rotate an image with a given angle
imagesetbrush -- Set the brush image for line drawing
imagesetpixel -- Set a single pixel
imagesetstyle -- Set the style for line drawing
imagesetthickness -- Set the thickness for line drawing
imagesettile -- Set the tile image for filling
imagestring -- Draw a string horizontally
imagestringup -- Draw a string vertically
imagesx -- Get image width
imagesy -- Get image height
imagetruecolortopalette -- Convert a true color image to a palette image
imagettfbbox -- Give the bounding box of a text using TrueType fonts
imagettftext -- Write text to the image using TrueType fonts
imagetypes -- Return the image types supported by this PHP build
imagewbmp -- Output image to browser or file
iptcembed -- Embed binary IPTC data into a JPEG image
iptcparse -- Parse a binary IPTC http://www.iptc.org/ block into single tags.
jpeg2wbmp -- Convert JPEG image file to WBMP image file
png2wbmp -- Convert PNG image file to WBMP image file
read_exif_data -- Reads header information stored in TIFF and JPEG images

exif_imagetype

(PHP 4 >= 4.3.0)

exif_imagetype -- Determine the type of an image

Description

int|false **exif_imagetype** (string filename)

exif_imagetype() reads the first bytes of an image and checks its signature. When a correct signature is found a constant will be returned otherwise the return value is FALSE. The return value is the same value that **getimagesize()** returns in index 2 but this

function is much faster.

The following constants are defined: 1 = IMAGETYPE_GIF, 2 = IMAGETYPE_JPEG, 3 = IMAGETYPE_PNG, 4 = IMAGETYPE_SWF, 5 = IMAGETYPE_PSD, 6 = IMAGETYPE_BMP, 7 = IMAGETYPE_TIFF_II (intel byte order), 8 = IMAGETYPE_TIFF_MM (motorola byte order), 9 = IMAGETYPE_JPC, 10 = IMAGETYPE_JP2, 11 = IMAGETYPE_JPX, 12 = IMAGETYPE_SWC.

This function can be used to avoid calls to other exif functions with unsupported file types or in conjunction with $_SERVER['HTTP_ACCEPT'] to check whether or not the viewer is able to see a specific image in his browser.

> **Note:** This function is only available in PHP 4 compiled using `--enable-exif`.
>
> This function does not require the GD image library.
>
> See also **getimagesize()**.

exif_read_data

(PHP 4 >= 4.2.0)

exif_read_data -- Read the EXIF headers from JPEG or TIFF. This way you can read meta data generated by digital cameras.

Description

array **exif_read_data** (string filename [, string sections [, bool arrays [, bool thumbnail]]])

The **exif_read_data()** function reads the EXIF headers from a JPEG or TIFF image file. It returns an associative array where the indexes are the header names and the values are the values associated with those headers. If no data can be returned the result is **FALSE**.

filename is the name of the file to read. This cannot be a url.

sections a comma separated list of sections that need to be present in file to produce a result array.

FILE	FileName, FileSize, FileDateTime, SectionsFound
COMPUTED	html, Width, Height, IsColor and some more if available.
ANY_TAG	Any information that has a Tag e.g. IFDo, EXIF, ...
IFDo	All tagged data of IFDo. In normal imagefiles this contains image size and so forth.
THUMBNAIL	A file is supposed to contain a thumbnail if it has a second IFD. All tagged information about the embedded thumbnail is stored in this section.
COMMENT	Comment headers of JPEG images.
EXIF	The EXIF section is a sub section of IFDo. It contains more detailed information about an image. Most of these entries are digital camera related.

arrays specifies whether or not each section becomes an array. The sections *FILE*, *COMPUTED* and *THUMBNAIL* allways become arrays as they may contain values whose names are conflict with other sections.

thumbnail whether or not to read the thumbnail itself and not only its tagged data.

> **Note:** Exif headers tend to be present in JPEG/TIFF images generated by digital cameras, but unfortunately each digital camera maker has a different idea of how to actually tag their images, so you can't always rely on a specific Exif header being present.

Example 1. exif_read_data() example

```
<?php
echo "test1.jpg:<br />\n";
$exif = exif_read_data ('tests/test1.jpg','IFD0');
echo $exif===false ? "No header data found.<br />\n" : "Image contains headers<br />";
$exif = exif_read_data ('tests/test2.jpg',0,true);
echo "test2.jpg:<br />\n";
foreach($exif as $key=>$section) {
    foreach($section as $name=>$val) {
        echo "$key.$name: $val<br />\n";
    }
}?>
```

The first call fails because the image has no header information.

```
test1.jpg:
```

```
No header data found.
test2.jpg:
FILE.FileName: test2.jpg
FILE.FileDateTime: 1017666176
FILE.FileSize: 1240
FILE.FileType: 2
FILE.SectionsFound: ANY_TAG, IFD0, THUMBNAIL, COMMENT
COMPUTED.html: width="1" height="1"
COMPUTED.Height: 1
COMPUTED.Width: 1
COMPUTED.IsColor: 1
COMPUTED.ByteOrderMotorola: 1
COMPUTED.UserComment: Exif test image.
COMPUTED.UserCommentEncoding: ASCII
COMPUTED.Copyright: Photo (c) M.Boerger, Edited by M.Boerger.
COMPUTED.Copyright.Photographer: Photo (c) M.Boerger
COMPUTED.Copyright.Editor: Edited by M.Boerger.
IFD0.Copyright: Photo (c) M.Boerger
IFD0.UserComment: ASCII
THUMBNAIL.JPEGInterchangeFormat: 134
THUMBNAIL.JPEGInterchangeFormatLength: 523
COMMENT.0: Comment #1.
COMMENT.1: Comment #2.
COMMENT.2: Comment #3end

THUMBNAIL.JPEGInterchangeFormat: 134
THUMBNAIL.Thumbnail.Height: 1
THUMBNAIL.Thumbnail.Height: 1
```

Note: If the image contains any IFDo data then COMPUTED contains the entry ByteOrderMotorola which is o for little-endian (intel) and 1 for big-endian (motorola) byte order. This was added in PHP 4.3.

When an Exif header contains a Copyright note this itself can contain two values. As the solution is inconsitent in the Exif 2.10 standard the COMPUTED section will return both entries *Copyright.Photographer* and *Copyright.Editor* while the IFDo sections contains the byte array with the NULL character that splits both entries. Or just the first entry if the datatype was wrong (normal behaviour of Exif). The COMPUTED will contain also an entry *Copyright* Which is either the original copyright string or it is a comma separated list of photo and editor copyright.

Note: The tag UserComment has the same problem as the Copyright tag. It can store two values first the encoding used and second the value itself. If so the IFD section only contains the encoding or a byte array. The COMPUTED section will store both in the entries *UserCommentEncoding* and *UserComment*. The entry *UserComment* is available in both cases so it should be used in preference to the value in IFDo section.

If the user comment uses Unicode or JIS encoding and the module mbstring is available this encoding will automatically changed according to the exif ini settings. This was added in PHP 4.3.

Note: Height and Width are computed the same way **getimagesize()** does so their values must not be part of any header returned. Also html is a height/width text string to be used inside normal HTML.

Note: Starting from PHP 4.3 the function can read all embedded IFD data including arrays (returned as such). Also the size of an embedded thumbnail is returned in *THUMBNAIL* subarray and the function **exif_read_data()** can return thumbnails in TIFF format. Last but not least there is no longer a maximum legth for returned values (not until memory limit is reached).

Note: This function is only available in PHP 4 compiled using `--enable-exif`. Its functionality and behaviour has changed in PHP 4.2. Earlier versions are very unstable.

Since PHP 4.3 user comment can automatically change encoding if PHP 4 was compiled using `--enable-mbstring`.

This function does not require the GD image library.

See also **exif_thumbnail()** and **getimagesize()**.

exif_thumbnail

(PHP 4 >= 4.2.0)

exif_thumbnail -- Retrieve the embedded thumbnail of a TIFF or JPEG image

Description

string **exif_thumbnail** (string filename [, int &width [, int &height [, int &imagetype]]])

exif_thumbnail() reads the embedded thumbnail of a TIFF or JPEG image. If the image contains no thumbnail **FALSE** will be returned.

The parameters *width*, *height* and *imagetype* are available since PHP 4.3 and return the size of the thumbnail as well as its type. It is possible that **exif_thumbnail()** cannot create an image but determine its size. In this case the return value is **FALSE** but *width* and *height* are set.

If you want to deliver thumbnails through this function you should send the mimetype information using **header()** function. The following example demonstrates this:

Example 1. exif_thumbnail() example

```
<?php
if (array_key_exists('file',$_REQUEST)) {
        $image = exif_thumbnail($_REQUEST['file'], $width, $height, $type);
} else {
    $image = false;
}
if ($image!==false) {
    header("Content-type: ".image_type_to_mime_type($type));
    echo $image;
    exit;
} else {
    // no thumbnail available, handle the error here
    echo "No thumbnail available";
}
?>
```

Starting from version PHP 4.3 the function **exif_thumbnail()** can return thumbnails in TIFF format.

> **Note:** This function is only available in PHP 4 compiled using `--enable-exif`. Its functionality and behaviour has changed in PHP 4.2
>
> This function does not require the GD image library.
>
> See also **exif_read_data()** and **image_type_to_mime_type()**.

gd_info

(PHP 4 >= 4.3.0)

gd_info -- Retrieve information about the currently installed GD library

Description

array **gd_info** (void)

Returns an associative array describing the version and capabilities of the installed GD library.

Table 1. Elements of array returned by gd_info()

Attribute	Meaning
GD Version	**string** value describing the installed `libgd` version.
Freetype Support	**boolean** value. **TRUE** if Freetype Support is installed.
Freetype Linkage	**string** value describing the way in which Freetype was linked. Expected values are: 'with freetype', 'with TTF library', and 'with unknown library'. This element will only be defined if `Freetype Support` evaluated to **TRUE**.
T1Lib Support	**boolean** value. **TRUE** if `T1Lib` support is included.
GIF Read Support	**boolean** value. **TRUE** if support for *reading* `GIF` images is included.
GIF Create Support	**boolean** value. **TRUE** if support for *creating* `GIF` images is included.
JPG Support	**boolean** value. **TRUE** if `JPG` support is included.
PNG Support	**boolean** value. **TRUE** if `PNG` support is included.
WBMP Support	**boolean** value. **TRUE** if `WBMP` support is included.
XBM Support	**boolean** value. **TRUE** if `XBM` support is included.

Example 1. Using gd_info()

```
<?php
var_dump(gd_info());

/* Typical Output
   ==============

array(9) {
  ["GD Version"]=>
  string(24) "bundled (2.0 compatible)"
  ["FreeType Support"]=>
  bool(false)
  ["T1Lib Support"]=>
  bool(false)
  ["GIF Read Support"]=>
  bool(true)
  ["GIF Create Support"]=>
  bool(false)
  ["JPG Support"]=>
  bool(false)
  ["PNG Support"]=>
  bool(true)
  ["WBMP Support"]=>
  bool(true)
  ["XBM Support"]=>
  bool(false)
}

*/
?>
```

See also: **imagepng()**, **imagejpeg()**, **imagegif()**, **imagewbmp()**, and **imagetypes()**.

getimagesize

(PHP 3, PHP 4)

getimagesize -- Get the size of an image

Description

array **getimagesize** (string filename [, array imageinfo])

The **getimagesize()** function will determine the size of any GIF, JPG, PNG, SWF, SWC, PSD, TIFF, BMP or IFF image file and return the dimensions along with the file type and a height/width text string to be used inside a normal HTML `IMG` tag.

Returns an array with 4 elements. Index 0 contains the width of the image in pixels. Index 1 contains the height. Index 2 is a flag indicating the type of the image: 1 = GIF, 2 = JPG, 3 = PNG, 4 = SWF, 5 = PSD, 6 = BMP, 7 = TIFF(intel byte order), 8 = TIFF(motorola byte order), 9 = JPC, 10 = JP2, 11 = JPX, 12 = JB2, 13 = SWC, 14 = IFF. These values correspond to the IMAGETYPE constants that were added in PHP 4.3. Index 3 is a text string with the correct height="yyy" width="xxx" string that can be used directly in an IMG tag.

Example 1. getimagesize (file)

```
<?php
$size = getimagesize ("img/flag.jpg");
echo "<img src=\"img/flag.jpg\" {$size[3]}>";
?>
```

Example 2. getimagesize (URL)

```
<?php $size = getimagesize ("http://www.example.com/gifs/logo.gif"); ?>
```

With JPG images, two extra indexes are returned: `channels` and `bits`. `channels` will be 3 for RGB pictures and 4 for CMYK pictures. `bits` is the number of bits for each color.

Beginning with PHP 4.3, `bits` and `channels` are present for other image types, too. However, the presence of these values can be a bit confusing. As an example, GIF always uses 3 channels per pixel, but the number of bits per pixel cannot be calculated for an animated GIF with a global color table.

Some formats may contain no image or may contain multiple images. In these cases, **getimagesize()** might not be able to properly determine the image size. **getimagesize()** will return zero for width and height in these cases.

Beginning with PHP 4.3, **getimagesize()** also returns an additional parameter, `mime`, that corresponds with the MIME type of the image. This information can be used to deliver images with correct HTTP Content-type headers:

Example 3. getimagesize() and MIME types

```
<?php
$size = getimagesize ($filename);
$fp=fopen($filename, "rb");
if ($size && $fp) {
  header("Content-type: {$size['mime']}");
  fpassthru($fp);
  exit;
} else {
  // error
}
?>
```

If accessing the *filename* image is impossible, or if it isn't a valid picture, **getimagesize()** will return **FALSE** and generate a warning.

The optional *imageinfo* parameter allows you to extract some extended information from the image file. Currently, this will return the different JPG APP markers as an associative array. Some programs use these APP markers to embed text information in images. A very common one is to embed IPTC http://www.iptc.org/ information in the APP13 marker. You can use the **iptcparse()** function to parse the binary APP13 marker into something readable.

Example 4. getimagesize() returning IPTC

```
<?php
$size = getimagesize ("testimg.jpg",&$info);
if (isset ($info["APP13"])) {
    $iptc = iptcparse ($info["APP13"]);
    var_dump ($iptc);
}
?>
```

> **Note:** TIFF support was added in PHP 4.2.
>
> This function does not require the GD image library.
>
> See also **image_type_to_mime_type()**, **exif_imagetype()**, **exif_read_data()** and **exif_thumbnail()**.
>
> URL support was added in PHP 4.0.5.

image_type_to_mime_type

(PHP 4 >= 4.3.0)

image_type_to_mime_type -- Get Mime-Type for image-type returned by getimagesize, exif_read_data, exif_thumbnail, exif_imagetype

Description

string **image_type_to_mime_type** (int imagetype)

The **image_type_to_mime_type()** function will determine the Mime-Type for an IMAGETYPE constant.

Example 1. image_type_to_mime_type (file)

```
<?php
header ("Content-type: ".image_type_to_mime_type (IMAGETYPE_PNG));
?>
```

> **Note:** This function does not require the GD image library.
>
> See also **getimagesize()**, **exif_imagetype()**, **exif_read_data()** and **exif_thumbnail()**.

image2wbmp

(PHP 4 >= 4.0.5)

image2wbmp -- Output image to browser or file

Description

int **image2wbmp** (resource image [, string filename [, int threshold]])

image2wbmp() creates the WBMP file in filename from the image *image*. The *image* argument is the return from **imagecreate()**.

The filename argument is optional, and if left off, the raw image stream will be output directly. By sending an image/vnd.wap.wbmp content-type using **header()**, you can create a PHP script that outputs WBMP images directly.

Note: WBMP support is only available if PHP was compiled against GD-1.8 or later.

See also **imagewbmp()**.

imagealphablending

(PHP 4 >= 4.0.6)

imagealphablending -- Set the blending mode for an image

Description

int **imagealphablending** (resource image, bool blendmode)

imagealphablending() allows for two different modes of drawing on truecolor images. In blending mode, the alpha channel component of the color supplied to all drawing function, such as **imagesetpixel()** determines how much of the underlying color should be allowed to shine through. As a result, gd automatically blends the existing color at that point with the drawing color, and stores the result in the image. The resulting pixel is opaque. In non-blending mode, the drawing color is copied literally with its alpha channel information, replacing the destination pixel. Blending mode is not available when drawing on palette images. If *blendmode* is **TRUE**, then blending mode is enabled, otherwise disabled.

Note: This function was added in PHP 4.0.6 and requires GD 2.0.1

imagearc

(PHP 3, PHP 4)

imagearc -- Draw a partial ellipse

Description

int **imagearc** (resource image, int cx, int cy, int w, int h, int s, int e, int col)

imagearc() draws a partial ellipse centered at *cx*, *cy* (top left is 0, 0) in the image represented by *image*. *w* and *h* specifies the ellipse's width and height respectively while the start and end points are specified in degrees indicated by the *s* and *e* arguments. 0° is located at the three-o'clock position, and the arc is drawn counter-clockwise.

See also **imageellipse()**, **imagefilledellipse()**, and **imagefilledarc()**.

imagechar

(PHP 3, PHP 4)

imagechar -- Draw a character horizontally

Description

int **imagechar** (resource image, int font, int x, int y, string c, int col)

imagechar() draws the first character of *c* in the image identified by *id* with its upper-left at *x*,*y* (top left is 0, 0) with the color *col*. If *font* is 1, 2, 3, 4 or 5, a built-in font is used (with higher numbers corresponding to larger fonts).

See also **imageloadfont()**.

imagecharup

(PHP 3, PHP 4)

imagecharup -- Draw a character vertically

Description

int **imagecharup** (resource image, int font, int x, int y, string c, int col)

imagecharup() draws the character *c* vertically in the image identified by *image* at coordinates *x*, *y* (top left is 0, 0) with the color *col*. If *font* is 1, 2, 3, 4 or 5, a built-in font is used.

See also **imageloadfont()**.

imagecolorallocate

(PHP 3, PHP 4)

imagecolorallocate -- Allocate a color for an image

Description

int **imagecolorallocate** (resource image, int red, int green, int blue)

imagecolorallocate() returns a color identifier representing the color composed of the given RGB components. The *im* argument is the return from the **imagecreate()** function. *red*, *green* and *blue* are the values of the red, green and blue component of the requested color respectively. These parameters are integers between 0 and 255. **imagecolorallocate()** must be called to create each color that is to be used in the image represented by *image*.

```
$white = imagecolorallocate ($im, 255, 255, 255);
$black = imagecolorallocate ($im, 0, 0, 0);
```

Returns -1 if the allocation failed.

imagecolorallocatealpha

(no version information, might be only in CVS)

imagecolorallocatealpha -- Allocate a color for an image

Description

int **imagecolorallocatealpha** (resource image, int red, int green, int blue, int alpha)

imagecolorallocatealpha() behaves identically to **imagecolorallocate()** with the addition of the transparency parameter *alpha* which may have a value between `0` and `127`. `0` indicates completely opaque while `127` indicates completely transparent.

Returns **FALSE** if the allocation failed.

imagecolorat

(PHP 3, PHP 4)

imagecolorat -- Get the index of the color of a pixel

Description

int **imagecolorat** (resource image, int x, int y)

Returns the index of the color of the pixel at the specified location in the image specified by *image*.

If PHP is compiled against GD library 2.0 or higher and the image is a truecolor image, this function returns the RGB value of that pixel as integer. Use bitshifting and masking to access the distinct red, green and blue component values:

Example 1. Access distinct RGB values

```
<?php
$im = ImageCreateFromPng("rockym.png");
$rgb = ImageColorAt($im, 100, 100);
$r = ($rgb >> 16) & 0xFF;
$g = ($rgb >> 8) & 0xFF;
$b = $rgb & 0xFF;
?>
```

See also **imagecolorset()** and **imagecolorsforindex()**.

imagecolorclosest

(PHP 3, PHP 4)

imagecolorclosest -- Get the index of the closest color to the specified color

Description

int **imagecolorclosest** (resource image, int red, int green, int blue)

Returns the index of the color in the palette of the image which is "closest" to the specified RGB value.

The "distance" between the desired color and each color in the palette is calculated as if the RGB values represented points in three-dimensional space.

See also **imagecolorexact()**.

imagecolorclosestalpha

(PHP 4 >= 4.0.6)

imagecolorclosestalpha -- Get the index of the closest color to the specified color + alpha

Description

int **imagecolorclosestalpha** (resource image, int red, int green, int blue, int alpha)

Returns the index of the color in the palette of the image which is "closest" to the specified RGB value and *alpha* level.

See also **imagecolorexactalpha()**.

Note: This function was added in PHP 4.0.6 and requires GD 2.0.1

imagecolorclosesthwb

(PHP 4 >= 4.0.1)

imagecolorclosesthwb -- Get the index of the color which has the hue, white and blackness nearest to the given color

Description

int **imagecolorclosesthwb** (resource image, int red, int green, int blue)

Warning
This function is currently not documented; only the argument list is available.

imagecolordeallocate

(PHP 3>= 3.0.6, PHP 4)

imagecolordeallocate -- De-allocate a color for an image

Description

int **imagecolordeallocate** (resource image, int color)

The **imagecolordeallocate()** function de-allocates a color previously allocated with the **imagecolorallocate()** function.

```
$white = imagecolorallocate ($im, 255, 255, 255);
imagecolordeallocate ($im, $white);
```

imagecolorexact

(PHP 3, PHP 4)

imagecolorexact -- Get the index of the specified color

Description

int **imagecolorexact** (resource image, int red, int green, int blue)

Returns the index of the specified color in the palette of the image.

If the color does not exist in the image's palette, -1 is returned.

See also **imagecolorclosest()**.

imagecolorexactalpha

(PHP 4 >= 4.0.6)

imagecolorexactalpha -- Get the index of the specified color + alpha

Description

int **imagecolorexactalpha** (resource image, int red, int green, int blue, int alpha)

Returns the index of the specified color+alpha in the palette of the image.

If the color does not exist in the image's palette, -1 is returned.

See also **imagecolorclosestalpha()**.

Note: This function was added in PHP 4.0.6 and requires GD 2.0.1 or later

imagecolorresolve

(PHP 3>= 3.0.2, PHP 4)

imagecolorresolve -- Get the index of the specified color or its closest possible alternative

Description

int **imagecolorresolve** (resource image, int red, int green, int blue)

This function is guaranteed to return a color index for a requested color, either the exact color or the closest possible alternative.

See also **imagecolorclosest()**.

imagecolorresolvealpha

(PHP 4 >= 4.0.6)

imagecolorresolvealpha -- Get the index of the specified color + alpha or its closest possible alternative

Description

int **imagecolorresolvealpha** (resource image, int red, int green, int blue, int alpha)

This function is guaranteed to return a color index for a requested color, either the exact color or the closest possible alternative.

See also **imagecolorclosestalpha()**.

> **Note:** This function was added in PHP 4.0.6 and requires GD 2.0.1

imagecolorset

(PHP 3, PHP 4)

imagecolorset -- Set the color for the specified palette index

Description

bool **imagecolorset** (resource image, int index, int red, int green, int blue)

This sets the specified index in the palette to the specified color. This is useful for creating flood-fill-like effects in paletted images without the overhead of performing the actual flood-fill.

See also **imagecolorat()**.

imagecolorsforindex

(PHP 3, PHP 4)

imagecolorsforindex -- Get the colors for an index

Description

array **imagecolorsforindex** (resource image, int index)

This returns an associative array with red, green, and blue keys that contain the appropriate values for the specified color index.

See also **imagecolorat()** and **imagecolorexact()**.

imagecolorstotal

(PHP 3, PHP 4)

imagecolorstotal -- Find out the number of colors in an image's palette

Description

int **imagecolorstotal** (resource image)

This returns the number of colors in the specified image's palette.

See also **imagecolorat()** and **imagecolorsforindex()**.

imagecolortransparent

(PHP 3, PHP 4)

imagecolortransparent -- Define a color as transparent

Description

int **imagecolortransparent** (resource image [, int color])

imagecolortransparent() sets the transparent color in the *image* image to *color*. *image* is the image identifier returned by **imagecreate()** and *color* is a color identifier returned by **imagecolorallocate()**.

> **Note:** The transparent color is a property of the image, transparency is not a property of the color. Once you have a set a color to be the transparent color, any regions of the image in that color that were drawn previously will be transparent.

The identifier of the new (or current, if none is specified) transparent color is returned.

imagecopy

(PHP 3>= 3.0.6, PHP 4)

imagecopy -- Copy part of an image

Description

int **imagecopy** (resource dst_im, resource src_im, int dst_x, int dst_y, int src_x, int src_y, int src_w, int src_h)

Copy a part of *src_im* onto *dst_im* starting at the x,y coordinates *src_x*, *src_y* with a width of *src_w* and a height of *src_h*. The portion defined will be copied onto the x,y coordinates, *dst_x* and *dst_y*.

imagecopymerge

(PHP 4 >= 4.0.1)

imagecopymerge -- Copy and merge part of an image

Description

int **imagecopymerge** (resource dst_im, resource src_im, int dst_x, int dst_y, int src_x, int src_y, int src_w, int src_h, int pct)

Copy a part of *src_im* onto *dst_im* starting at the x,y coordinates *src_x*, *src_y* with a width of *src_w* and a height of *src_h*. The portion defined will be copied onto the x,y coordinates, *dst_x* and *dst_y*. The two images will be merged according to *pct* which can range from 0 to 100. When *pct* = 0, no action is taken, when 100 this function behaves identically to **imagecopy()**.

> **Note:** This function was added in PHP 4.0.6

imagecopymergegray

(PHP 4 >= 4.0.6)

imagecopymergegray -- Copy and merge part of an image with gray scale

Description

int **imagecopymergegray** (resource dst_im, resource src_im, int dst_x, int dst_y, int src_x, int src_y, int src_w, int src_h, int pct)

imagecopymergegray() copy a part of *src_im* onto *dst_im* starting at the x,y coordinates *src_x*, *src_y* with a width of *src_w* and a height of *src_h*. The portion defined will be copied onto the x,y coordinates, *dst_x* and *dst_y*. The two images will be merged according to *pct* which can range from 0 to 100. When *pct* = 0, no action is taken, when 100 this function behaves identically to **imagecopy()**.

This function is identical to **imagecopymerge()** except that when merging it preservese the hue of the source by converting the

destination pixels to gray scale before the copy operation.

> **Note:** This function was added in PHP 4.0.6

imagecopyresampled

(PHP 4 >= 4.0.6)

imagecopyresampled -- Copy and resize part of an image with resampling

Description

int **imagecopyresampled** (resource dst_im, resource src_im, int dstX, int dstY, int srcX, int srcY, int dstW, int dstH, int srcW, int srcH)

imagecopyresampled() copies a rectangular portion of one image to another image, smoothly interpolating pixel values so that, in particular, reducing the size of an image still retains a great deal of clarity. *Dst_im* is the destination image, *src_im* is the source image identifier. If the source and destination coordinates and width and heights differ, appropriate stretching or shrinking of the image fragment will be performed. The coordinates refer to the upper left corner. This function can be used to copy regions within the same image (if *dst_im* is the same as *src_im*) but if the regions overlap the results will be unpredictable.

> **Note:** There is a problem due to palette image limitations (255+1 colors). Resampling or filtering an image commonly needs more colors than 255, a kind of approximation is used to calculate the new resampled pixel and its color. With a palette image we try to allocate a new color, if that failed, we choose the closest (in theory) computed color. This is not always the closest visual color. That may produce a weird result, like blank (or visually blank) images. To skip this problem, please use a truecolor image as a destination image, such as one created by **imagecreatetruecolor()**.

> **Note: imagecopyresampled()** requires GD 2.0.l or greater.

See also **imagecopyresized()**.

imagecopyresized

(PHP 3, PHP 4)

imagecopyresized -- Copy and resize part of an image

Description

int **imagecopyresized** (resource dst_im, resource src_im, int dstX, int dstY, int srcX, int srcY, int dstW, int dstH, int srcW, int srcH)

imagecopyresized() copies a rectangular portion of one image to another image. *Dst_im* is the destination image, *src_im* is the source image identifier. If the source and destination coordinates and width and heights differ, appropriate stretching or shrinking of the image fragment will be performed. The coordinates refer to the upper left corner. This function can be used to copy regions within the same image (if *dst_im* is the same as *src_im*) but if the regions overlap the results will be unpredictable.

> **Note:** There is a problem due to palette image limitations (255+1 colors). Resampling or filtering an image commonly needs more colors than 255, a kind of approximation is used to calculate the new resampled pixel and its color. With a palette image we try to allocate a new color, if that failed, we choose the closest (in theory) computed color. This is not always the closest visual color. That may produce a weird result, like blank (or visually blank) images. To skip this problem, please use a truecolor image as a destination image, such as one created by **imagecreatetruecolor()**.

See also **imagecopyresampled()**.

imagecreate

(PHP 3, PHP 4)

imagecreate -- Create a new palette based image

Description

resource **imagecreate** (int x_size, int y_size)

imagecreate() returns an image identifier representing a blank image of size *x_size* by *y_size*.

We recommend the use of **imagecreatetruecolor()**.

Example 1. Creating a new GD image stream and outputting an image.

```
<?php
header ("Content-type: image/png");
$im = @imagecreate (50, 100)
    or die ("Cannot Initialize new GD image stream");
$background_color = imagecolorallocate ($im, 255, 255, 255);
$text_color = imagecolorallocate ($im, 233, 14, 91);
imagestring ($im, 1, 5, 5,  "A Simple Text String", $text_color);
imagepng ($im);
?>
```

See also **imagecreatetruecolor()**.

imagecreatefromgd2

(PHP 4 >= 4.1.0)

imagecreatefromgd2 -- Create a new image from GD2 file or URL

Description

resource **imagecreatefromgd2** (string filename)

Warning
This function is currently not documented; only the argument list is available.

Tip: You can use a URL as a filename with this function if the fopen wrappers have been enabled. See **fopen()** for more details on how to specify the filename and Appendix I for a list of supported URL protocols.

Warning
Windows versions of PHP prior to PHP 4.3 do not support accessing remote files via this function, even if allow_url_fopen is enabled.

imagecreatefromgd2part

(PHP 4 >= 4.1.0)

imagecreatefromgd2part -- Create a new image from a given part of GD2 file or URL

Description

resource **imagecreatefromgd2part** (string filename, int srcX, int srcY, int width, int height)

Warning
This function is currently not documented; only the argument list is available.

Tip: You can use a URL as a filename with this function if the fopen wrappers have been enabled. See **fopen()** for more details on how to specify the filename and Appendix I for a list of supported URL protocols.

Warning
Windows versions of PHP prior to PHP 4.3 do not support accessing remote files via this function, even if allow_url_fopen is enabled.

imagecreatefromgd

(PHP 4 >= 4.1.0)

imagecreatefromgd -- Create a new image from GD file or URL

Description

resource **imagecreatefromgd** (string filename)

Warning
This function is currently not documented; only the argument list is available.

Tip: You can use a URL as a filename with this function if the fopen wrappers have been enabled. See **fopen()** for more details on how to specify the filename and Appendix I for a list of supported URL protocols.

Warning
Windows versions of PHP prior to PHP 4.3 do not support accessing remote files via this function, even if allow_url_fopen is enabled.

imagecreatefromgif

(PHP 3, PHP 4)

imagecreatefromgif -- Create a new image from file or URL

Description

resource **imagecreatefromgif** (string filename)

imagecreatefromgif() returns an image identifier representing the image obtained from the given filename.

imagecreatefromgif() returns an empty string on failure. It also outputs an error message, which unfortunately displays as a broken link in a browser. To ease debugging the following example will produce an error GIF:

Example 1. Example to handle an error during creation (courtesy vic@zymsys.com)

```
function LoadGif ($imgname) {
    $im = @imagecreatefromgif ($imgname); /* Attempt to open */
    if (!$im) { /* See if it failed */
        $im = imagecreate (150, 30); /* Create a blank image */
        $bgc = imagecolorallocate ($im, 255, 255, 255);
        $tc = imagecolorallocate ($im, 0, 0, 0);
        imagefilledrectangle ($im, 0, 0, 150, 30, $bgc);
        /* Output an errmsg */
        imagestring ($im, 1, 5, 5, "Error loading $imgname", $tc);
    }
    return $im;
}
```

Note: Since all GIF support was removed from the GD library in version 1.6, this function is not available if you are using that version of the GD library.

Tip: You can use a URL as a filename with this function if the fopen wrappers have been enabled. See **fopen()** for more details on how to specify the filename and Appendix I for a list of supported URL protocols.

Warning
Windows versions of PHP prior to PHP 4.3 do not support accessing remote files via this function, even if allow_url_fopen is enabled.

imagecreatefromjpeg

(PHP 3>= 3.0.16, PHP 4)

imagecreatefromjpeg -- Create a new image from file or URL

Description

resource **imagecreatefromjpeg** (string filename)

imagecreatefromjpeg() returns an image identifier representing the image obtained from the given filename.

imagecreatefromjpeg() returns an empty string on failure. It also outputs an error message, which unfortunately displays as a broken link in a browser. To ease debugging the following example will produce an error JPEG:

Example 1. Example to handle an error during creation (courtesy vic@zymsys.com)

```
function LoadJpeg ($imgname) {
    $im = @imagecreatefromjpeg ($imgname); /* Attempt to open */
    if (!$im) { /* See if it failed */
        $im  = imagecreate (150, 30); /* Create a blank image */
        $bgc = imagecolorallocate ($im, 255, 255, 255);
        $tc  = imagecolorallocate ($im, 0, 0, 0);
        imagefilledrectangle ($im, 0, 0, 150, 30, $bgc);
        /* Output an errmsg */
        imagestring ($im, 1, 5, 5, "Error loading $imgname", $tc);
    }
    return $im;
}
```

Tip: You can use a URL as a filename with this function if the fopen wrappers have been enabled. See **fopen()** for more details on how to specify the filename and Appendix I for a list of supported URL protocols.

Warning
Windows versions of PHP prior to PHP 4.3 do not support accessing remote files via this function, even if allow_url_fopen is enabled.

imagecreatefrompng

(PHP 3>= 3.0.13, PHP 4)

imagecreatefrompng -- Create a new image from file or URL

Description

resource **imagecreatefrompng** (string filename)

imagecreatefrompng() returns an image identifier representing the image obtained from the given filename.

imagecreatefrompng() returns an empty string on failure. It also outputs an error message, which unfortunately displays as a broken link in a browser. To ease debugging the following example will produce an error PNG:

Example 1. Example to handle an error during creation (courtesy vic@zymsys.com)

```
function LoadPNG ($imgname) {
    $im = @imagecreatefrompng ($imgname); /* Attempt to open */
    if (!$im) { /* See if it failed */
        $im  = imagecreate (150, 30); /* Create a blank image */
        $bgc = imagecolorallocate ($im, 255, 255, 255);
        $tc  = imagecolorallocate ($im, 0, 0, 0);
        imagefilledrectangle ($im, 0, 0, 150, 30, $bgc);
        /* Output an errmsg */
        imagestring ($im, 1, 5, 5, "Error loading $imgname", $tc);
    }
    return $im;
}
```

Tip: You can use a URL as a filename with this function if the fopen wrappers have been enabled. See **fopen()** for more details on how to specify the filename and Appendix I for a list of supported URL protocols.

Warning
Windows versions of PHP prior to PHP 4.3 do not support accessing remote files via this function, even if allow_url_fopen is enabled.

imagecreatefromstring

(PHP 4 >= 4.0.4)

imagecreatefromstring -- Create a new image from the image stream in the string

Description

resource **imagecreatefromstring** (string image)

imagecreatefromstring() returns an image identifier representing the image obtained from the given string.

imagecreatefromwbmp

(PHP 4 >= 4.0.1)

imagecreatefromwbmp -- Create a new image from file or URL

Description

resource **imagecreatefromwbmp** (string filename)

imagecreatefromwbmp() returns an image identifier representing the image obtained from the given filename.

imagecreatefromwbmp() returns an empty string on failure. It also outputs an error message, which unfortunately displays as a broken link in a browser. To ease debugging the following example will produce an error WBMP:

Example 1. Example to handle an error during creation (courtesy vic@zymsys.com)

```
function LoadWBMP ($imgname) {
    $im = @imagecreatefromwbmp ($imgname); /* Attempt to open */
    if (!$im) { /* See if it failed */
        $im  = imagecreate (20, 20); /* Create a blank image */
        $bgc = imagecolorallocate ($im, 255, 255, 255);
        $tc  = imagecolorallocate ($im, 0, 0, 0);
        imagefilledrectangle ($im, 0, 0, 10, 10, $bgc);
        /* Output an errmsg */
        imagestring ($im, 1, 5, 5, "Error loading $imgname", $tc);
    }
    return $im;
}
```

Note: WBMP support is only available if PHP was compiled against GD-1.8 or later.

Tip: You can use a URL as a filename with this function if the fopen wrappers have been enabled. See **fopen()** for more details on how to specify the filename and Appendix I for a list of supported URL protocols.

Warning
Windows versions of PHP prior to PHP 4.3 do not support accessing remote files via this function, even if allow_url_fopen is enabled.

imagecreatefromxbm

(PHP 4 >= 4.0.1)

imagecreatefromxbm -- Create a new image from file or URL

Description

resource **imagecreatefromxbm** (string filename)

imagecreatefromxbm() returns an image identifier representing the image obtained from the given filename.

Tip: You can use a URL as a filename with this function if the fopen wrappers have been enabled. See **fopen()** for more details on how to specify the filename and Appendix I for a list of supported URL protocols.

Warning
Windows versions of PHP prior to PHP 4.3 do not support accessing remote files via this function, even if allow_url_fopen is enabled.

imagecreatefromxpm

(PHP 4 >= 4.0.1)

imagecreatefromxpm -- Create a new image from file or URL

Description

resource **imagecreatefromxpm** (string filename)

imagecreatefromxpm() returns an image identifier representing the image obtained from the given filename.

> **Tip:** You can use a URL as a filename with this function if the fopen wrappers have been enabled. See **fopen()** for more details on how to specify the filename and Appendix I for a list of supported URL protocols.

Warning
Windows versions of PHP prior to PHP 4.3 do not support accessing remote files via this function, even if allow_url_fopen is enabled.

imagecreatetruecolor

(PHP 4 >= 4.0.6)

imagecreatetruecolor -- Create a new true color image

Description

resource **imagecreatetruecolor** (int x_size, int y_size)

imagecreatetruecolor() returns an image identifier representing a black image of size *x_size* by *y_size*.

> **Note:** This function was added in PHP 4.0.6 and requires GD 2.0.1 or later
>
> **Note:** This function will not work with GIF file formats.

imagedashedline

(PHP 3, PHP 4)

imagedashedline -- Draw a dashed line

Description

int **imagedashedline** (resource image, int x1, int y1, int x2, int y2, int col)

This function is deprecated. Use combination of **imagesetstyle()** and **imageline()** instead.

imagedestroy

(PHP 3, PHP 4)

imagedestroy -- Destroy an image

Description

int **imagedestroy** (resource image)

imagedestroy() frees any memory associated with image *image*. *image* is the image identifier returned by the **imagecreate()** function.

imageellipse

(PHP 4 >= 4.0.6)

imageellipse -- Draw an ellipse

Description

int **imageellipse** (resource image, int cx, int cy, int w, int h, int col)

imageellipse() draws an ellipse centered at *cx*, *cy* (top left is 0, 0) in the image represented by *image*. *w* and *h* specifies the ellipse's width and height respectively. The color of the ellipse is specified by *color*.

> **Note:** This function was added in PHP 4.0.6 and requires GD 2.0.2 or later which can be obtained at http://www.boutell.com/gd/

See also **imagearc()**.

imagefill

(PHP 3, PHP 4)

imagefill -- Flood fill

Description

int **imagefill** (resource image, int x, int y, int col)

imagefill() performs a flood fill starting at coordinate *x*, *y* (top left is 0, 0) with color *col* in the image *image*.

imagefilledarc

(PHP 4 >= 4.0.6)

imagefilledarc -- Draw a partial ellipse and fill it

Description

int **imagefilledarc** (resource image, int cx, int cy, int w, int h, int s, int e, int col, int style)

imagefilledarc() draws a partial ellipse centered at *cx*, *cy* (top left is 0, 0) in the image represented by *image*. *w* and *h* specifies the ellipse's width and height respectively while the start and end points are specified in degrees indicated by the *s* and *e* arguments. *style* is a bitwise OR of the following possibilities:

1. IMG_ARC_PIE
2. IMG_ARC_CHORD
3. IMG_ARC_NOFILL
4. IMG_ARC_EDGED

IMG_ARC_PIE and IMG_ARC_CHORD are mutually exclusive; IMG_ARC_CHORD just connects the starting and ending angles with a straight line, while IMG_ARC_PIE produces a rounded edge. IMG_ARC_NOFILL indicates that the arc or chord should be outlined, not filled. IMG_ARC_EDGED, used together with IMG_ARC_NOFILL, indicates that the beginning and ending angles should be connected to the center - this is a good way to outline (rather than fill) a 'pie slice'.

> **Note:** This function was added in PHP 4.0.6 and requires GD 2.0.1

imagefilledellipse

(PHP 4 >= 4.0.6)

imagefilledellipse -- Draw a filled ellipse

Description

int **imagefilledellipse** (resource image, int cx, int cy, int w, int h, int col)

imagefilledellipse() draws an ellipse centered at *cx*, *cy* (top left is 0, 0) in the image represented by *image*. *w* and *h* specifies the ellipse's width and height respectively. The ellipse is filled using *color*

> **Note:** This function was added in PHP 4.0.6 and requires GD 2.0.1 or later

See also **imagefilledarc()**.

imagefilledpolygon

(PHP 3, PHP 4)

imagefilledpolygon -- Draw a filled polygon

Description

int **imagefilledpolygon** (resource image, array points, int num_points, int col)

imagefilledpolygon() creates a filled polygon in image *image*. *points* is a PHP array containing the polygon's vertices, ie. points[0] = x0, points[1] = y0, points[2] = x1, points[3] = y1, etc. *num_points* is the total number of vertices.

imagefilledrectangle

(PHP 3, PHP 4)

imagefilledrectangle -- Draw a filled rectangle

Description

int **imagefilledrectangle** (resource image, int x1, int y1, int x2, int y2, int col)

imagefilledrectangle() creates a filled rectangle of color *col* in image *image* starting at upper left coordinates *x1*, *y1* and ending at bottom right coordinates *x2*, *y2*. 0, 0 is the top left corner of the image.

imagefilltoborder

(PHP 3, PHP 4)

imagefilltoborder -- Flood fill to specific color

Description

int **imagefilltoborder** (resource image, int x, int y, int border, int col)

imagefilltoborder() performs a flood fill whose border color is defined by *border*. The starting point for the fill is *x*, *y* (top left is 0, 0) and the region is filled with color *col*.

imagefontheight

(PHP 3, PHP 4)

imagefontheight -- Get font height

Description

int **imagefontheight** (int font)

Returns the pixel height of a character in the specified font.

See also **imagefontwidth()** and **imageloadfont()**.

imagefontwidth

(PHP 3, PHP 4)

imagefontwidth -- Get font width

Description

int **imagefontwidth** (int font)

Returns the pixel width of a character in font.

See also **imagefontheight()** and **imageloadfont()**.

imageftbbox

(PHP 4 >= 4.1.0)

imageftbbox -- Give the bounding box of a text using fonts via freetype2

Description

array **imageftbbox** (int size, int angle, string font_file, string text [, array extrainfo])

Warning
This function is currently not documented; only the argument list is available.

imagefttext

(PHP 4 >= 4.1.0)

imagefttext -- Write text to the image using fonts using FreeType 2

Description

array **imagefttext** (resource image, int size, int angle, int x, int y, int col, string font_file, string text [, array extrainfo])

Warning
This function is currently not documented; only the argument list is available.

imagegammacorrect

(PHP 3>= 3.0.13, PHP 4)

imagegammacorrect -- Apply a gamma correction to a GD image

Description

int **imagegammacorrect** (resource image, float inputgamma, float outputgamma)

The **imagegammacorrect()** function applies gamma correction to a gd image stream (*image*) given an input gamma, the parameter *inputgamma* and an output gamma, the parameter *outputgamma*.

imagegd2

(PHP 4 >= 4.1.0)

imagegd2 -- Output GD2 image to browser or file

Description

int **imagegd2** (resource image [, string filename [, int chunk_size [, string type]]])

Warning
This function is currently not documented; only the argument list is available.

The optional *type* parameter is either `raw` or `compressed`.

> **Note:** The optional *chunk_size* and *type* parameters became available in PHP 4.3.1.

imagegd

(PHP 4 >= 4.1.0)

imagegd -- Output GD image to browser or file

Description

int **imagegd** (resource image [, string filename])

Warning
This function is currently not documented; only the argument list is available.

imagegif

(PHP 3, PHP 4)

imagegif -- Output image to browser or file

Description

int **imagegif** (resource image [, string filename])

imagegif() creates the GIF file in filename from the image *image*. The *image* argument is the return from the **imagecreate()** function.

The image format will be GIF87a unless the image has been made transparent with **imagecolortransparent()**, in which case the image format will be GIF89a.

The filename argument is optional, and if left off, the raw image stream will be output directly. By sending an image/gif content-type using **header()**, you can create a PHP script that outputs GIF images directly.

> **Note:** Since all GIF support was removed from the GD library in version 1.6, this function is not available if you are using that version of the GD library.
>
> The following code snippet allows you to write more portable PHP applications by auto-detecting the type of GD support which is available. Replace the sequence `header ("Content-type: image/gif"); imagegif ($im);` by the more flexible sequence:

```
<?php
if (function_exists("imagegif")) {
    header ("Content-type: image/gif");
    imagegif ($im);
}
elseif (function_exists("imagejpeg")) {
    header ("Content-type: image/jpeg");
```

```
    imagejpeg ($im, "", 0.5);
}
elseif (function_exists("imagepng")) {
    header ("Content-type: image/png");
    imagepng ($im);
}
elseif (function_exists("imagewbmp")) {
    header ("Content-type: image/vnd.wap.wbmp");
    imagewbmp ($im);
}
else
    die("No image support in this PHP server");
?>
```

Note: As of version 3.0.18 and 4.0.2 you can use the function **imagetypes()** in place of **function_exists()** for checking the presence of the various supported image formats:

```
if (imagetypes() & IMG_GIF) {
    header ("Content-type: image/gif");
    imagegif ($im);
}
elseif (imagetypes() & IMG_JPG) {
        ... etc.
```

See also **imagepng()**, **imagewbmp()**, **imagejpeg()**, **imagetypes()**.

imageinterlace

(PHP 3, PHP 4)

imageinterlace -- Enable or disable interlace

Description

int **imageinterlace** (resource image [, int interlace])

imageinterlace() turns the interlace bit on or off. If interlace is 1 the image will be interlaced, and if interlace is 0 the interlace bit is turned off.

If the interlace bit is set and the image is used as a JPEG image, the image is created as a progressive JPEG.

This function returns whether the interlace bit is set for the image.

imagejpeg

(PHP 3>= 3.0.16, PHP 4)

imagejpeg -- Output image to browser or file

Description

int **imagejpeg** (resource image [, string filename [, int quality]])

imagejpeg() creates the JPEG file in filename from the image *image*. The *image* argument is the return from the **imagecreate()** function.

The filename argument is optional, and if left off, the raw image stream will be output directly. To skip the filename argument in order to provide a quality argument just use an empty string (''). By sending an image/jpeg content-type using **header()**, you can create a PHP script that outputs JPEG images directly.

Note: JPEG support is only available if PHP was compiled against GD-1.8 or later.

quality is optional, and ranges from 0 (worst quality, smaller file) to 100 (best quality, biggest file). The default is the default IJG quality value (about 75).

If you want to output Progressive JPEGs, you need to set interlacing on with **imageinterlace()**.

See also **imagepng()**, **imagegif()**, **imagewbmp()**, **imageinterlace()** and **imagetypes()**.

imageline

(PHP 3, PHP 4)

imageline -- Draw a line

Description

int **imageline** (resource image, int x1, int y1, int x2, int y2, int col)

imageline() draws a line from *x1*, *y1* to *x2*, *y2* (top left is 0, 0) in image im of color *col*.

See also **imagecreate()** and **imagecolorallocate()**.

imageloadfont

(PHP 3, PHP 4)

imageloadfont -- Load a new font

Description

int **imageloadfont** (string file)

imageloadfont() loads a user-defined bitmap font and returns an identifier for the font (that is always greater than 5, so it will not conflict with the built-in fonts).

The font file format is currently binary and architecture dependent. This means you should generate the font files on the same type of CPU as the machine you are running PHP on.

Table 1. Font file format

byte position	C data type	description
byte 0-3	int	number of characters in the font
byte 4-7	int	value of first character in the font (often 32 for space)
byte 8-11	int	pixel width of each character
byte 12-15	int	pixel height of each character
byte 16-	char	array with character data, one byte per pixel in each character, for a total of (nchars*width*height) bytes.

See also **imagefontwidth()** and **imagefontheight()**.

imagepalettecopy

(PHP 4 >= 4.0.1)

imagepalettecopy -- Copy the palette from one image to another

Description

int **imagepalettecopy** (resource destination, resource source)

imagepalettecopy() copies the palette from the *source* image to the *destination* image.

imagepng

(PHP 3>= 3.0.13, PHP 4)

imagepng -- Output a PNG image to either the browser or a file

Description

int **imagepng** (resource image [, string filename])

The **imagepng()** outputs a GD image stream (*image*) in PNG format to standard output (usually the browser) or, if a filename is given by the *filename* it outputs the image to the file.

```
<?php
$im = imagecreatefrompng ("test.png");
imagepng ($im);
?>
```

See also **imagegif()**, **imagewbmp()**, **imagejpeg()**, **imagetypes()**.

imagepolygon

(PHP 3, PHP 4)

imagepolygon -- Draw a polygon

Description

int **imagepolygon** (resource image, array points, int num_points, int col)

imagepolygon() creates a polygon in image id. *points* is a PHP array containing the polygon's vertices, ie. points[0] = x0, points[1] = y0, points[2] = x1, points[3] = y1, etc. *num_points* is the total number of points (vertices).

See also **imagecreate()** and **imagecreatetruecolor()**.

imagepsbbox

(PHP 3>= 3.0.9, PHP 4)

imagepsbbox -- Give the bounding box of a text rectangle using PostScript Type1 fonts

Description

array **imagepsbbox** (string text, int font, int size [, int space [, int tightness [, float angle]]])

Size is expressed in pixels.

Space allows you to change the default value of a space in a font. This amount is added to the normal value and can also be negative.

Tightness allows you to control the amount of white space between characters. This amount is added to the normal character width and can also be negative.

Angle is in degrees.

Parameters *space* and *tightness* are expressed in character space units, where 1 unit is 1/1000th of an em-square.

Parameters *space*, *tightness*, and *angle* are optional.

The bounding box is calculated using information available from character metrics, and unfortunately tends to differ slightly from the results achieved by actually rasterizing the text. If the angle is 0 degrees, you can expect the text to need 1 pixel more to every direction.

This function returns an array containing the following elements:

0	lower left x-coordinate
1	lower left y-coordinate
2	upper right x-coordinate
3	upper right y-coordinate

See also **imagepstext()**.

imagepscopyfont

(PHP 3>= 3.0.9, PHP 4)

imagepscopyfont -- Make a copy of an already loaded font for further modification

Description

int **imagepscopyfont** (int fontindex)

Use this function if you need make further modifications to the font, for example extending/condensing, slanting it or changing it's character encoding vector, but need to keep the original along as well. Note that the font you want to copy must be one obtained using **imagepsloadfont()**, not a font that is itself a copied one. You can although make modifications to it before copying.

If you use this function, you *must* free the fonts obtained this way yourself and in reverse order. Otherwise your script *will* hang.

In the case everything went right, a valid font index will be returned and can be used for further purposes. Otherwise the function returns **FALSE** and prints a message describing what went wrong.

See also **imagepsloadfont()**.

imagepsencodefont

(PHP 3>= 3.0.9, PHP 4)

imagepsencodefont -- Change the character encoding vector of a font

Description

int **imagepsencodefont** (int font_index, string encodingfile)

Loads a character encoding vector from from a file and changes the fonts encoding vector to it. As a PostScript fonts default vector lacks most of the character positions above 127, you'll definitely want to change this if you use an other language than english. The exact format of this file is described in T1libs documentation. T1lib comes with two ready-to-use files, IsoLatin1.enc and IsoLatin2.enc.

If you find yourself using this function all the time, a much better way to define the encoding is to set ps.default_encoding in the configuration file to point to the right encoding file and all fonts you load will automatically have the right encoding.

imagepsextendfont

(PHP 3>= 3.0.9, PHP 4)

imagepsextendfont -- Extend or condense a font

Description

bool **imagepsextendfont** (int font_index, float extend)

Extend or condense a font (*font_index*), if the value of the *extend* parameter is less than one you will be condensing the font.

imagepsfreefont

(PHP 3>= 3.0.9, PHP 4)

imagepsfreefont -- Free memory used by a PostScript Type 1 font

Description

void **imagepsfreefont** (int fontindex)

See also **imagepsloadfont()**.

imagepsloadfont

(PHP 3>= 3.0.9, PHP 4)

imagepsloadfont -- Load a PostScript Type 1 font from file

Description

int **imagepsloadfont** (string filename)

In the case everything went right, a valid font index will be returned and can be used for further purposes. Otherwise the function returns **FALSE** and prints a message describing what went wrong, which you cannot read directly, while the output type is image.

```
<?php
header ("Content-type: image/jpeg");
$im = imagecreate (350, 45);
$black = imagecolorallocate ($im, 0, 0, 0);
$white = imagecolorallocate ($im, 255, 255, 255);
$font = imagepsloadfont ("bchbi.pfb"); // or locate your .pfb files on your machine
imagepstext ($im, "Testing... It worked!", $font, 32, $white, $black, 32, 32);
imagepsfreefont ($font);
imagejpeg ($im, "", 100); //for best quality...your mileage may vary
imagedestroy ($im);
?>
```

See also **imagepsfreefont()**.

imagepsslantfont

(PHP 3>= 3.0.9, PHP 4)

imagepsslantfont -- Slant a font

Description

bool **imagepsslantfont** (int font_index, float slant)

Slant a font given by the *font_index* parameter with a slant of the value of the *slant* parameter.

imagepstext

(PHP 3>= 3.0.9, PHP 4)

imagepstext -- To draw a text string over an image using PostScript Type1 fonts

Description

array **imagepstext** (resource image, string text, int font, int size, int foreground, int background, int x, int y [, int space [, int tightness [, float angle [, int antialias_steps]]]])

Size is expressed in pixels.

Foreground is the color in which the text will be painted. *Background* is the color to which the text will try to fade in with antialiasing. No pixels with the color *background* are actually painted, so the background image does not need to be of solid color.

The coordinates given by *x*, *y* will define the origin (or reference point) of the first character (roughly the lower-left corner of

the character). This is different from the **imagestring()**, where *x*, *y* define the upper-right corner of the first character. Refer to PostScipt documentation about fonts and their measuring system if you have trouble understanding how this works.

Space allows you to change the default value of a space in a font. This amount is added to the normal value and can also be negative.

Tightness allows you to control the amount of white space between characters. This amount is added to the normal character width and can also be negative.

Angle is in degrees.

Antialias_steps allows you to control the number of colours used for antialiasing text. Allowed values are 4 and 16. The higher value is recommended for text sizes lower than 20, where the effect in text quality is quite visible. With bigger sizes, use 4. It's less computationally intensive.

Parameters *space* and *tightness* are expressed in character space units, where 1 unit is 1/1000th of an em-square.

Parameters *space*, *tightness*, *angle* and *antialias* are optional.

This function returns an array containing the following elements:

0	lower left x-coordinate
1	lower left y-coordinate
2	upper right x-coordinate
3	upper right y-coordinate

See also **imagepsbbox()**.

imagerectangle

(PHP 3, PHP 4)

imagerectangle -- Draw a rectangle

Description

int **imagerectangle** (resource image, int x1, int y1, int x2, int y2, int col)

imagerectangle() creates a rectangle of color col in image *image* starting at upper left coordinate x1, y1 and ending at bottom right coordinate x2, y2. 0, 0 is the top left corner of the image.

imagerotate

(PHP 4 >= 4.3.0)

imagerotate -- Rotate an image with a given angle

Description

ressource **imagerotate** (resource src_im, float angle, int bgd_color)

Rotates the *src_im* image using a given *angle* in degree. *bgd_color* specifies the color of the uncovered zone after the rotation.

imagesetbrush

(PHP 4 >= 4.0.6)

imagesetbrush -- Set the brush image for line drawing

Description

int **imagesetbrush** (resource image, resource brush)

imagesetbrush() sets the brush image to be used by all line drawing functions (such as **imageline()** and **imagepolygon()**) when drawing with the special colors `IMG_COLOR_BRUSHED` or `IMG_COLOR_STYLEDBRUSHED`.

> **Note:** You need not take special action when you are finished with a brush, but if you destroy the brush image, you must not use the `IMG_COLOR_BRUSHED` or `IMG_COLOR_STYLEDBRUSHED` colors until you have set a new brush image!

> **Note:** This function was added in PHP 4.0.6

imagesetpixel

(PHP 3, PHP 4)

imagesetpixel -- Set a single pixel

Description

int **imagesetpixel** (resource image, int x, int y, int col)

imagesetpixel() draws a pixel at *x*, *y* (top left is 0, 0) in image *image* of color *col*.

See also **imagecreate()** and **imagecolorallocate()**.

imagesetstyle

(PHP 4 >= 4.0.6)

imagesetstyle -- Set the style for line drawing

Description

int **imagesetstyle** (resource image, array style)

imagesetstyle() sets the style to be used by all line drawing functions (such as **imageline()** and **imagepolygon()**) when drawing with the special color `IMG_COLOR_STYLED` or lines of images with color `IMG_COLOR_STYLEDBRUSHED`.

The *style* parameter is an array of pixels. Following example script draws a dashed line from upper left to lower right corner of the canvas:

Example 1. imagesetstyle() example

```
<?php
header ("Content-type: image/jpeg");
$im  = imagecreate (100, 100);
$w   = imagecolorallocate ($im, 255, 255, 255);
$red = imagecolorallocate ($im, 255, 0, 0);

/* Draw a dashed line, 5 red pixels, 5 white pixels */
$style = array ($red,$red,$red,$red,$red,$w,$w,$w,$w,$w);
imagesetstyle ($im, $style);
imageline ($im, 0, 0, 100, 100, IMG_COLOR_STYLED);

/* Draw a line of happy faces using imagesetbrush() with imagesetstyle */
$style = array ($w,$w,$w,$w,$w,$w,$w,$w,$w,$w,$w,$w,$w,$red);
imagesetstyle ($im, $style);

$brush = imagecreatefrompng ("http://www.libpng.org/pub/png/images/smile.happy.png");
$w2 = imagecolorallocate($brush,255,255,255);
imagecolortransparent ($brush, $w2);
imagesetbrush ($im, $brush);
imageline ($im, 100, 0, 0, 100, IMG_COLOR_STYLEDBRUSHED);

imagejpeg ($im);
imagedestroy ($im);
?>
```

See also **imagesetbrush()**, **imageline()**.

> **Note:** This function was added in PHP 4.0.6

imagesetthickness

(PHP 4 >= 4.0.6)

imagesetthickness -- Set the thickness for line drawing

Description

void **imagesetthickness** (resource image, int thickness)

imagesetthickness() sets the thickness of the lines drawn when drawing rectangles, polygons, ellipses etc. etc. to *thickness* pixels.

> **Note:** This function was added in PHP 4.0.6 and requires GD 2.0.1 or later

imagesettile

(PHP 4 >= 4.0.6)

imagesettile -- Set the tile image for filling

Description

int **imagesettile** (resource image, resource tile)

imagesettile() sets the tile image to be used by all region filling functions (such as **imagefill()** and **imagefilledpolygon()**) when filling with the special color IMG_COLOR_TILED.

A tile is an image used to fill an area with a repeated pattern. *Any* GD image can be used as a tile, and by setting the transparent color index of the tile image with **imagecolortransparent()**, a tile allows certain parts of the underlying area to shine through can be created.

> **Note:** You need not take special action when you are finished with a tile, but if you destroy the tile image, you must not use the IMG_COLOR_TILED color until you have set a new tile image!

> **Note:** This function was added in PHP 4.0.6

imagestring

(PHP 3, PHP 4)

imagestring -- Draw a string horizontally

Description

int **imagestring** (resource image, int font, int x, int y, string s, int col)

imagestring() draws the string *s* in the image identified by *image* at coordinates *x*, *y* (top left is 0, 0) in color *col*. If font is 1, 2, 3, 4 or 5, a built-in font is used.

See also **imageloadfont()**.

imagestringup

(PHP 3, PHP 4)

imagestringup -- Draw a string vertically

Description

int **imagestringup** (resource image, int font, int x, int y, string s, int col)

imagestringup() draws the string *s* vertically in the image identified by *image* at coordinates *x*, *y* (top left is 0, 0) in color *col*. If font is 1, 2, 3, 4 or 5, a built-in font is used.

See also **imageloadfont()**.

imagesx

(PHP 3, PHP 4)

imagesx -- Get image width

Description

int **imagesx** (resource image)

imagesx() returns the width of the image identified by *image*.

See also **imagecreate()** and **imagesy()**.

imagesy

(PHP 3, PHP 4)

imagesy -- Get image height

Description

int **imagesy** (resource image)

imagesy() returns the height of the image identified by *image*.

See also **imagecreate()** and **imagesx()**.

imagetruecolortopalette

(PHP 4 >= 4.0.6)

imagetruecolortopalette -- Convert a true color image to a palette image

Description

void **imagetruecolortopalette** (resource image, bool dither, int ncolors)

imagetruecolortopalette() converts a truecolor image to a palette image. The code for this function was originally drawn from the Independent JPEG Group library code, which is excellent. The code has been modified to preserve as much alpha channel information as possible in the resulting palette, in addition to preserving colors as well as possible. This does not work as well as might be hoped. It is usually best to simply produce a truecolor output image instead, which guarantees the highest output quality.

dither indicates if the image should be dithered - if it is **TRUE** then dithering will be used which will result in a more speckled image but with better color approximation.

ncolors sets the maximum number of colors that should be retained in the palette.

> **Note:** This function was added in PHP 4.0.6 and requires GD 2.0.1 or later

imagettfbbox

(PHP 3>= 3.0.1, PHP 4)

imagettfbbox -- Give the bounding box of a text using TrueType fonts

Description

array **imagettfbbox** (int size, int angle, string fontfile, string text)

This function calculates and returns the bounding box in pixels for a TrueType text.

text

The string to be measured.

size

The font size in pixels.

fontfile

The name of the TrueType font file. (Can also be an URL.) Depending on which version of the GD library that PHP is using, it may attempt to search for files that do not begin with a leading '/' by appending '.ttf' to the filename and searching along a library-defined font path.

angle

Angle in degrees in which *text* will be measured.

imagettfbbox() returns an array with 8 elements representing four points making the bounding box of the text:

0	lower left corner, X position
1	lower left corner, Y position
2	lower right corner, X position
3	lower right corner, Y position
4	upper right corner, X position
5	upper right corner, Y position
6	upper left corner, X position
7	upper left corner, Y position

The points are relative to the *text* regardless of the angle, so "upper left" means in the top left-hand corner seeing the text horizontallty.

This function requires both the GD library and the FreeType library.

See also **imagettftext()**.

imagettftext

(PHP 3, PHP 4)

imagettftext -- Write text to the image using TrueType fonts

Description

array **imagettftext** (resource im, int size, int angle, int x, int y, int col, string fontfile, string text)

imagettftext() draws the string *text* in the image identified by *image*, starting at coordinates *x*, *y* (top left is 0, 0), at an angle of *angle* in color *col*, using the TrueType font file identified by *fontfile*. Depending on which version of the GD library that PHP is using, when *fontfile* does not begin with a leading '/', '.ttf' will be appended to the filename and the library will attempt to search for that filename along a library-defined font path.

The coordinates given by *x*, *y* will define the basepoint of the first character (roughly the lower-left corner of the character). This is different from the **imagestring()**, where x, y define the upper-right corner of the first character.

angle is in degrees, with 0 degrees being left-to-right reading text (3 0'clock direction), and higher values representing a counter-clockwise rotation. (i.e., a value of 90 would result in bottom-to-top reading text).

fontfile is the path to the TrueType font you wish to use.

text is the text string which may include UTF-8 character sequences (of the form: {) to access characters in a font beyond the first 255.

Col is the color index. Using the negative of a color index has the effect of turning off antialiasing.

imagettftext() returns an array with 8 elements representing four points making the bounding box of the text. The order of the points is lower left, lower right, upper right, upper left. The points are relative to the text regardless of the angle, so "upper left" means in the top left-hand corner when you see the text horizontallty.

This example script will produce a black GIF 400x30 pixels, with the words "Testing..." in white in the font Arial.

Example 1. imagettftext() example

```
<?php
  header("Content-type: image/jpeg");
  $im = imagecreate(400,30);
  $white = imagecolorallocate($im, 255,255,255);
  $black = imagecolorallocate($im, 0,0,0);

  // Replace path by your own font path
  imagettftext($im, 20, 0, 10, 20, $black, "/path/arial.ttf",
  "Testing... Omega: &amp;#937;");
  imagejpeg($im);
  imagedestroy($im);
?>
```

This function requires both the GD library and the FreeType library.

See also **imagettfbbox()**.

imagetypes

(PHP 3 CVS only, PHP 4 >= 4.0.2)

imagetypes -- Return the image types supported by this PHP build

Description

int **imagetypes** (void)

This function returns a bit-field corresponding to the image formats supported by the version of GD linked into PHP. The following bits are returned, `IMG_GIF` | `IMG_JPG` | `IMG_PNG` | `IMG_WBMP`. To check for PNG support, for example, do this:

Example 1. imagetypes

```
<?php
if (imagetypes() & IMG_PNG) {
    echo "PNG Support is enabled";
}
?>
```

imagewbmp

(PHP 3>= 3.0.15, PHP 4 >= 4.0.1)

imagewbmp -- Output image to browser or file

Description

int **imagewbmp** (resource image [, string filename [, int foreground]])

imagewbmp() creates the WBMP file in filename from the image *image*. The *image* argument is the return from the **imagecreate()** function.

The filename argument is optional, and if left off, the raw image stream will be output directly. By sending an image/vnd.wap.wbmp content-type using **header()**, you can create a PHP script that outputs WBMP images directly.

> **Note:** WBMP support is only available if PHP was compiled against GD-1.8 or later.

Using the optional *foreground* parameter, you can set the foreground color. Use an identifier obtained from **imagecolorallocate()**. The default foreground color is black.

See also **image2wbmp()**, **imagepng()**, **imagegif()**, **imagejpeg()**, **imagetypes()**.

iptcembed

(PHP 3>= 3.0.7, PHP 4)

iptcembed -- Embed binary IPTC data into a JPEG image

Description

array **iptcembed** (string iptcdata, string jpeg_file_name [, int spool])

Warning
This function is currently not documented; only the argument list is available.

iptcparse

(PHP 3>= 3.0.6, PHP 4)

iptcparse -- Parse a binary IPTC http://www.iptc.org/ block into single tags.

Description

array **iptcparse** (string iptcblock)

This function parses a binary IPTC block into its single tags. It returns an array using the tagmarker as an index and the value as the value. It returns **FALSE** on error or if no IPTC data was found. See **getimagesize()** for a sample.

jpeg2wbmp

(PHP 4 >= 4.0.5)

jpeg2wbmp -- Convert JPEG image file to WBMP image file

Description

int **jpeg2wbmp** (string jpegname, string wbmpname, int d_height, int d_width, int threshold)

Converts the *jpegname* JPEG file to WBMP format, and saves it as *wbmpname*. With the *d_height* and *d_width* you specify the height and width of the destination image.

Note: WBMP support is only available if PHP was compiled against GD-1.8 or later.

See also **png2wbmp()**.

png2wbmp

(PHP 4 >= 4.0.5)

png2wbmp -- Convert PNG image file to WBMP image file

Description

int **png2wbmp** (string pngname, string wbmpname, int d_height, int d_width, int threshold)

Converts the *pngname* PNG file to WBMP format, and saves it as *wbmpname*. With the *d_height* and *d_width* you specify the height and width of the destination image.

Note: WBMP support is only available if PHP was compiled against GD-1.8 or later.

See also **jpeg2wbmp()**.

read_exif_data

(PHP 4 >= 4.0.1)

read_exif_data -- Reads header information stored in TIFF and JPEG images

Description

array **exif_read_data** (string filename, string sections, bool arrays, bool thumbnail)

> **Note:** The **read_exif_data()** function is an alias for **exif_read_data()**.

See also **exif_thumbnail()**.

XLII. IMAP, POP3 and NNTP functions

Introduction

These functions are not limited to the IMAP protocol, despite their name. The underlying c-client library also supports NNTP, POP3 and local mailbox access methods.

Requirements

This extension requires the c-client library to be installed. Grab the latest version from ftp://ftp.cac.washington.edu/imap/ and compile it.

It's important that you do not copy the IMAP source files directly into the system include directory as there may be conflicts. Instead, create a new directory inside the system include directory, such as `/usr/local/imap-2000b/` (location and name depend on your setup and IMAP version), and inside this new directory create additional directories named `lib/` and `include/`. From the `c-client` directory from your IMAP source tree, copy all the `*.h` files into `include/` and all the `*.c` files into `lib/`. Additionally when you compiled IMAP, a file named `c-client-a` was created. Also put this in the `lib/` directory but rename it as `libc-client.a`.

> **Note:** To build the c-client library with SSL or/and Kerberos support read the docs supplied with the package.

Installation

To get these functions to work, you have to compile PHP with `--with-imap[=DIR]`, where DIR is the c-client install prefix. From our example above, you would use `--with-imap=/usr/local/imap-2000b`. This location depends on where you created this directory according to the description above.

> **Note:** Depending how the c-client was configured, you might also need to add `--with-imap-ssl=/path/to/openssl/` and/or `--with-kerberos=/path/to/kerberos` into the PHP configure line.

Warning
The IMAP extension cannot be used in conjuction with the recode or YAZ extensions. This is due to the fact that they both share the same internal symbol.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

NIL (integer)

OP_DEBUG (integer)

OP_READONLY (integer)

OP_ANONYMOUS (integer)

OP_SHORTCACHE (integer)

OP_SILENT (integer)

OP_PROTOTYPE (integer)

OP_HALFOPEN (integer)

OP_EXPUNGE (integer)

OP_SECURE (integer)

CL_EXPUNGE (integer)

FT_UID (integer)

FT_PEEK (integer)

FT_NOT (integer)

FT_INTERNAL (integer)

FT_PREFETCHTEXT (integer)

ST_UID (integer)

ST_SILENT (integer)

ST_SET (integer)

CP_UID (integer)

CP_MOVE (integer)

SE_UID (integer)

SE_FREE (integer)

SE_NOPREFETCH (integer)

SO_FREE (integer)

SO_NOSERVER (integer)

SA_MESSAGES (integer)

SA_RECENT (integer)

SA_UNSEEN (integer)

SA_UIDNEXT (integer)

SA_UIDVALIDITY (integer)

SA_ALL (integer)

LATT_NOINFERIORS (integer)

LATT_NOSELECT (integer)

LATT_MARKED (integer)

LATT_UNMARKED (integer)

SORTDATE (integer)

SORTARRIVAL (integer)

SORTFROM (integer)

SORTSUBJECT (integer)

SORTTO (integer)

SORTCC (integer)

SORTSIZE (integer)

TYPETEXT (integer)

TYPEMULTIPART (integer)

TYPEMESSAGE (integer)

TYPEAPPLICATION (integer)

TYPEAUDIO (integer)

TYPEIMAGE (integer)

TYPEVIDEO (integer)

TYPEOTHER (integer)

ENC7BIT (integer)

ENC8BIT (integer)

ENCBINARY (integer)

ENCBASE64 (integer)

ENCQUOTEDPRINTABLE (integer)

ENCOTHER (integer)

See Also

This document can't go into detail on all the topics touched by the provided functions. Further information is provided by the documentation of the c-client library source (`docs/internal.txt`). and the following RFC documents:

- RFC2821: Simple Mail Transfer Protocol (SMTP).
- RFC2822: Standard for ARPA internet text messages.
- RFC2060: Internet Message Access Protocol (IMAP) Version 4rev1.
- RFC1939: Post Office Protocol Version 3 (POP3).
- RFC977: Network News Transfer Protocol (NNTP).
- RFC2076: Common Internet Message Headers.
- RFC2045 , RFC2046 , RFC2047 , RFC2048 & RFC2049: Multipurpose Internet Mail Extensions (MIME).

A detailed overview is also available in the books Programming Internet Email by David Wood and Managing IMAP by Dianna Mullet & Kevin Mullet.

Table of Contents

imap_append -- Append a string message to a specified mailbox
imap_base64 -- Decode BASE64 encoded text
imap_binary -- Convert an 8bit string to a base64 string
imap_body -- Read the message body
imap_bodystruct -- Read the structure of a specified body section of a specific message
imap_check -- Check current mailbox
imap_clearflag_full -- Clears flags on messages
imap_close -- Close an IMAP stream
imap_createmailbox -- Create a new mailbox
imap_delete -- Mark a messge for deletion from current mailbox
imap_deletemailbox -- Delete a mailbox
imap_errors -- This function returns all of the IMAP errors (if any) that have occurred during this page request or since the error stack was reset.
imap_expunge -- Delete all messages marked for deletion
imap_fetch_overview -- Read an overview of the information in the headers of the given message
imap_fetchbody -- Fetch a particular section of the body of the message
imap_fetchheader -- Returns header for a message
imap_fetchstructure -- Read the structure of a particular message
imap_get_quota -- Retrieve the quota level settings, and usage statics per mailbox
imap_get_quotaroot -- Retrieve the quota settings per user
imap_getmailboxes -- Read the list of mailboxes, returning detailed information on each one
imap_getsubscribed -- List all the subscribed mailboxes
imap_header -- Read the header of the message
imap_headerinfo -- Read the header of the message
imap_headers -- Returns headers for all messages in a mailbox
imap_last_error -- This function returns the last IMAP error (if any) that occurred during this page request
imap_list -- Read the list of mailboxes
imap_listmailbox -- Read the list of mailboxes
imap_listscan -- Read the list of mailboxes, takes a string to search for in the text of the mailbox
imap_listsubscribed -- List all the subscribed mailboxes
imap_lsub -- List all the subscribed mailboxes
imap_mail_compose -- Create a MIME message based on given envelope and body sections
imap_mail_copy -- Copy specified messages to a mailbox
imap_mail_move -- Move specified messages to a mailbox
imap_mail -- Send an email message
imap_mailboxmsginfo -- Get information about the current mailbox
imap_mime_header_decode -- Decode MIME header elements
imap_msgno -- This function returns the message sequence number for the given UID
imap_num_msg -- Gives the number of messages in the current mailbox
imap_num_recent -- Gives the number of recent messages in current mailbox
imap_open -- Open an IMAP stream to a mailbox
imap_ping -- Check if the IMAP stream is still active
imap_qprint -- Convert a quoted-printable string to an 8 bit string
imap_renamemailbox -- Rename an old mailbox to new mailbox
imap_reopen -- Reopen IMAP stream to new mailbox
imap_rfc822_parse_adrlist -- Parses an address string
imap_rfc822_parse_headers -- Parse mail headers from a string
imap_rfc822_write_address -- Returns a properly formatted email address given the mailbox, host, and personal info.
imap_scanmailbox -- Read the list of mailboxes, takes a string to search for in the text of the mailbox
imap_search -- This function returns an array of messages matching the given search criteria
imap_set_quota -- Sets a quota for a given mailbox
imap_setacl -- Sets the ACL for a giving mailbox
imap_setflag_full -- Sets flags on messages
imap_sort -- Sort an array of message headers
imap_status -- This function returns status information on a mailbox other than the current one
imap_subscribe -- Subscribe to a mailbox
imap_thread -- Return threaded by REFERENCES tree
imap_uid -- This function returns the UID for the given message sequence number
imap_undelete -- Unmark the message which is marked deleted
imap_unsubscribe -- Unsubscribe from a mailbox
imap_utf7_decode -- Decodes a modified UTF-7 encoded string.
imap_utf7_encode -- Converts 8bit data to modified UTF-7 text.
imap_utf8 -- Converts text to UTF8

imap_8bit

(PHP 3, PHP 4)

imap_8bit -- Convert an 8bit string to a quoted-printable string

Description

string **imap_8bit** (string string)

Convert an 8bit string to a quoted-printable string (according to RFC2045, section 6.7).

Returns a quoted-printable string.

See also **imap_qprint()**.

imap_alerts

(PHP 3>= 3.0.12, PHP 4)

imap_alerts -- This function returns all IMAP alert messages (if any) that have occurred during this page request or since the alert stack was reset

Description

array **imap_alerts** (void)

This function returns an array of all of the IMAP alert messages generated since the last **imap_alerts()** call, or the beginning of the page. When **imap_alerts()** is called, the alert stack is subsequently cleared. The IMAP specification requires that these messages be passed to the user.

imap_append

(PHP 3, PHP 4)

imap_append -- Append a string message to a specified mailbox

Description

bool **imap_append** (resource imap_stream, string mbox, string message [, string options])

Returns **TRUE** on sucess, **FALSE** on error.

imap_append() appends a string message to the specified mailbox *mbox*. If the optional *options* is specified, writes the *options* to that mailbox also.

When talking to the Cyrus IMAP server, you must use "\r\n" as your end-of-line terminator instead of "\n" or the operation will fail.

Example 1. imap_append() example

```
$stream = imap_open("{your.imap.host}INBOX.Drafts","username", "password");

$check = imap_check($stream);
print "Msg Count before append: ". $check->Nmsgs."\n";

imap_append($stream,"{your.imap.host}INBOX.Drafts"
                   ,"From: me@my.host\r\n"
                   ."To: you@your.host\r\n"
                   ."Subject: test\r\n"
                   ."\r\n"
                   ."this is a test message, please ignore\r\n"
                   );

$check = imap_check($stream);
print "Msg Count after append : ". $check->Nmsgs."\n";

imap_close($stream);
```

imap_base64

(PHP 3, PHP 4)

imap_base64 -- Decode BASE64 encoded text

Description

string **imap_base64** (string text)

imap_base64() function decodes BASE-64 encoded text (see RFC2045, Section 6.8). The decoded message is returned as a string.

See also **imap_binary()**.

imap_binary

(PHP 3>= 3.0.2, PHP 4)

imap_binary -- Convert an 8bit string to a base64 string

Description

string **imap_binary** (string string)

Convert an 8bit string to a base64 string (according to RFC2045, Section 6.8).

Returns a base64 string.

See also **imap_base64()**.

imap_body

(PHP 3, PHP 4)

imap_body -- Read the message body

Description

string **imap_body** (resource imap_stream, int msg_number [, int options])

imap_body() returns the body of the message, numbered *msg_number* in the current mailbox. The optional *flags* are a bit mask with one or more of the following:

- FT_UID - The *msgno* is a UID
- FT_PEEK - Do not set the \Seen flag if not already set
- FT_INTERNAL - The return string is in internal format, will not canonicalize to CRLF.

imap_body() will only return a verbatim copy of the message body. To extract single parts of a multipart MIME-encoded message you have to use **imap_fetchstructure()** to analyze its structure and **imap_fetchbody()** to extract a copy of a single body component.

imap_bodystruct

(PHP 3>= 3.0.4, PHP 4)

imap_bodystruct -- Read the structure of a specified body section of a specific message

Description

object **imap_bodystruct** (resource stream_id, int msg_no, int section)

Warning
This function is currently not documented; only the argument list is available.

imap_check

(PHP 3, PHP 4)

imap_check -- Check current mailbox

Description

object **imap_check** (resource imap_stream)

Returns information about the current mailbox. Returns **FALSE** on failure.

The **imap_check()** function checks the current mailbox status on the server and returns the information in an object with following properties:

- Date - last change of mailbox contents
- Driver - protocol used to access this mailbox: POP3, IMAP, NNTP
- Mailbox - the mailbox name
- Nmsgs - number of messages in the mailbox
- Recent - number of recent messages in the mailbox

imap_clearflag_full

(PHP 3>= 3.0.3, PHP 4)

imap_clearflag_full -- Clears flags on messages

Description

bool **imap_clearflag_full** (resource stream, string sequence, string flag, string options)

This function causes a store to delete the specified flag to the flags set for the messages in the specified sequence. The flags which you can unset are "\\Seen", "\\Answered", "\\Flagged", "\\Deleted", and "\\Draft" (as defined by RFC2060).

The options are a bit mask with one or more of the following:

ST_UID The sequence argument contains UIDs instead of sequence numbers

imap_close

(PHP 3, PHP 4)

imap_close -- Close an IMAP stream

Description

bool **imap_close** (resource imap_stream [, int options])

Close the imap stream. Takes an optional *flag* CL_EXPUNGE, which will silently expunge the mailbox before closing, removing all messages marked for deletion.

imap_createmailbox

(PHP 3, PHP 4)

imap_createmailbox -- Create a new mailbox

Description

bool **imap_createmailbox** (resource imap_stream, string mbox)

imap_createmailbox() creates a new mailbox specified by *mbox*. Names containing international characters should be encoded by imap_utf7_encode()

Returns **TRUE** on success and **FALSE** on error.

See also imap_renamemailbox(), imap_deletemailbox() and imap_open() for the format of *mbox* names.

Example 1. imap_createmailbox() example

```
$mbox = imap_open("{your.imap.host}","username","password",OP_HALFOPEN)
     or die("can't connect: ".imap_last_error());

$name1 = "phpnewbox";
$name2 = imap_utf7_encode("phpnewb&ouml;x");

$newname = $name1;

echo "Newname will be '$name1'<br>\n";

# we will now create a new mailbox "phptestbox" in your inbox folder,
# check its status after creation and finaly remove it to restore
# your inbox to its initial state
if(@imap_createmailbox($mbox,imap_utf7_encode("{your.imap.host}INBOX.$newname"))) {
  $status = @imap_status($mbox,"{your.imap.host}INBOX.$newname",SA_ALL);
  if($status) {
    print("your new mailbox '$name1' has the following status:<br>\n");
    print("Messages:   ". $status->messages   ."<br>\n";
    print("Recent:     ". $status->recent     ."<br>\n";
    print("Unseen:     ". $status->unseen     ."<br>\n";
    print("UIDnext:    ". $status->uidnext    ."<br>\n";
    print("UIDvalidity:". $status->uidvalidity)."<br>\n";

    if(imap_renamemailbox($mbox,"{your.imap.host}INBOX.$newname","{your.imap.host}INBOX.$name2")) {
      echo "renamed new mailbox from '$name1' to '$name2'<br>\n";
      $newname=$name2;
    } else {
      print "imap_renamemailbox on new mailbox failed: ".imap_last_error()."<br>\n";
    }
  } else {
    print  "imap_status on new mailbox failed: ".imap_last_error()."<br>\n";
  }
  if(@imap_deletemailbox($mbox,"{your.imap.host}INBOX.$newname")) {
    print "new mailbox removed to restore initial state<br>\n";
  } else {
    print  "imap_deletemailbox on new mailbox failed: ".implode("<br>\n",imap_errors())."<br>\n";
  }

} else {
  print  "could not create new mailbox: ".implode("<br>\n",imap_errors())."<br>\n";
}

imap_close($mbox);
```

imap_delete

(PHP 3, PHP 4)

imap_delete -- Mark a messge for deletion from current mailbox

Description

bool **imap_delete** (int imap_stream, int msg_number [, int options])

Returns **TRUE**.

imap_delete() marks messages listed in *msg_number* for deletion. The optional *flags* parameter only has a single option, *FT_UID*, which tells the function to treat the *msg_number* argument as a *UID*. Messages marked for deletion will stay in the mailbox until either imap_expunge() is called or imap_close() is called with the optional parameter CL_EXPUNGE.

> **Note:** POP3 mailboxes do not have their message flags saved between connections, so imap_expunge() must be called during the same connection in order for messages marked for deletion to actually be purged.

Example 1. imap_delete() Beispiel

```
$mbox = imap_open ("{your.imap.host}INBOX", "username", "password")
    or die ("can't connect: " . imap_last_error());

$check = imap_mailboxmsginfo ($mbox);
print "Messages before delete: " . $check->Nmsgs . "<br>\n" ;
imap_delete ($mbox, 1);
$check = imap_mailboxmsginfo ($mbox);
print "Messages after  delete: " . $check->Nmsgs . "<br>\n" ;
imap_expunge ($mbox);
$check = imap_mailboxmsginfo ($mbox);
print "Messages after expunge: " . $check->Nmsgs . "<br>\n" ;
imap_close ($mbox);
```

imap_deletemailbox

(PHP 3, PHP 4)

imap_deletemailbox -- Delete a mailbox

Description

bool **imap_deletemailbox** (resource imap_stream, string mbox)

imap_deletemailbox() deletes the specified mailbox (see **imap_open()** for the format of *mbox* names).

Returns **TRUE** on success and **FALSE** on error.

See also **imap_createmailbox()**, **imap_renamemailbox()**, and **imap_open()** for the format of *mbox*.

imap_errors

(PHP 3>= 3.0.12, PHP 4)

imap_errors -- This function returns all of the IMAP errors (if any) that have occurred during this page request or since the error stack was reset.

Description

array **imap_errors** (void)

This function returns an array of all of the IMAP error messages generated since the last **imap_errors()** call, or the beginning of the page. When **imap_errors()** is called, the error stack is subsequently cleared.

imap_expunge

(PHP 3, PHP 4)

imap_expunge -- Delete all messages marked for deletion

Description

bool **imap_expunge** (resource imap_stream)

imap_expunge() deletes all the messages marked for deletion by **imap_delete()**, **imap_mail_move()**, or **imap_setflag_full()**.

Returns **TRUE**.

imap_fetch_overview

(PHP 3>= 3.0.4, PHP 4)

imap_fetch_overview -- Read an overview of the information in the headers of the given message

Description

array **imap_fetch_overview** (resource imap_stream, string sequence [, int options])

This function fetches mail headers for the given *sequence* and returns an overview of their contents. *sequence* will contain a sequence of message indices or UIDs, if *flags* contains FT_UID. The returned value is an array of objects describing one message header each:

- subject - the messages subject
- from - who sent it
- date - when was it sent
- message_id - Message-ID
- references - is a reference to this message id
- size - size in bytes
- uid - UID the message has in the mailbox
- msgno - message sequence number in the maibox
- recent - this message is flagged as recent
- flagged - this message is flagged
- answered - this message is flagged as answered
- deleted - this message is flagged for deletion
- seen - this message is flagged as already read
- draft - this message is flagged as being a draft

Example 1. imap_fetch_overview() example

```
$mbox = imap_open("{your.imap.host:143}","username","password")
     or die("can't connect: ".imap_last_error());

$overview = imap_fetch_overview($mbox,"2,4:6",0);

if(is_array($overview)) {
        reset($overview);
        while(  list($key,$val) = each($overview)) {
                print     $val->msgno
                . " - " . $val->date
                . " - " . $val->subject
                . "\n";
        }
}

imap_close($mbox);
```

imap_fetchbody

(PHP 3, PHP 4)

imap_fetchbody -- Fetch a particular section of the body of the message

Description

string **imap_fetchbody** (resource imap_stream, int msg_number, string part_number [, flags options])

This function causes a fetch of a particular section of the body of the specified messages as a text string and returns that text string. The section specification is a string of integers delimited by period which index into a body part list as per the IMAP4 specification. Body parts are not decoded by this function.

The options for **imap_fetchbody()** is a bitmask with one or more of the following:

- FT_UID - The *msg_number* is a UID

- FT_PEEK - Do not set the \Seen flag if not already set
- FT_INTERNAL - The return string is in "internal" format, without any attempt to canonicalize CRLF.

See also: **imap_fetchstructure()**.

imap_fetchheader

(PHP 3>= 3.0.3, PHP 4)

imap_fetchheader -- Returns header for a message

Description

string **imap_fetchheader** (resource imap_stream, int msgno, int options)

This function causes a fetch of the complete, unfiltered RFC2822 format header of the specified message as a text string and returns that text string.

The options are:

FT_UID The msgno argument is a UID
FT_INTERNAL The return string is in "internal" format, without any attempt to canonicalize to CRLF newlines
FT_PREFETCHTEXT The RFC822.TEXT should be pre-fetched at the same time. This avoids an extra RTT on an IMAP connection if a full message text is desired (e.g. in a "save to local file" operation)

imap_fetchstructure

(PHP 3, PHP 4)

imap_fetchstructure -- Read the structure of a particular message

Description

object **imap_fetchstructure** (resource imap_stream, int msg_number [, int options])

This function fetches all the structured information for a given message. The optional *flags* parameter only has a single option, *FT_UID*, which tells the function to treat the *msg_number* argument as a *UID*. The returned object includes the envelope, internal date, size, flags and body structure along with a similar object for each mime attachement. The structure of the returned objects is as follows:

Table 1. Returned Objects for imap_fetchstructure()

type	Primary body type
encoding	Body transfer encoding
ifsubtype	**TRUE** if there is a subtype string
subtype	MIME subtype
ifdescription	**TRUE** if there is a description string
description	Content description string
ifid	**TRUE** if there is an identification string
id	Identification string
lines	Number of lines
bytes	Number of bytes
ifdisposition	**TRUE** if there is a disposition string

disposition	Disposition string
ifdparameters	**TRUE** if the dparameters array exists
dparameters	An array of objects where each object has an "attribute" and a "value" property corresponding to the parameters on the Content-disposition MIMEheader.
ifparameters	**TRUE** if the parameters array exists
parameters	An array of objects where each object has an "attribute" and a "value" property.
parts	An array of objects identical in structure to the top-level object, each of which corresponds to a MIME body part.

Table 2. Primary body type

0	text
1	multipart
2	message
3	application
4	audio
5	image
6	video
7	other

Table 3. Transfer encodings

0	7BIT
1	8BIT
2	BINARY
3	BASE64
4	QUOTED-PRINTABLE
5	OTHER

See also: **imap_fetchbody()**.

imap_get_quota

(PHP 4 >= 4.0.5)

imap_get_quota -- Retrieve the quota level settings, and usage statics per mailbox

Description

array **imap_get_quota** (resource imap_stream, string quota_root)

Returns an array with integer values limit and usage for the given mailbox. The value of limit represents the total amount of space allowed for this mailbox. The usage value represents the mailboxes current level of capacity. Will return **FALSE** in the case of failure.

This function is currently only available to users of the c-client2000 or greater library.

NOTE: For this function to work, the mail stream is required to be opened as the mail-admin user. For a non-admin user version of this function, please see the **imap_get_quotaroot()** function of PHP.

imap_stream should be the value returned from an **imap_open()** call. NOTE: This stream is required to be opened as the mail admin user for the get_quota function to work. *quota_root* should normally be in the form of user.name where name is the mailbox you wish to retrieve information about.

Example 1. imap_get_quota() example

```
$mbox = imap_open("{your.imap.host}","mailadmin","password",OP_HALFOPEN)
      or die("can't connect: ".imap_last_error());

$quota_value = imap_get_quota($mbox, "user.kalowsky");
if(is_array($quota_value)) {
    print "Usage level is: " . $quota_value['usage'];
```

```
    print "Limit level is: " . $quota_value['limit'];
}

imap_close($mbox);
```

As of PHP version 4.3, the function more properly reflects the functionality as dictated by the RFC 2087. The array return value has changed to support an unlimited number of returned resources (i.e. messages, or sub-folders) with each named resource receiving an individual array key. Each key value then contains an another array with the usage and limit values within it. The example below shows the updated returned output.

For backwards compatibility reasons, the originial access methods are still available for use, although it is suggested to update.

Example 2. imap_get_quota() 4.3 or greater example

```
$mbox = imap_open("{your.imap.host}","mailadmin","password",OP_HALFOPEN)
      or die("can't connect: ".imap_last_error());

$quota_values = imap_get_quota($mbox, "user.kalowsky");
if(is_array($quota_values)) {
   $storage = $quota_values['STORAGE'];
   print "STORAGE usage level is: " .  $storage['usage'];
   print "STORAGE limit level is: " .  $storage['limit'];

   $message = $quota_values['MESSAGE'];
   print "MESSAGE usage level is: " .  $message['usage'];
   print "MESSAGE usage level is: " .  $message['limit'];

   /* ...  */
}

imap_close($mbox);
```

See also imap_open(), imap_set_quota(), imap_get_quotaroot().

imap_get_quotaroot

(PHP 4 >= 4.3.0)

imap_get_quotaroot -- Retrieve the quota settings per user

Description

array **imap_get_quotaroot** (resource imap_stream, string quota_root)

Returns an array of integer values pertaining to the specified user mailbox. All values contain a key based upon the resource name, and a corresponding array with the usage and limit values within.

The limit value represents the total amount of space allowed for this user's total mailbox usage. The usage value represents the user's current total mailbox capacity. This function will return **FALSE** in the case of call failure, and an array of information about the connection upon an un-parsable response from the server.

This function is currently only available to users of the c-client2000 or greater library.

imap_stream should be the value returned from an imap_open() call. This stream should be opened as the user whose mailbox you wish to check. *quota_root* should normally be in the form of which mailbox (i.e. INBOX).

Example 1. imap_get_quotaroot() example

```
$mbox = imap_open("{your.imap.host}","kalowsky","password",OP_HALFOPEN)
      or die("can't connect: ".imap_last_error());

$quota = imap_get_quotaroot($mbox, "INBOX");
if(is_array($quota)) {
   $storage = $quota_values['STORAGE'];
   print "STORAGE usage level is: " .  $storage['usage'];
   print "STORAGE limit level is: " .  $storage['limit'];

   $message = $quota_values['MESSAGE'];
   print "MESSAGE usage level is: " .  $message['usage'];
   print "MESSAGE usage level is: " .  $message['limit'];

   /* ...  */

}

imap_close($mbox);
```

See also **imap_open()**, **imap_set_quota()**, **imap_get_quota()**.

imap_getmailboxes

(PHP 3>= 3.0.12, PHP 4)

imap_getmailboxes -- Read the list of mailboxes, returning detailed information on each one

Description

array **imap_getmailboxes** (resource imap_stream, string ref, string pattern)

Returns an array of objects containing mailbox information. Each object has the attributes *name*, specifying the full name of the mailbox; *delimiter*, which is the hierarchy delimiter for the part of the hierarchy this mailbox is in; and *attributes*. *Attributes* is a bitmask that can be tested against:

- LATT_NOINFERIORS - This mailbox has no "children" (there are no mailboxes below this one).
- LATT_NOSELECT - This is only a container, not a mailbox - you cannot open it.
- LATT_MARKED - This mailbox is marked. Only used by UW-IMAPD.
- LATT_UNMARKED - This mailbox is not marked. Only used by UW-IMAPD.

Mailbox names containing international Characters outside the printable ASCII range will be encoded and may be decoded by **imap_utf7_decode()**.

ref should normally be just the server specification as described in **imap_open()**, and *pattern* specifies where in the mailbox hierarchy to start searching. If you want all mailboxes, pass '*' for *pattern*.

There are two special characters you can pass as part of the *pattern*: '*' and '%'. '*' means to return all mailboxes. If you pass *pattern* as '*', you will get a list of the entire mailbox hierarchy. '%' means to return the current level only. '%' as the *pattern* parameter will return only the top level mailboxes; '~/mail/%' on UW_IMAPD will return every mailbox in the ~/mail directory, but none in subfolders of that directory.

Example 1. imap_getmailboxes() example

```
$mbox = imap_open("{your.imap.host}","username","password",OP_HALFOPEN)
      or die("can't connect: ".imap_last_error());

$list = imap_getmailboxes($mbox,"{your.imap.host}","*");
if(is_array($list)) {
  reset($list);
  while (list($key, $val) = each($list))
  {
    print "($key) ";
    print imap_utf7_decode($val->name).",";
    print "'".$val->delimiter."',";
    print $val->attributes."<br>\n";
  }
} else
  print "imap_getmailboxes failed: ".imap_last_error()."\n";

imap_close($mbox);
```

See also **imap_getsubscribed()**.

imap_getsubscribed

(PHP 3>= 3.0.12, PHP 4)

imap_getsubscribed -- List all the subscribed mailboxes

Description

array **imap_getsubscribed** (resource imap_stream, string ref, string pattern)

This function is identical to **imap_getmailboxes()**, except that it only returns mailboxes that the user is subscribed to.

imap_header

(PHP 3, PHP 4)

imap_header -- Read the header of the message

Description

object **imap_header** (resource imap_stream, int msg_number [, int fromlength [, int subjectlength [, string defaulthost]]])

This function is an alias to **imap_headerinfo()** and is identical to it in every way.

imap_headerinfo

(PHP 3, PHP 4)

imap_headerinfo -- Read the header of the message

Description

object **imap_headerinfo** (resource imap_stream, int msg_number [, int fromlength [, int subjectlength [, string defaulthost]]])

This function returns an object of various header elements.

```
    remail, date, Date, subject, Subject, in_reply_to, message_id,
    newsgroups, followup_to, references

message flags:
  Recent -  'R' if recent and seen,
            'N' if recent and not seen,
            ' ' if not recent
  Unseen -  'U' if not seen AND not recent,
            ' ' if seen OR not seen and recent
  Answered -'A' if answered,
            ' ' if unanswered
  Deleted - 'D' if deleted,
            ' ' if not deleted
  Draft -   'X' if draft,
            ' ' if not draft
  Flagged - 'F' if flagged,
            ' ' if not flagged

NOTE that the Recent/Unseen behavior is a little odd. If you want to
know if a message is Unseen, you must check for

Unseen == 'U' || Recent == 'N'

toaddress (full to: line, up to 1024 characters)

to[] (returns an array of objects from the To line, containing):
  personal
  adl
  mailbox
  host

fromaddress (full from: line, up to 1024 characters)

from[] (returns an array of objects from the From line, containing):
  personal
  adl
  mailbox
  host

ccaddress (full cc: line, up to 1024 characters)
```

cc[] (returns an array of objects from the Cc line, containing):
  personal
  adl
  mailbox
  host

bccaddress (full bcc line, up to 1024 characters)
bcc[] (returns an array of objects from the Bcc line, containing):
  personal
  adl
  mailbox
  host

reply_toaddress (full reply_to: line, up to 1024 characters)
reply_to[] (returns an array of objects from the Reply_to line,
containing):
  personal
  adl
  mailbox
  host

senderaddress (full sender: line, up to 1024 characters)
sender[] (returns an array of objects from the sender line, containing):
  personal
  adl
  mailbox
  host

return_path (full return-path: line, up to 1024 characters)
return_path[] (returns an array of objects from the return_path line,
containing):
  personal
  adl
  mailbox
  host

udate (mail message date in unix time)

fetchfrom (from line formatted to fit *fromlength*
characters)

fetchsubject (subject line formatted to fit *subjectlength* characters)

imap_headers

(PHP 3, PHP 4)

imap_headers -- Returns headers for all messages in a mailbox

Description

array **imap_headers** (resource imap_stream)

Returns an array of string formatted with header info. One element per mail message.

imap_last_error

(PHP 3>= 3.0.12, PHP 4)

imap_last_error -- This function returns the last IMAP error (if any) that occurred during this page request

Description

string **imap_last_error** (void)

This function returns the full text of the last IMAP error message that occurred on the current page. The error stack is untouched; calling **imap_last_error()** subsequently, with no intervening errors, will return the same error.

imap_list

(PHP 3>= 3.0.4, PHP 4)

imap_list -- Read the list of mailboxes

Description

array **imap_list** (resource imap_stream, string ref, string pattern)

Returns an array containing the names of the mailboxes. See **imap_getmailboxes()** for a description of *ref* and *pattern*.

Example 1. imap_list() example

```
$mbox = imap_open("{your.imap.host}","username","password",OP_HALFOPEN)
      or die("can't connect: ".imap_last_error());

$list = imap_list($mbox,"{your.imap.host}","*");
if(is_array($list)) {
  reset($list);
  while (list($key, $val) = each($list))
    print imap_utf7_decode($val)."<br>\n";
} else
  print "imap_list failed: ".imap_last_error()."\n";

imap_close($mbox);
```

imap_listmailbox

(PHP 3, PHP 4)

imap_listmailbox -- Read the list of mailboxes

Description

array **imap_listmailbox** (resource imap_stream, string ref, string pattern)

This function is an alias to **imap_list()** and is identical to it in every way.

imap_listscan

(no version information, might be only in CVS)

imap_listscan -- Read the list of mailboxes, takes a string to search for in the text of the mailbox

Description

array **imap_listscan** (resource imap_stream, string ref, string pattern, string content)

Returns an array containing the names of the mailboxes that have *content* in the text of the mailbox. This function is similar to **imap_listmailbox()**, but it will additionally check for the presence of the string *content* inside the mailbox data. See **imap_getmailboxes()** for a description of *ref* and *pattern*.

imap_listsubscribed

(PHP 3, PHP 4)

imap_listsubscribed -- List all the subscribed mailboxes

Description

array **imap_listsubscribed** (resource imap_stream, string ref, string pattern)

This function is an alias to **imap_lsub()** and is identical to it in every way.

imap_lsub

(PHP 3>= 3.0.4, PHP 4)

imap_lsub -- List all the subscribed mailboxes

Description

array **imap_lsub** (resource imap_stream, string ref, string pattern)

Returns an array of all the mailboxes that you have subscribed. This is almost identical to **imap_listmailbox()**, but will only return mailboxes the user you logged in as has subscribed.

imap_mail_compose

(PHP 3>= 3.0.5, PHP 4)

imap_mail_compose -- Create a MIME message based on given envelope and body sections

Description

string **imap_mail_compose** (array envelope, array body)

Example 1. imap_mail_compose() example

```
<?php

$envelope["from"]="musone@afterfive.com";
$envelope["to"]="musone@darkstar";
$envelope["cc"]="musone@edgeglobal.com";

$part1["type"]=TYPEMULTIPART;
$part1["subtype"]="mixed";

$filename="/tmp/imap.c.gz";
$fp=fopen($filename,"r");
$contents=fread($fp,filesize($filename));
fclose($fp);

$part2["type"]=TYPEAPPLICATION;
$part2["encoding"]=ENCBINARY;
$part2["subtype"]="octet-stream";
$part2["description"]=basename($filename);
$part2["contents.data"]=$contents;

$part3["type"]=TYPETEXT;
$part3["subtype"]="plain";
$part3["description"]="description3";
$part3["contents.data"]="contents.data3\n\n\n\t";

$body[1]=$part1;
$body[2]=$part2;
$body[3]=$part3;

echo nl2br(imap_mail_compose($envelope,$body));

?>
```

imap_mail_copy

(PHP 3, PHP 4)

imap_mail_copy -- Copy specified messages to a mailbox

Description

bool **imap_mail_copy** (resource imap_stream, string msglist, string mbox [, int options])

Returns **TRUE** on success and **FALSE** on error.

Copies mail messages specified by *msglist* to specified mailbox. *msglist* is a range not just message numbers (as described in RFC2060).

Flags is a bitmask of one or more of

- CP_UID - the sequence numbers contain UIDS
- CP_MOVE - Delete the messages from the current mailbox after copying

imap_mail_move

(PHP 3, PHP 4)

imap_mail_move -- Move specified messages to a mailbox

Description

bool **imap_mail_move** (resource imap_stream, string msglist, string mbox [, int options])

Returns **TRUE** on success and **FALSE** on error.

Moves mail messages specified by *msglist* to specified mailbox. *msglist* is a range not just message numbers (as described in RFC2060).

Flags is a bitmask and may contain the single option

- CP_UID - the sequence numbers contain UIDS

imap_mail

(PHP 3>= 3.0.14, PHP 4)

imap_mail -- Send an email message

Description

string **imap_mail** (string to, string subject, string message [, string additional_headers [, string cc [, string bcc [, string rpath]]]])

This function allows sending of emails with correct handling of Cc and Bcc receivers. The parameters to, cc and bcc are all strings and are all parsed as rfc822 address lists. The receivers specified in bcc will get the mail, but are excluded from the headers. Use the rpath parameter to specify return path. This is useful when using php as a mail client for multiple users.

imap_mailboxmsginfo

(PHP 3>= 3.0.2, PHP 4)

imap_mailboxmsginfo -- Get information about the current mailbox

Description

object **imap_mailboxmsginfo** (resource imap_stream)

Returns information about the current mailbox. Returns **FALSE** on failure.

The **imap_mailboxmsginfo()** function checks the current mailbox status on the server. It is similar to **imap_status()**, but will additionally sum up the size of all messages in the mailbox, which will take some additional time to execute. It returns the

information in an object with following properties.

Table 1. Mailbox properties

Date	date of last change
Driver	driver
Mailbox	name of the mailbox
Nmsgs	number of messages
Recent	number of recent messages
Unread	number of unread messages
Deleted	number of deleted messages
Size	mailbox size

Example 1. imap_mailboxmsginfo() example

```
<?php

$mbox = imap_open("{your.imap.host}INBOX","username", "password")
      or die("can't connect: ".imap_last_error());

$check = imap_mailboxmsginfo($mbox);

if($check) {
    print "Date: "    . $check->Date    ."<br>\n" ;
    print "Driver: "  . $check->Driver  ."<br>\n" ;
    print "Mailbox: " . $check->Mailbox ."<br>\n" ;
    print "Messages: ". $check->Nmsgs   ."<br>\n" ;
    print "Recent: "  . $check->Recent  ."<br>\n" ;
    print "Unread: "  . $check->Unread  ."<br>\n" ;
    print "Deleted: " . $check->Deleted ."<br>\n" ;
    print "Size: "    . $check->Size    ."<br>\n" ;
} else {
    print "imap_check() failed: ".imap_last_error(). "<br>\n";
}

imap_close($mbox);

?>
```

imap_mime_header_decode

(PHP 3>= 3.0.17, PHP 4)

imap_mime_header_decode -- Decode MIME header elements

Description

array **imap_mime_header_decode** (string text)

imap_mime_header_decode() function decodes MIME message header extensions that are non ASCII text (see RFC2047) The decoded elements are returned in an array of objects, where each object has two properties, "charset" & "text". If the element hasn't been encoded, and in other words is in plain US-ASCII,the "charset" property of that element is set to "default".

Example 1. imap_mime_header_decode() example

```
$text="=?ISO-8859-1?Q?Keld_J=F8rn_Simonsen?= <keld@dkuug.dk>";

$elements=imap_mime_header_decode($text);
for($i=0;$i<count($elements);$i++) {
    echo "Charset: {$elements[$i]->charset}\n";
    echo "Text: {$elements[$i]->text}\n\n";
}
```

In the above example we would have two elements, whereas the first element had previously been encoded with ISO-8859-1, and the second element would be plain US-ASCII.

imap_msgno

(PHP 3>= 3.0.3, PHP 4)

imap_msgno -- This function returns the message sequence number for the given UID

Description

int **imap_msgno** (resource imap_stream, int uid)

This function returns the message sequence number for the given UID. It is the inverse of **imap_uid()**.

imap_num_msg

(PHP 3, PHP 4)

imap_num_msg -- Gives the number of messages in the current mailbox

Description

int **imap_num_msg** (resource imap_stream)

Return the number of messages in the current mailbox.

See also: **imap_num_recent()** and **imap_status()**.

imap_num_recent

(PHP 3, PHP 4)

imap_num_recent -- Gives the number of recent messages in current mailbox

Description

int **imap_num_recent** (resource imap_stream)

Returns the number of recent messages in the current mailbox.

See also: **imap_num_msg()** and **imap_status()**.

imap_open

(PHP 3, PHP 4)

imap_open -- Open an IMAP stream to a mailbox

Description

resource **imap_open** (string mailbox, string username, string password [, int options])

Returns an IMAP stream on success and **FALSE** on error. This function can also be used to open streams to POP3 and NNTP servers, but some functions and features are only available on IMAP servers.

A mailbox name consists of a server part and a mailbox path on this server. The special name INBOX stands for the current users personal mailbox. The server part, which is enclosed in '{' and '}', consists of the servers name or ip address, an optional port (prefixed by ':'), and an optional protocol specification (prefixed by '/'). The server part is mandatory in all mailbox parameters. Mailbox names that contain international characters besides those in the printable ASCII space have to be encoded with **imap_utf7_encode()**.

The options are a bit mask with one or more of the following:

- OP_READONLY - Open mailbox read-only
- OP_ANONYMOUS - Dont use or update a `.newsrc` for news (NNTP only)
- OP_HALFOPEN - For IMAP and NNTP names, open a connection but dont open a mailbox

- CL_EXPUNGE - Expunge mailbox automatically upon mailbox close

To connect to an IMAP server running on port 143 on the local machine, do the following:

```
$mbox = imap_open ("{localhost:143}INBOX", "user_id", "password");
```

To connect to a POP3 server on port 110 on the local server, use:

```
$mbox = imap_open ("{localhost:110/pop3}INBOX", "user_id", "password");
```

To connect to an SSL IMAP or POP3 server, add /ssl after the protocol specification:

```
$mbox = imap_open ("{localhost:993/imap/ssl}INBOX", "user_id", "password");
```

To connect to an SSL IMAP or POP3 server with a self-signed certificate, add /ssl/novalidate-cert after the protocol specification:

```
$mbox = imap_open ("{localhost:995/pop3/ssl/novalidate-cert}", "user_id", "password");
```

To connect to an NNTP server on port 119 on the local server, use:

```
$nntp = imap_open ("{localhost:119/nntp}comp.test", "", "");
```

To connect to a remote server replace "localhost" with the name or the IP address of the server you want to connect to.

Example 1. imap_open() example

```
$mbox = imap_open ("{your.imap.host:143}", "username", "password");

echo "<p><h1>Mailboxes</h1>\n";
$folders = imap_listmailbox ($mbox, "{your.imap.host:143}", "*");

if ($folders == false) {
    echo "Call failed<br>\n";
} else {
    while (list ($key, $val) = each ($folders)) {
        echo $val."<br>\n";
    }
}

echo "<p><h1>Headers in INBOX</h1>\n";
$headers = imap_headers ($mbox);

if ($headers == false) {
    echo "Call failed<br>\n";
} else {
    while (list ($key,$val) = each ($headers)) {
        echo $val."<br>\n";
    }
}

imap_close($mbox);
```

imap_ping

(PHP 3, PHP 4)

imap_ping -- Check if the IMAP stream is still active

Description

bool **imap_ping** (resource imap_stream)

Returns **TRUE** if the stream is still alive, **FALSE** otherwise.

imap_ping() function pings the stream to see it is still active. It may discover new mail; this is the preferred method for a periodic "new mail check" as well as a "keep alive" for servers which have inactivity timeout. (As PHP scripts do not tend to run that long, i can hardly imagine that this function will be usefull to anyone.)

imap_qprint

(PHP 3, PHP 4)

imap_qprint -- Convert a quoted-printable string to an 8 bit string

Description

string **imap_qprint** (string string)

Convert a quoted-printable string to an 8 bit string (according to RFC2045, section 6.7).

Returns an 8 bit (binary) string.

See also **imap_8bit()**.

imap_renamemailbox

(PHP 3, PHP 4)

imap_renamemailbox -- Rename an old mailbox to new mailbox

Description

bool **imap_renamemailbox** (resource imap_stream, string old_mbox, string new_mbox)

This function renames on old mailbox to new mailbox (see **imap_open()** for the format of *mbox* names).

Returns **TRUE** on success and **FALSE** on error.

See also **imap_createmailbox()**, **imap_deletemailbox()**, and **imap_open()** for the format of *mbox*.

imap_reopen

(PHP 3, PHP 4)

imap_reopen -- Reopen IMAP stream to new mailbox

Description

bool **imap_reopen** (resource imap_stream, string mailbox [, string options])

This function reopens the specified stream to a new mailbox on an IMAP or NNTP server.

The options are a bit mask with one or more of the following:

- OP_READONLY - Open mailbox read-only
- OP_ANONYMOUS - Dont use or update a `.newsrc` for news (NNTP only)
- OP_HALFOPEN - For IMAP and NNTP names, open a connection but dont open a mailbox.
- CL_EXPUNGE - Expunge mailbox automatically upon mailbox close (see also **imap_delete()** and **imap_expunge()**)

Returns **TRUE** on success and **FALSE** on error.

imap_rfc822_parse_adrlist

(PHP 3>= 3.0.2, PHP 4)

imap_rfc822_parse_adrlist -- Parses an address string

Description

array **imap_rfc822_parse_adrlist** (string address, string default_host)

This function parses the address string as defined in RFC2822 and for each address, returns an array of objects. The objects

properties are:

- mailbox - the mailbox name (username)
- host - the host name
- personal - the personal name
- adl - at domain source route

Example 1. imap_rfc822_parse_adrlist() example

```
$address_string = "Hartmut Holzgraefe <hartmut@cvs.php.net>, postmaster@somedomain.net, root";
$address_array  = imap_rfc822_parse_adrlist($address_string,"somedomain.net");
if(! is_array($address_array)) die("somethings wrong\n");

reset($address_array);
while(list($key,$val)=each($address_array)){
  print "mailbox : ".$val->mailbox."<br>\n";
  print "host    : ".$val->host."<br>\n";
  print "personal: ".$val->personal."<br>\n";
  print "adl     : ".$val->adl."<p>\n";
}
```

imap_rfc822_parse_headers

(PHP 4)

imap_rfc822_parse_headers -- Parse mail headers from a string

Description

object **imap_rfc822_parse_headers** (string headers [, string defaulthost])

This function returns an object of various header elements, similar to **imap_header()**, except without the flags and other elements that come from the IMAP server.

imap_rfc822_write_address

(PHP 3>= 3.0.2, PHP 4)

imap_rfc822_write_address -- Returns a properly formatted email address given the mailbox, host, and personal info.

Description

string **imap_rfc822_write_address** (string mailbox, string host, string personal)

Returns a properly formatted email address as defined in RFC2822 given the mailbox, host, and personal info.

Example 1. imap_rfc822_write_address() example

```
print imap_rfc822_write_address("hartmut","cvs.php.net","Hartmut Holzgraefe")."\n";
```

imap_scanmailbox

(PHP 3, PHP 4)

imap_scanmailbox -- Read the list of mailboxes, takes a string to search for in the text of the mailbox

Description

array **imap_scanmailbox** (resource imap_stream, string ref, string pattern, string content)

This function is an alias to **imap_listscan()** and is identical to it in every way.

imap_search

(PHP 3>= 3.0.12, PHP 4)

imap_search -- This function returns an array of messages matching the given search criteria

Description

array **imap_search** (resource imap_stream, string criteria, int options)

This function performs a search on the mailbox currently opened in the given imap stream. *criteria* is a string, delimited by spaces, in which the following keywords are allowed. Any multi-word arguments (eg. FROM "joey smith") must be quoted.

- ALL - return all messages matching the rest of the criteria
- ANSWERED - match messages with the \\ANSWERED flag set
- BCC "string" - match messages with "string" in the Bcc: field
- BEFORE "date" - match messages with Date: before "date"
- BODY "string" - match messages with "string" in the body of the message
- CC "string" - match messages with "string" in the Cc: field
- DELETED - match deleted messages
- FLAGGED - match messages with the \\FLAGGED (sometimes referred to as Important or Urgent) flag set
- FROM "string" - match messages with "string" in the From: field
- KEYWORD "string" - match messages with "string" as a keyword
- NEW - match new messages
- OLD - match old messages
- ON "date" - match messages with Date: matching "date"
- RECENT - match messages with the \\RECENT flag set
- SEEN - match messages that have been read (the \\SEEN flag is set)
- SINCE "date" - match messages with Date: after "date"
- SUBJECT "string" - match messages with "string" in the Subject:
- TEXT "string" - match messages with text "string"
- TO "string" - match messages with "string" in the To:
- UNANSWERED - match messages that have not been answered
- UNDELETED - match messages that are not deleted
- UNFLAGGED - match messages that are not flagged
- UNKEYWORD "string" - match messages that do not have the keyword "string"
- UNSEEN - match messages which have not been read yet

For example, to match all unanswered messages sent by Mom, you'd use: "UNANSWERED FROM mom". Searches appear to be case insensitive. This list of criteria is from a reading of the UW c-client source code and may be uncomplete or inaccurate (see also RFC2060, section 6.4.4).

Valid values for flags are SE_UID, which causes the returned array to contain UIDs instead of messages sequence numbers.

imap_set_quota

(PHP 4 >= 4.0.5)

imap_set_quota -- Sets a quota for a given mailbox

Description

bool **imap_set_quota** (resource imap_stream, string quota_root, int quota_limit)

Sets an upper limit quota on a per mailbox basis. This function requires the *imap_stream* to have been opened as the mail administrator account. It will not work if opened as any other user.

This function is currently only available to users of the c-client2000 or greater library.

imap_stream is the stream pointer returned from a **imap_open()** call. This stream must be opened as the mail administrator, other wise this function will fail. *quota_root* is the mailbox to have a quota set. This should follow the IMAP standard format for a mailbox, 'user.name'. *quota_limit* is the maximum size (in KB) for the *quota_root*.

Returns **TRUE** on success and **FALSE** on error.

Example 1. imap_set_quota() example

```
$mbox = imap_open ("{your.imap.host:143}", "mailadmin", "password");

if(!imap_set_quota($mbox, "user.kalowsky", 3000)) {
    print "Error in setting quota\n";
    return;
}

imap_close($mbox);
```

See also **imap_open()**, **imap_set_quota()**.

imap_setacl

(PHP 4 >= 4.1.0)

imap_setacl -- Sets the ACL for a giving mailbox

Description

bool **imap_setacl** (resource stream_id, string mailbox, string id, string rights)

Warning
This function is currently not documented; only the argument list is available.

imap_setflag_full

(PHP 3>= 3.0.3, PHP 4)

imap_setflag_full -- Sets flags on messages

Description

bool **imap_setflag_full** (resource stream, string sequence, string flag, string options)

This function causes a store to add the specified flag to the flags set for the messages in the specified sequence.

The flags which you can set are "\\Seen", "\\Answered", "\\Flagged", "\\Deleted", and "\\Draft" (as defined by RFC2060).

The options are a bit mask with one or more of the following:

ST_UID The sequence argument contains UIDs instead of sequence numbers

Example 1. imap_setflag_full() example

```
$mbox = imap_open("{your.imap.host:143}","username","password")
     or die("can't connect: ".imap_last_error());

$status = imap_setflag_full($mbox,"2,5","\\Seen \\Flagged");

print gettype($status)."\n";
print $status."\n";

imap_close($mbox);
```

imap_sort

(PHP 3>= 3.0.3, PHP 4)

imap_sort -- Sort an array of message headers

Description

array **imap_sort** (resource stream, int criteria, int reverse [, int options [, string search_criteria]])

Returns an array of message numbers sorted by the given parameters.

Reverse is 1 for reverse-sorting.

Criteria can be one (and only one) of the following:

SORTDATE message Date
SORTARRIVAL arrival date
SORTFROM mailbox in first From address
SORTSUBJECT message Subject
SORTTO mailbox in first To address
SORTCC mailbox in first cc address
SORTSIZE size of message in octets

The flags are a bitmask of one or more of the following:

SE_UID Return UIDs instead of sequence numbers
SE_NOPREFETCH Don't prefetch searched messages.

imap_status

(PHP 3>= 3.0.4, PHP 4)

imap_status -- This function returns status information on a mailbox other than the current one

Description

object **imap_status** (resource imap_stream, string mailbox, int options)

This function returns an object containing status information. Valid flags are:

- SA_MESSAGES - set status->messages to the number of messages in the mailbox
- SA_RECENT - set status->recent to the number of recent messages in the mailbox
- SA_UNSEEN - set status->unseen to the number of unseen (new) messages in the mailbox
- SA_UIDNEXT - set status->uidnext to the next uid to be used in the mailbox
- SA_UIDVALIDITY - set status->uidvalidity to a constant that changes when uids for the mailbox may no longer be valid
- SA_ALL - set all of the above

status->flags is also set, which contains a bitmask which can be checked against any of the above constants.

Example 1. imap_status() example

```
$mbox = imap_open("{your.imap.host}","username","password",OP_HALFOPEN)
      or die("can't connect: ".imap_last_error());

$status = imap_status($mbox,"{your.imap.host}INBOX",SA_ALL);
if($status) {
  print("Messages:   ". $status->messages   )."<br>\n";
  print("Recent:     ". $status->recent     )."<br>\n";
  print("Unseen:     ". $status->unseen     )."<br>\n";
  print("UIDnext:    ". $status->uidnext    )."<br>\n";
  print("UIDvalidity:". $status->uidvalidity)."<br>\n";
} else
  print "imap_status failed: ".imap_last_error()."\n";

imap_close($mbox);
```

imap_subscribe

(PHP 3, PHP 4)

imap_subscribe -- Subscribe to a mailbox

Description

bool **imap_subscribe** (resource imap_stream, string mbox)

Subscribe to a new mailbox.

Returns **TRUE** on success and **FALSE** on error.

imap_thread

(PHP 4 >= 4.1.0)

imap_thread -- Return threaded by REFERENCES tree

Description

array **imap_thread** (resource stream_id [, int options])

Warning
This function is currently not documented; only the argument list is available.

imap_uid

(PHP 3>= 3.0.3, PHP 4)

imap_uid -- This function returns the UID for the given message sequence number

Description

int **imap_uid** (resource imap_stream, int msgno)

This function returns the UID for the given message sequence number. An UID is an unique identifier that will not change over time while a message sequence number may change whenever the content of the mailbox changes. This function is the inverse of **imap_msgno()**.

> **Note:** This is not supported by POP3 mailboxes.

imap_undelete

(PHP 3, PHP 4)

imap_undelete -- Unmark the message which is marked deleted

Description

bool **imap_undelete** (resource imap_stream, int msg_number)

This function removes the deletion flag for a specified message, which is set by **imap_delete()** or **imap_mail_move()**.

Returns **TRUE** on success and **FALSE** on error.

imap_unsubscribe

(PHP 3, PHP 4)

imap_unsubscribe -- Unsubscribe from a mailbox

Description

bool **imap_unsubscribe** (string imap_stream, string mbox)

Unsubscribe from a specified mailbox.

Returns **TRUE** on success and **FALSE** on error.

imap_utf7_decode

(PHP 3>= 3.0.15, PHP 4)

imap_utf7_decode -- Decodes a modified UTF-7 encoded string.

Description

string **imap_utf7_decode** (string text)

Decodes modified UTF-7 *text* into 8bit data.

Returns the decoded 8bit data, or **FALSE** if the input string was not valid modified UTF-7. This function is needed to decode mailbox names that contain international characters outside of the printable ASCII range. The modified UTF-7 encoding is defined in RFC 2060, section 5.1.3 (original UTF-7 was defned in RFC1642).

imap_utf7_encode

(PHP 3>= 3.0.15, PHP 4)

imap_utf7_encode -- Converts 8bit data to modified UTF-7 text.

Description

string **imap_utf7_encode** (string data)

Converts 8bit *data* to modified UTF-7 text. This is needed to encode mailbox names that contain international characters outside of the printable ASCII range. The modified UTF-7 encoding is defined in RFC 2060, section 5.1.3 (original UTF-7 was defned in RFC1642).

Returns the modified UTF-7 text.

imap_utf8

(PHP 3>= 3.0.13, PHP 4)

imap_utf8 -- Converts text to UTF8

Description

string **imap_utf8** (string text)

Converts the given *text* to UTF8 (as defined in RFC2044).

XLIII. Informix functions

Introduction

The Informix driver for Informix (IDS) 7.x, SE 7.x, Universal Server (IUS) 9.x and IDS 2000 is implemented in "ifx.ec" and "php3_ifx.h" in the informix extension directory. IDS 7.x support is fairly complete, with full support for BYTE and TEXT columns. IUS 9.x support is partly finished: the new data types are there, but SLOB and CLOB support is still under construction.

Requirements

Configuration notes: You need a version of ESQL/C to compile the PHP Informix driver. ESQL/C versions from 7.2x on should be OK. ESQL/C is now part of the Informix Client SDK.

Make sure that the "INFORMIXDIR" variable has been set, and that $INFORMIXDIR/bin is in your PATH before you run the "configure" script.

Installation

To be able to use the functions defined in this module you must compile your PHP interpreter using the configure line `--with_informix[=DIR]`, where DIR is the Informix base install directory, defaults to nothing.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Note: Make sure that the Informix environment variables INFORMIXDIR and INFORMIXSERVER are available to the PHP ifx driver, and that the INFORMIX bin directory is in the PATH. Check this by running a script that contains a call to **phpinfo()** before you start testing. The **phpinfo()** output should list these environment variables. This is **TRUE** for both CGI php and Apache mod_php. You may have to set these environment variables in your Apache startup script.

The Informix shared libraries should also be available to the loader (check LD_LIBRARY_PATH or ld.so.conf/ldconfig).

Some notes on the use of BLOBs (TEXT and BYTE columns): BLOBs are normally addressed by BLOB identifiers. Select queries return a "blob id" for every BYTE and TEXT column. You can get at the contents with "string_var = ifx_get_blob($blob_id);" if you choose to get the BLOBs in memory (with : "ifx_blobinfile(0);"). If you prefer to receive the content of BLOB columns in a file, use "ifx_blobinfile(1);", and "ifx_get_blob($blob_id);" will get you the filename. Use normal file I/O to get at the blob contents.

For insert/update queries you must create these "blob id's" yourself with "**ifx_create_blob()**;". You then plug the blob id's into an array, and replace the blob columns with a question mark (?) in the query string. For updates/inserts, you are responsible for setting the blob contents with **ifx_update_blob()**.

The behaviour of BLOB columns can be altered by configuration variables that also can be set at runtime :

configuration variable : ifx.textasvarchar

configuration variable : ifx.byteasvarchar

runtime functions :

ifx_textasvarchar(0) : use blob id's for select queries with TEXT columns

ifx_byteasvarchar(0) : use blob id's for select queries with BYTE columns

ifx_textasvarchar(1) : return TEXT columns as if they were VARCHAR columns, so that you don't need to use blob id's for select queries.

ifx_byteasvarchar(1) : return BYTE columns as if they were VARCHAR columns, so that you don't need to use blob id's for select queries.

configuration variable : ifx.blobinfile

runtime function :

ifx_blobinfile_mode(0) : return BYTE columns in memory, the blob id lets you get at the contents.

ifx_blobinfile_mode(1) : return BYTE columns in a file, the blob id lets you get at the file name.

If you set ifx_text/byteasvarchar to 1, you can use TEXT and BYTE columns in select queries just like normal (but rather long) VARCHAR fields. Since all strings are "counted" in PHP, this remains "binary safe". It is up to you to handle this correctly. The returned data can contain anything, you are responsible for the contents.

If you set ifx_blobinfile to 1, use the file name returned by ifx_get_blob(..) to get at the blob contents. Note that in this case YOU ARE RESPONSIBLE FOR DELETING THE TEMPORARY FILES CREATED BY INFORMIX when fetching the row. Every new row fetched will create new temporary files for every BYTE column.

The location of the temporary files can be influenced by the environment variable "blobdir", default is "." (the current directory). Something like : putenv(blobdir=tmpblob"); will ease the cleaning up of temp files accidentally left behind (their names all start with "blb").

Automatically trimming "char" (SQLCHAR and SQLNCHAR) data: This can be set with the configuration variable

ifx.charasvarchar : if set to 1 trailing spaces will be automatically trimmed, to save you some "chopping".

NULL values: The configuration variable ifx.nullformat (and the runtime function **ifx_nullformat()**) when set to **TRUE** will return **NULL** columns as the string "**NULL**", when set to **FALSE** they return the empty string. This allows you to discriminate between **NULL** columns and empty columns.

Table 1. Informix configuration options

Name	Default	Changeable
ifx.allow_persistent	"1"	PHP_INI_SYSTEM
ifx.max_persistent	"-1"	PHP_INI_SYSTEM
ifx.max_links	"-1"	PHP_INI_SYSTEM
ifx.default_host	NULL	PHP_INI_SYSTEM
ifx.default_user	NULL	PHP_INI_SYSTEM
ifx.default_password	NULL	PHP_INI_SYSTEM
ifx.blobinfile	"1"	PHP_INI_ALL
ifx.textasvarchar	"0"	PHP_INI_ALL
ifx.byteasvarchar	"0"	PHP_INI_ALL
ifx.charasvarchar	"0"	PHP_INI_ALL
ifx.nullformat	"0"	PHP_INI_ALL

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Here is a short explanation of the configuration directives.

ifx.allow_persistent **boolean**

Whether to allow persistent Informix connections.

ifx.max_persistent **integer**

The maximum number of persistent Informix connections per process.

ifx.max_links **integer**

The maximum number of Informix connections per process, including persistent connections.

ifx.default_host **string**

The default host to connect to when no host is specified in **ifx_connect()** or **ifx_pconnect()**. Doesn't apply in safe mode.

ifx.default_user **string**

The default user id to use when none is specified in **ifx_connect()** or **ifx_pconnect()**. Doesn't apply in safe mode.

ifx.default_password **string**

The default password to use when none is specified in **ifx_connect()** or **ifx_pconnect()**. Doesn't apply in safe mode.

ifx.blobinfile **boolean**

Set to **TRUE** if you want to return blob columns in a file, **FALSE** if you want them in memory. You can override the setting at runtime with **ifx_blobinfile_mode()**.

ifx.textasvarchar **boolean**

Set to **TRUE** if you want to return TEXT columns as normal strings in select statements, **FALSE** if you want to use blob id parameters. You can override the setting at runtime with **ifx_textasvarchar()**.

ifx.byteasvarchar **boolean**

Set to **TRUE** if you want to return BYTE columns as normal strings in select queries, **FALSE** if you want to use blob id parameters. You can override the setting at runtime with **ifx_textasvarchar()**.

ifx.charasvarchar **boolean**

Set to **TRUE** if you want to trim trailing spaces from CHAR columns when fetching them.

ifx.nullformat **boolean**

Set to **TRUE** if you want to return **NULL** columns as the literal string "**NULL**", **FALSE** if you want them returned as the empty string "". You can override this setting at runtime with **ifx_nullformat()**.

Resource Types

Predefined Constants

This extension has no constants defined.

Table of Contents

ifx_affected_rows

(PHP 3>= 3.0.3, PHP 4)

ifx_affected_rows -- Get number of rows affected by a query

Description

int **ifx_affected_rows** (int result_id)

result_id is a valid result id returned by **ifx_query()** or **ifx_prepare()**.

Returns the number of rows affected by a query associated with *result_id*.

For inserts, updates and deletes the number is the real number (sqlerrd[2]) of affected rows. For selects it is an estimate (sqlerrd[0]). Don't rely on it. The database server can never return the actual number of rows that will be returned by a SELECT because it has not even begun fetching them at this stage (just after the "PREPARE" when the optimizer has determined the query plan).

Useful after **ifx_prepare()** to limit queries to reasonable result sets.

See also: **ifx_num_rows()**

Example 1. Informix affected rows

```
$rid = ifx_prepare ("select * from emp
                     where name like " . $name, $connid);
if (! $rid) {
    ... error ...
}
$rowcount = ifx_affected_rows ($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)\n<br>", $rowcount);
    die ("Please restrict your query<br>\n");
}
```

ifx_blobinfile_mode

(PHP 3>= 3.0.4, PHP 4)

ifx_blobinfile_mode -- Set the default blob mode for all select queries

Description

void **ifx_blobinfile_mode** (int mode)

Set the default blob mode for all select queries. Mode "0" means save Byte-Blobs in memory, and mode "1" means save Byte-Blobs in a file.

ifx_byteasvarchar

(PHP 3>= 3.0.4, PHP 4)

ifx_byteasvarchar -- Set the default byte mode

Description

void **ifx_byteasvarchar** (int mode)

Sets the default byte mode for all select-queries. Mode "0" will return a blob id, and mode "1" will return a varchar with text content.

ifx_close

(PHP 3>= 3.0.3, PHP 4)

ifx_close -- Close Informix connection

Description

int **ifx_close** ([int link_identifier])

Returns: always **TRUE**.

ifx_close() closes the link to an Informix database that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

Note that this isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

ifx_close() will not close persistent links generated by **ifx_pconnect()**.

See also: **ifx_connect()**, and **ifx_pconnect()**.

Example 1. Closing a Informix connection

```
$conn_id = ifx_connect ("mydb@ol_srv", "itsme", "mypassword");
... some queries and stuff ...
ifx_close($conn_id);
```

ifx_connect

(PHP 3>= 3.0.3, PHP 4)

ifx_connect -- Open Informix server connection

Description

int **ifx_connect** ([string database [, string userid [, string password]]])

Returns a connection identifier on success, or **FALSE** on error.

ifx_connect() establishes a connection to an Informix server. All of the arguments are optional, and if they're missing, defaults are taken from values supplied in configuration file (ifx.default_host for the host (Informix libraries will use INFORMIXSERVER environment value if not defined), ifx.default_user for user, ifx.default_password for the password (none if not defined).

In case a second call is made to **ifx_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling **ifx_close()**.

See also **ifx_pconnect()**, and **ifx_close()**.

Example 1. Connect to a Informix database

```
$conn_id = ifx_connect ("mydb@ol_srv1", "imyself", "mypassword");
```

ifx_copy_blob

(PHP 3>= 3.0.4, PHP 4)

ifx_copy_blob -- Duplicates the given blob object

Description

int **ifx_copy_blob** (int bid)

Duplicates the given blob object. *bid* is the ID of the blob object.

Returns **FALSE** on error otherwise the new blob object-id.

ifx_create_blob

(PHP 3>= 3.0.4, PHP 4)

ifx_create_blob -- Creates an blob object

Description

int **ifx_create_blob** (int type, int mode, string param)

Creates an blob object.

type: 1 = TEXT, 0 = BYTE

mode: 0 = blob-object holds the content in memory, 1 = blob-object holds the content in file.

param: if mode = 0: pointer to the content, if mode = 1: pointer to the filestring.

Return **FALSE** on error, otherwise the new blob object-id.

ifx_create_char

(PHP 3>= 3.0.6, PHP 4)

ifx_create_char -- Creates an char object

Description

int **ifx_create_char** (string param)

Creates an char object. *param* should be the char content.

ifx_do

(PHP 3>= 3.0.4, PHP 4)

ifx_do -- Execute a previously prepared SQL-statement

Description

int **ifx_do** (int result_id)

Returns **TRUE** on success or **FALSE** on failure.

Executes a previously prepared query or opens a cursor for it.

Does NOT free *result_id* on error.

Also sets the real number of **ifx_affected_rows()** for non-select statements for retrieval by **ifx_affected_rows()**

See also: **ifx_prepare()**.

ifx_error

(PHP 3>= 3.0.3, PHP 4)

ifx_error -- Returns error code of last Informix call

Description

string **ifx_error** (void)

The Informix error codes (SQLSTATE & SQLCODE) formatted as follows :

x [SQLSTATE = aa bbb SQLCODE=cccc]

where x = space : no error

E : error

N : no more data

W : warning

? : undefined

If the "x" character is anything other than space, SQLSTATE and SQLCODE describe the error in more detail.

See the Informix manual for the description of SQLSTATE and SQLCODE

Returns in a string one character describing the general results of a statement and both SQLSTATE and SQLCODE associated with the most recent SQL statement executed. The format of the string is "(char) [SQLSTATE=(two digits) (three digits) SQLCODE=(one digit)]". The first character can be ' ' (space) (success), 'W' (the statement caused some warning), 'E' (an error happened when executing the statement) or 'N' (the statement didn't return any data).

See also: **ifx_errormsg()**

ifx_errormsg

(PHP 3>= 3.0.4, PHP 4)

ifx_errormsg -- Returns error message of last Informix call

Description

string **ifx_errormsg** ([int errorcode])

Returns the Informix error message associated with the most recent Informix error, or, when the optional "*errorcode*" param is present, the error message corresponding to "*errorcode*".

See also: **ifx_error()**

```
printf("%s\n&lt;br>", ifx_errormsg(-201));
```

ifx_fetch_row

(PHP 3>= 3.0.3, PHP 4)

ifx_fetch_row -- Get row as enumerated array

Description

array **ifx_fetch_row** (int result_id [, mixed position])

Returns an associative array that corresponds to the fetched row, or **FALSE** if there are no more rows.

Blob columns are returned as integer blob id values for use in **ifx_get_blob()** unless you have used ifx_textasvarchar(1) or ifx_byteasvarchar(1), in which case blobs are returned as string values. Returns **FALSE** on error

result_id is a valid resultid returned by **ifx_query()** or **ifx_prepare()** (select type queries only!).

position is an optional parameter for a "fetch" operation on "scroll" cursors: "NEXT", "PREVIOUS", "CURRENT", "FIRST", "LAST" or a number. If you specify a number, an "absolute" row fetch is executed. This parameter is optional, and only valid for SCROLL cursors.

ifx_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0, with the column name as key.

Subsequent calls to **ifx_fetch_row()** would return the next row in the result set, or **FALSE** if there are no more rows.

Example 1. Informix fetch rows

```
$rid = ifx_prepare ("select * from emp where name like " . $name,
                    $connid, IFX_SCROLL);
if (! $rid) {
    ... error ...
}
$rowcount = ifx_affected_rows($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)\n<br>", $rowcount);
    die ("Please restrict your query<br>\n");
}
if (! ifx_do ($rid)) {
    ... error ...
}
$row = ifx_fetch_row ($rid, "NEXT");
while (is_array($row)) {
    for(reset($row); $fieldname=key($row); next($row)) {
        $fieldvalue = $row[$fieldname];
        printf ("%s = %s,", $fieldname, $fieldvalue);
    }
    printf("\n<br>");
    $row = ifx_fetch_row ($rid, "NEXT");
}
ifx_free_result ($rid);
```

ifx_fieldproperties

(PHP 3>= 3.0.3, PHP 4)

ifx_fieldproperties -- List of SQL fieldproperties

Description

array **ifx_fieldproperties** (int result_id)

Returns an associative array with fieldnames as key and the SQL fieldproperties as data for a query with *result_id*. Returns **FALSE** on error.

Returns the Informix SQL fieldproperties of every field in the query as an associative array. Properties are encoded as: "SQLTYPE;length;precision;scale;ISNULLABLE" where SQLTYPE = the Informix type like "SQLVCHAR" etc. and ISNULLABLE = "Y" or "N".

Example 1. Informix SQL fieldproperties

```
$properties = ifx_fieldproperties ($resultid);
if (! isset($properties)) {
   ... error ...
}
for ($i = 0; $i < count($properties); $i++) {
    $fname = key ($properties);
    printf ("%s:\t type =  %s\n", $fname, $properties[$fname]);
    next ($properties);
}
```

ifx_fieldtypes

(PHP 3>= 3.0.3, PHP 4)

ifx_fieldtypes -- List of Informix SQL fields

Description

array **ifx_fieldtypes** (int result_id)

Returns an associative array with fieldnames as key and the SQL fieldtypes as data for query with *result_id*. Returns **FALSE** on error.

Example 1. Fieldnames and SQL fieldtypes

```
$types = ifx_fieldtypes ($resultid);
if (! isset ($types)) {
  ... error ...
}
for ($i = 0; $i < count($types); $i++) {
    $fname = key($types);
    printf("%s :\t type =  %s\n", $fname, $types[$fname]);
    next($types);
}
```

ifx_free_blob

(PHP 3>= 3.0.4, PHP 4)

ifx_free_blob -- Deletes the blob object

Description

int **ifx_free_blob** (int bid)

Deletes the blobobject for the given blob object-id *bid*. Returns **TRUE** on success or **FALSE** on failure.

ifx_free_char

(PHP 3>= 3.0.6, PHP 4)

ifx_free_char -- Deletes the char object

Description

int **ifx_free_char** (int bid)

Deletes the charobject for the given char object-id *bid*. Returns **TRUE** on success or **FALSE** on failure.

ifx_free_result

(PHP 3>= 3.0.3, PHP 4)

ifx_free_result -- Releases resources for the query

Description

int **ifx_free_result** (int result_id)

Releases resources for the query associated with *result_id*. Returns **FALSE** on error.

ifx_get_blob

(PHP 3>= 3.0.4, PHP 4)

ifx_get_blob -- Return the content of a blob object

Description

int **ifx_get_blob** (int bid)

Returns the content of the blob object for the given blob object-id *bid*.

ifx_get_char

(PHP 3>= 3.0.6, PHP 4)

ifx_get_char -- Return the content of the char object

Description

int **ifx_get_char** (int bid)

Returns the content of the char object for the given char object-id *bid*.

ifx_getsqlca

(PHP 3>= 3.0.8, PHP 4)

ifx_getsqlca -- Get the contents of sqlca.sqlerrd[0..5] after a query

Description

array **ifx_getsqlca** (int result_id)

result_id is a valid result id returned by **ifx_query()** or **ifx_prepare()**.

Returns a pseudo-row (associative array) with sqlca.sqlerrd[0] ... sqlca.sqlerrd[5] after the query associated with *result_id*.

For inserts, updates and deletes the values returned are those as set by the server after executing the query. This gives access to the number of affected rows and the serial insert value. For SELECTs the values are those saved after the PREPARE statement. This gives access to the *estimated* number of affected rows. The use of this function saves the overhead of executing a "select dbinfo('sqlca.sqlerrdx')" query, as it retrieves the values that were saved by the ifx driver at the appropriate moment.

Example 1. Retrieve Informix sqlca.sqlerrd[x] values

```
/* assume the first column of 'sometable' is a serial */
$qid = ifx_query("insert into sometable
                  values (0, '2nd column', 'another column') ", $connid);
if (! $qid) {
    ... error ...
}
$sqlca = ifx_getsqlca ($qid);
$serial_value = $sqlca["sqlerrd1"];
echo "The serial value of the inserted row is : " . $serial_value<br>\n";
```

ifx_htmltbl_result

(PHP 3>= 3.0.3, PHP 4)

ifx_htmltbl_result -- Formats all rows of a query into a HTML table

Description

int **ifx_htmltbl_result** (int result_id [, string html_table_options])

Returns the number of rows fetched or **FALSE** on error.

Formats all rows of the *result_id* query into a html table. The optional second argument is a string of <table> tag options

Example 1. Informix results as HTML table

```
$rid = ifx_prepare ("select * from emp where name like " . $name,
                    $connid, IFX_SCROLL);
if (! $rid) {
   ... error ...
}
$rowcount = ifx_affected_rows ($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)\n<br>", $rowcount);
    die ("Please restrict your query<br>\n");
}
if (! ifx_do($rid) {
  ... error ...
}

ifx_htmltbl_result ($rid, "border=\"2\"");

ifx_free_result($rid);
```

ifx_nullformat

(PHP 3>= 3.0.4, PHP 4)

ifx_nullformat -- Sets the default return value on a fetch row

Description

void **ifx_nullformat** (int mode)

Sets the default return value of a NULL-value on a fetch row. Mode "0" returns "", and mode "1" returns "**NULL**".

ifx_num_fields

(PHP 3>= 3.0.3, PHP 4)

ifx_num_fields -- Returns the number of columns in the query

Description

int **ifx_num_fields** (int result_id)

Returns the number of columns in query for *result_id* or **FALSE** on error

After preparing or executing a query, this call gives you the number of columns in the query.

ifx_num_rows

(PHP 3>= 3.0.3, PHP 4)

ifx_num_rows -- Count the rows already fetched from a query

Description

int **ifx_num_rows** (int result_id)

Gives the number of rows fetched so far for a query with *result_id* after a **ifx_query()** or **ifx_do()** query.

ifx_pconnect

(PHP 3>= 3.0.3, PHP 4)

ifx_pconnect -- Open persistent Informix connection

Description

int **ifx_pconnect** ([string database [, string userid [, string password]]])

Returns: A positive Informix persistent link identifier on success, or **FALSE** on error

ifx_pconnect() acts very much like **ifx_connect()** with two major differences.

This function behaves exactly like **ifx_connect()** when PHP is not running as an Apache module. First, when connecting, the function would first try to find a (persistent) link that's already open with the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (**ifx_close()** will not close links established by **ifx_pconnect()**).

This type of links is therefore called 'persistent'.

See also: **ifx_connect()**.

ifx_prepare

(PHP 3>= 3.0.4, PHP 4)

ifx_prepare -- Prepare an SQL-statement for execution

Description

int **ifx_prepare** (string query, int conn_id [, int cursor_def, mixed blobidarray])

Returns a integer *result_id* for use by **ifx_do()**. Sets *affected_rows* for retrieval by the **ifx_affected_rows()** function.

Prepares *query* on connection *conn_id*. For "select-type" queries a cursor is declared and opened. The optional *cursor_type* parameter allows you to make this a "scroll" and/or "hold" cursor. It's a bitmask and can be either IFX_SCROLL, IFX_HOLD, or both or'ed together.

For either query type the estimated number of affected rows is saved for retrieval by **ifx_affected_rows()**.

If you have BLOB (BYTE or TEXT) columns in the query, you can add a *blobidarray* parameter containing the corresponding "blob ids", and you should replace those columns with a "?" in the query text.

If the contents of the TEXT (or BYTE) column allow it, you can also use "ifx_textasvarchar(1)" and "ifx_byteasvarchar(1)". This allows you to treat TEXT (or BYTE) columns just as if they were ordinary (but long) VARCHAR columns for select queries, and you don't need to bother with blob id's.

With ifx_textasvarchar(0) or ifx_byteasvarchar(0) (the default situation), select queries will return BLOB columns as blob id's (integer value). You can get the value of the blob as a string or file with the blob functions (see below).

See also: **ifx_do()**.

ifx_query

(PHP 3>= 3.0.3, PHP 4)

ifx_query -- Send Informix query

Description

int **ifx_query** (string query, int link_identifier [, int cursor_type [, mixed blobidarray]])

Returns: A positive Informix result identifier on success, or **FALSE** on error.

A "result_id" resource used by other functions to retrieve the query results. Sets "affected_rows" for retrieval by the

ifx_affected_rows() function.

ifx_query() sends a query to the currently active database on the server that's associated with the specified link identifier.

Executes *query* on connection *conn_id*. For "select-type" queries a cursor is declared and opened. The optional *cursor_type* parameter allows you to make this a "scroll" and/or "hold" cursor. It's a bitmask and can be either IFX_SCROLL, IFX_HOLD, or both or'ed together. Non-select queries are "execute immediate". IFX_SCROLL and IFX_HOLD are symbolic constants and as such shouldn't be between quotes. I you omit this parameter the cursor is a normal sequential cursor.

For either query type the number of (estimated or real) affected rows is saved for retrieval by ifx_affected_rows().

If you have BLOB (BYTE or TEXT) columns in an update query, you can add a *blobidarray* parameter containing the corresponding "blob ids", and you should replace those columns with a "?" in the query text.

If the contents of the TEXT (or BYTE) column allow it, you can also use "ifx_textasvarchar(1)" and "ifx_byteasvarchar(1)". This allows you to treat TEXT (or BYTE) columns just as if they were ordinary (but long) VARCHAR columns for select queries, and you don't need to bother with blob id's.

With ifx_textasvarchar(0) or ifx_byteasvarchar(0) (the default situation), select queries will return BLOB columns as blob id's (integer value). You can get the value of the blob as a string or file with the blob functions (see below).

See also: ifx_connect().

Example 1. Show all rows of the "orders" table as a html table

```
ifx_textasvarchar(1);      // use "text mode" for blobs
$res_id = ifx_query("select * from orders", $conn_id);
if (! $res_id) {
    printf("Can't select orders : %s\n<br>%s<br>\n", ifx_error());
    ifx_errormsg();
    die;
}
ifx_htmltbl_result($res_id, "border=\"1\"");
ifx_free_result($res_id);
```

Example 2. Insert some values into the "catalog" table

```
// create blob id's for a byte and text column
$textid = ifx_create_blob(0, 0, "Text column in memory");
$byteid = ifx_create_blob(1, 0, "Byte column in memory");
                    // store blob id's in a blobid array
$blobidarray[] = $textid;
$blobidarray[] = $byteid;
                    // launch query
$query = "insert into catalog (stock_num, manu_code, " .
         "cat_descr,cat_picture) values(1,'HRO',?,?)";
$res_id = ifx_query($query, $conn_id, $blobidarray);
if (! $res_id) {
  ... error ...
}
                    // free result id
ifx_free_result($res_id);
```

ifx_textasvarchar

(PHP 3>= 3.0.4, PHP 4)

ifx_textasvarchar -- Set the default text mode

Description

void **ifx_textasvarchar** (int mode)

Sets the default text mode for all select-queries. Mode "0" will return a blob id, and mode "1" will return a varchar with text content.

ifx_update_blob

(PHP 3>= 3.0.4, PHP 4)

ifx_update_blob -- Updates the content of the blob object

Description

ifx_update_blob (int bid, string content)

Updates the content of the blob object for the given blob object *bid*. *content* is a string with new data. Returns **TRUE** on success or **FALSE** on failure.

ifx_update_char

(PHP 3>= 3.0.6, PHP 4)

ifx_update_char -- Updates the content of the char object

Description

int **ifx_update_char** (int bid, string content)

Updates the content of the char object for the given char object *bid*. *content* is a string with new data. Returns **TRUE** on success or **FALSE** on failure.

ifxus_close_slob

(PHP 3>= 3.0.4, PHP 4)

ifxus_close_slob -- Deletes the slob object

Description

int **ifxus_close_slob** (int bid)

Deletes the slob object on the given slob object-id *bid*. Returns **TRUE** on success or **FALSE** on failure.

ifxus_create_slob

(PHP 3>= 3.0.4, PHP 4)

ifxus_create_slob -- Creates an slob object and opens it

Description

int **ifxus_create_slob** (int mode)

Creates an slob object and opens it. Modes: 1 = LO_RDONLY, 2 = LO_WRONLY, 4 = LO_APPEND, 8 = LO_RDWR, 16 = LO_BUFFER, 32 = LO_NOBUFFER -> or-mask. You can also use constants named IFX_LO_RDONLY, IFX_LO_WRONLY etc. Return **FALSE** on error otherwise the new slob object-id.

ifxus_free_slob

(PHP 3>= 3.0.4, PHP 4)

ifxus_free_slob -- Deletes the slob object

Description

int **ifxus_free_slob** (int bid)

Deletes the slob object. *bid* is the Id of the slob object. Returns **TRUE** on success or **FALSE** on failure.

ifxus_open_slob

(PHP 3>= 3.0.4, PHP 4)

ifxus_open_slob -- Opens an slob object

Description

int **ifxus_open_slob** (long bid, int mode)

Opens an slob object. *bid* should be an existing slob id. Modes: 1 = LO_RDONLY, 2 = LO_WRONLY, 4 = LO_APPEND, 8 = LO_RDWR, 16 = LO_BUFFER, 32 = LO_NOBUFFER -> or-mask. Returns **FALSE** on error otherwise the new slob object-id.

ifxus_read_slob

(PHP 3>= 3.0.4, PHP 4)

ifxus_read_slob -- Reads nbytes of the slob object

Description

int **ifxus_read_slob** (long bid, long nbytes)

Reads nbytes of the slob object. *bid* is a existing slob id and *nbytes* is the number of bytes read. Return **FALSE** on error otherwise the string.

ifxus_seek_slob

(PHP 3>= 3.0.4, PHP 4)

ifxus_seek_slob -- Sets the current file or seek position

Description

int **ifxus_seek_slob** (long bid, int mode, long offset)

Sets the current file or seek position of an open slob object. *bid* should be an existing slob id. Modes: 0 = LO_SEEK_SET, 1 = LO_SEEK_CUR, 2 = LO_SEEK_END and *offset* is an byte offset. Return **FALSE** on error otherwise the seek position.

ifxus_tell_slob

(PHP 3>= 3.0.4, PHP 4)

ifxus_tell_slob -- Returns the current file or seek position

Description

int **ifxus_tell_slob** (long bid)

Returns the current file or seek position of an open slob object *bid* should be an existing slob id. Return **FALSE** on error otherwise the seek position.

ifxus_write_slob

(PHP 3>= 3.0.4, PHP 4)

ifxus_write_slob -- Writes a string into the slob object

Description

int **ifxus_write_slob** (long bid, string content)

Writes a string into the slob object. *bid* is a existing slob id and *content* the content to write. Return **FALSE** on error otherwise bytes written.

XLIV. InterBase functions

Introduction

InterBase is a popular database put out by Borland/Inprise. More information about InterBase is available at http://www.interbase.com/. Oh, by the way, InterBase just joined the open source movement!

> **Note:** Full support for InterBase 6 was added in PHP 4.0.
>
> This database uses a single quote (') character for escaping, a behavior similar to the Sybase database, add to your `php.ini` the following directive:
>
> ```
> magic_quotes_sybase = On
> ```

Requirements

Installation

To enable InterBase support configure PHP `--with-interbase[=DIR]`, where DIR is the InterBase base install directory, which defaults to `/usr/interbase`.

> **Note to Win32 Users:** In order to enable this module on a Windows environment, you must copy *gds32.dll* from the DLL folder of the PHP/Win32 binary package to the SYSTEM32 folder of your windows machine. (Ex: C:\WINNT\SYSTEM32 or C:\WINDOWS\SYSTEM32). In case you installed the InterBase database server on the same machine PHP is running on, you will have this DLL already. Therfore you don't need to copy *gds32.dll* from the DLL folder.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. InterBase configuration options

Name	Default	Changeable
ibase.allow_persistent	"1"	PHP_INI_SYSTEM
ibase.max_persistent	"-1"	PHP_INI_SYSTEM
ibase.max_links	"-1"	PHP_INI_SYSTEM
ibase.default_user	NULL	PHP_INI_ALL
ibase.default_password	NULL	PHP_INI_ALL
ibase.timestampformat	"%m/%d/%Y%H:%M:%S"	PHP_INI_ALL
ibase.dateformat	"%m/%d/%Y"	PHP_INI_ALL
ibase.timeformat	"%H:%M:%S"	PHP_INI_ALL

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Resource Types

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

IBASE_DEFAULT (integer)

IBASE_TEXT (integer)

IBASE_UNIXTIME (integer)

IBASE_READ (integer)

IBASE_COMMITTED (integer)

IBASE_CONSISTENCY (integer)

IBASE_NOWAIT (integer)

IBASE_TIMESTAMP (integer)

IBASE_DATE (integer)

IBASE_TIME (integer)

Table of Contents

ibase_blob_add

(PHP 3>= 3.0.7, PHP 4)

ibase_blob_add -- Add data into created blob

Description

int **ibase_blob_add** (int blob_id, string data)

Warning
This function is currently not documented; only the argument list is available.

ibase_blob_cancel

(PHP 3>= 3.0.7, PHP 4)

ibase_blob_cancel -- Cancel creating blob

Description

int **ibase_blob_cancel** (int blob_id)

Warning
This function is currently not documented; only the argument list is available.

ibase_blob_close

(PHP 3>= 3.0.7, PHP 4)

ibase_blob_close -- Close blob

Description

int **ibase_blob_close** (int blob_id)

Warning
This function is currently not documented; only the argument list is available.

ibase_blob_create

(PHP 3>= 3.0.7, PHP 4)

ibase_blob_create -- Create blob for adding data

Description

int **ibase_blob_create** ([int link_identifier])

Warning
This function is currently not documented; only the argument list is available.

ibase_blob_echo

(PHP 3>= 3.0.7, PHP 4)

ibase_blob_echo -- Output blob contents to browser

Description

int **ibase_blob_echo** (string blob_id_str)

Warning
This function is currently not documented; only the argument list is available.

ibase_blob_get

(PHP 3>= 3.0.7, PHP 4)

ibase_blob_get -- Get len bytes data from open blob

Description

string **ibase_blob_get** (int blob_id, int len)

Warning
This function is currently not documented; only the argument list is available.

ibase_blob_import

(PHP 3>= 3.0.7, PHP 4)

ibase_blob_import -- Create blob, copy file in it, and close it

Description

string **ibase_blob_import** ([int link_identifier, int file_id])

Warning
This function is currently not documented; only the argument list is available.

ibase_blob_info

(PHP 3>= 3.0.7, PHP 4)

ibase_blob_info -- Return blob length and other useful info

Description

object **ibase_blob_info** (string blob_id_str)

Warning
This function is currently not documented; only the argument list is available.

ibase_blob_open

(PHP 3>= 3.0.7, PHP 4)

ibase_blob_open -- Open blob for retrieving data parts

Description

int **ibase_blob_open** (string blob_id)

Warning
This function is currently not documented; only the argument list is available.

ibase_close

(PHP 3>= 3.0.6, PHP 4)

ibase_close -- Close a connection to an InterBase database

Description

int **ibase_close** ([int connection_id])

Closes the link to an InterBase database that's associated with a connection id returned from **ibase_connect()**. If the connection id is omitted, the last opened link is assumed. Default transaction on link is committed, other transactions are rolled back.

ibase_commit

(PHP 3>= 3.0.7, PHP 4)

ibase_commit -- Commit a transaction

Description

int **ibase_commit** ([int link_identifier, int trans_number])

Commits transaction *trans_number* which was created with **ibase_trans()**.

ibase_connect

(PHP 3>= 3.0.6, PHP 4)

ibase_connect -- Open a connection to an InterBase database

Description

int **ibase_connect** (string database [, string username [, string password [, string charset [, int buffers [, int dialect [, string role]]]]]])

Establishes a connection to an InterBase server. The *database* argument has to be a valid path to database file on the server it resides on. If the server is not local, it must be prefixed with either 'hostname:' (TCP/IP), '//hostname/' (NetBEUI) or 'hostname@' (IPX/SPX), depending on the connection protocol used. *username* and *password* can also be specified with PHP configuration directives ibase.default_user and ibase.default_password. *charset* is the default character set for a database. *buffers* is the number of database buffers to allocate for the server-side cache. If 0 or omitted, server chooses its own default. *dialect* selects the default SQL dialect for any statement executed within a connection, and it defaults to the highest one supported by client libraries.

In case a second call is made to **ibase_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned. The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling **ibase_close()**.

Example 1. ibase_connect() example

```
<?php
    $host = 'localhost:/path/to/your.gdb';

    $dbh = ibase_connect($host, $username, $password);
    $stmt = 'SELECT * FROM tblname';
    $sth = ibase_query($dbh, $stmt);
    while ($row = ibase_fetch_object($sth)) {
        echo $row->email, "\n";
    }
    ibase_free_result($sth);
    ibase_close($dbh);
?>
```

Note: The optional *buffers* parameter was added in PHP 4.0.0.

Note: The optional *dialect* parameter was added in PHP 4.0.0 and is functional only with InterBase 6 and up.

Note: The optional *role* parameter was added in PHP 4.0.0 and is functional only with InterBase 5 and up.

See also **ibase_pconnect()**.

ibase_errmsg

(PHP 3>= 3.0.7, PHP 4)

ibase_errmsg -- Returns error messages

Description

string **ibase_errmsg** (void)

Returns a string containing an error message.

ibase_execute

(PHP 3>= 3.0.6, PHP 4)

ibase_execute -- Execute a previously prepared query

Description

int **ibase_execute** (int query [, int bind_args])

Execute a query prepared by **ibase_prepare()**. This is a lot more effective than using **ibase_query()** if you are repeating a same kind of query several times with only some parameters changing.

```
<?php
    $updates = array(
        1 => 'Eric',
        5 => 'Filip',
        7 => 'Larry'
    );

    $query = ibase_prepare("UPDATE FOO SET BAR = ? WHERE BAZ = ?");

    while (list($baz, $bar) = each($updates)) {
        ibase_execute($query, $bar, $baz);
    }
?>
```

ibase_fetch_object

(PHP 3>= 3.0.7, PHP 4)

ibase_fetch_object -- Get an object from a InterBase database

Description

object **ibase_fetch_object** (int result_id)

Fetches a row as a pseudo-object from a *result_id* obtained either by **ibase_query()** or **ibase_execute()**.

```
<php
    $dbh = ibase_connect ($host, $username, $password);
    $stmt = 'SELECT * FROM tblname';
    $sth = ibase_query ($dbh, $stmt);
    while ($row = ibase_fetch_object ($sth)) {
        print $row->email . "\n";
    }
    ibase_close ($dbh);
?>
```

Subsequent call to **ibase_fetch_object()** would return the next row in the result set, or **FALSE** if there are no more rows.

See also **ibase_fetch_row()**.

ibase_fetch_row

(PHP 3>= 3.0.6, PHP 4)

ibase_fetch_row -- Fetch a row from an InterBase database

Description

array **ibase_fetch_row** (int result_identifier)

Returns an array that corresponds to the fetched row, or **FALSE** if there are no more rows.

ibase_fetch_row() fetches one row of data from the result associated with the specified *result_identifier*. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to **ibase_fetch_row()** would return the next row in the result set, or **FALSE** if there are no more rows.

ibase_field_info

(PHP 3>= 3.0.7, PHP 4)

ibase_field_info -- Get information about a field

Description

array **ibase_field_info** (int result, int field number)

Returns an array with information about a field after a select query has been run. The array is in the form of name, alias, relation, length, type.

```
$rs=ibase_query("SELECT * FROM tablename");
$coln = ibase_num_fields($rs);
for ($i=0; $i < $coln; $i++) {
    $col_info = ibase_field_info($rs, $i);
    echo "name: ".$col_info['name']."\n";
    echo "alias: ".$col_info['alias']."\n";
    echo "relation: ".$col_info['relation']."\n";
    echo "length: ".$col_info['length']."\n";
    echo "type: ".$col_info['type']."\n";
    }
```

ibase_free_query

(PHP 3>= 3.0.6, PHP 4)

ibase_free_query -- Free memory allocated by a prepared query

Description

int **ibase_free_query** (int query)

Free a query prepared by **ibase_prepare()**.

ibase_free_result

(PHP 3>= 3.0.6, PHP 4)

ibase_free_result -- Free a result set

Description

int **ibase_free_result** (int result_identifier)

Free's a result set the has been created by **ibase_query()**.

ibase_num_fields

(PHP 3>= 3.0.7, PHP 4)

ibase_num_fields -- Get the number of fields in a result set

Description

int **ibase_num_fields** (int result_id)

Returns an integer containing the number of fields in a result set.

```
<?php
        $dbh = ibase_connect ($host, $username, $password);
        $stmt = 'SELECT * FROM tblname';
        $sth = ibase_query ($dbh, $stmt);

        if (ibase_num_fields($sth) > 0) {
        while ($row = ibase_fetch_object ($sth)) {
        print $row->email . "\n";
        }
        } else {
        die ("No Results were found for your query");
        }

        ibase_close ($dbh);
        ?>
```

See also: **ibase_field_info()**.

ibase_pconnect

(PHP 3>= 3.0.6, PHP 4)

ibase_pconnect -- Creates an persistent connection to an InterBase database

Description

int **ibase_pconnect** (string database [, string username [, string password [, string charset [, int buffers [, int dialect [, string role]]]]]])

ibase_pconnect() acts very much like **ibase_connect()** with two major differences. First, when connecting, the function will first try to find a (persistent) link that's already opened with the same parameters. If one is found, an identifier for it will be returned instead of opening a new connection. Second, the connection to the InterBase server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (**ibase_close()** will not close links established by **ibase_pconnect()**). This type of link is therefore called 'persistent'.

> **Note:** *buffers* was added in PHP4-RC2.
>
> **Note:** *dialect* was added in PHP4-RC2. It is functional only with InterBase 6 and versions higher than that.
>
> **Note:** *role* was added in PHP4-RC2. It is functional only with InterBase 5 and versions higher than that.

See also **ibase_connect()** for the meaning of parameters passed to this function. They are exactly the same.

ibase_prepare

(PHP 3>= 3.0.6, PHP 4)

ibase_prepare -- Prepare a query for later binding of parameter placeholders and execution

Description

int **ibase_prepare** ([int link_identifier, string query])

Prepare a query for later binding of parameter placeholders and execution (via **ibase_execute()**).

ibase_query

(PHP 3>= 3.0.6, PHP 4)

ibase_query -- Execute a query on an InterBase database

Description

int **ibase_query** ([int link_identifier, string query [, int bind_args]])

Performs a query on an InterBase database. If the query is not successful, returns **FALSE**. If it is successful and there are resulting rows (such as with a SELECT query), returns a result identifier. If the query was successful and there were no results, returns **TRUE**. Returns **FALSE** if the query fails.

See also **ibase_errmsg()**, **ibase_fetch_row()**, **ibase_fetch_object()**, and **ibase_free_result()**.

ibase_rollback

(PHP 3>= 3.0.7, PHP 4)

ibase_rollback -- Rolls back a transaction

Description

int **ibase_rollback** ([int link_identifier, int trans_number])

Rolls back transaction *trans_number* which was created with **ibase_trans()**.

ibase_timefmt

(PHP 3>= 3.0.6, PHP 4)

ibase_timefmt -- Sets the format of timestamp, date and time type columns returned from queries

Description

int **ibase_timefmt** (string format [, int columntype])

Sets the format of timestamp, date or time type columns returned from queries. Internally, the columns are formatted by c-function strftime(), so refer to it's documentation regarding to the format of the string. *columntype* is one of the constants IBASE_TIMESTAMP, IBASE_DATE and IBASE_TIME. If omitted, defaults to IBASE_TIMESTAMP for backwards compatibility.

```
<?php
       // InterBase 6 TIME-type columns will be returned in
       // the form '05 hours 37 minutes'.
       ibase_timefmt("%H hours %M minutes", IBASE_TIME);
       ?>
```

You can also set defaults for these formats with PHP configuration directives ibase.timestampformat, ibase.dateformat and ibase.timeformat.

> **Note:** *columntype* was added in PHP 4.0. It has any meaning only with InterBase version 6 and higher.
>
> **Note:** A backwards incompatible change happened in PHP 4.0 when PHP configuration directive ibase.timeformat was renamed to ibase.timestampformat and directives ibase.dateformat and ibase.timeformat were added, so that the names would match better their functionality.

ibase_trans

(PHP 3>= 3.0.7, PHP 4)

ibase_trans -- Begin a transaction

Description

int **ibase_trans** ([int trans_args [, int link_identifier]])

Begins a transaction.

XLV. Ingres II functions

Introduction

These functions allow you to access Ingres II database servers.

> **Note:** If you already used PHP extensions to access other database servers, note that Ingres doesn't allow concurrent queries and/or transaction over one connection, thus you won't find any result or transaction handle in this extension. The result of a query must be treated before sending another query, and a transaction must be commited or rolled back before opening another transaction (which is automaticaly done when sending the first query).

Warning
This extension is *EXPERIMENTAL*. The behaviour of this extension -- including the names of its functions and anything else documented about this extension -- may change without notice in a future release of PHP. Use this extension at your own risk.

Requirements

To compile PHP with Ingres support, you need the Open API library and header files included with Ingres II.

Installation

In order to have these functions available, you must compile PHP with Ingres support by using the `--with-ingres[=DIR]` option, where DIR is the Ingres base directory, which defaults to `/II/ingres`. If the II_SYSTEM environment variable isn't correctly set you may have to use `--with-ingres=DIR` to specify your Ingres installation directory.

When using this extension with Apache, if Apache does not start and complains with "PHP Fatal error: Unable to start ingres_ii module in Unknown on line o" then make sure the environement variable II_SYSTEM is correctly set. Adding "export II_SYSTEM="/home/ingres/II" in the script that starts Apache, just before launching httpd, should be fine.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. Ingres II configuration options

Name	Default	Changeable
ingres.allow_persistent	"1"	PHP_INI_SYSTEM
ingres.max_persistent	"-1"	PHP_INI_SYSTEM
ingres.max_links	"-1"	PHP_INI_SYSTEM
ingres.default_database	NULL	PHP_INI_ALL
ingres.default_user	NULL	PHP_INI_ALL
ingres.default_password	NULL	PHP_INI_ALL

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

INGRES_ASSOC (integer)

INGRES_NUM (integer)

INGRES_BOTH (integer)

Table of Contents

ingres_autocommit

(PHP 4 >= 4.0.2)

ingres_autocommit -- Switch autocommit on or off

Description

bool **ingres_autocommit** ([resource link])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ingres_autocommit() is called before opening a transaction (before the first call to **ingres_query()** or just after a call to **ingres_rollback()** or **ingres_commit()**) to switch the "autocommit" mode of the server on or off (when the script begins the autocommit mode is off).

When the autocommit mode is on, every query is automaticaly commited by the server, as if **ingres_commit()** was called after every call to **ingres_query()**.

See also **ingres_query()**, **ingres_rollback()**, and **ingres_commit()**.

ingres_close

(PHP 4 >= 4.0.2)

ingres_close -- Close an Ingres II database connection

Description

bool **ingres_close** ([resource link])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Returns **TRUE** on success, or **FALSE** on failure.

ingres_close() closes the connection to the Ingres server that's associated with the specified link. If the *link* parameter isn't specified, the last opened link is used.

ingres_close() isn't usually necessary, as it won't close persistent connections and all non-persistent connections are automatically closed at the end of the script.

See also **ingres_connect()** and **ingres_pconnect()**.

ingres_commit

(PHP 4 >= 4.0.2)

ingres_commit -- Commit a transaction

Description

bool **ingres_commit** ([resource link])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ingres_commit() commits the currently open transaction, making all changes made to the database permanent.

This closes the transaction. A new one can be open by sending a query with **ingres_query()**.

You can also have the server commit automaticaly after every query by calling **ingres_autocommit()** before opening the transaction.

See also **ingres_query()**, **ingres_rollback()**, and **ingres_autocommit()**.

ingres_connect

(PHP 4 >= 4.0.2)

ingres_connect -- Open a connection to an Ingres II database

Description

resource **ingres_connect** ([string database [, string username [, string password]]])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Returns a Ingres II link resource on success, or **FALSE** on failure.

ingres_connect() opens a connection with the Ingres database designated by *database*, which follows the syntax *[node_id::]dbname[/svr_class]*.

If some parameters are missing, **ingres_connect()** uses the values in `php.ini` for *ingres.default_database*, *ingres.default_user*,

and *ingres.default_password*.

The connection is closed when the script ends or when **ingres_close()** is called on this link.

All the other ingres functions use the last opened link as a default, so you need to store the returned value only if you use more than one link at a time.

Example 1. ingres_connect() example

```
<?php
    $link = ingres_connect ("mydb", "user", "pass")
        or die ("Could not connect");
    print ("Connected successfully");
    ingres_close ($link);
?>
```

Example 2. ingres_connect() example using default link

```
<?php
    ingres_connect ("mydb", "user", "pass")
        or die ("Could not connect");
    print ("Connected successfully");
    ingres_close ();
?>
```

See also **ingres_pconnect()** and **ingres_close()**.

ingres_fetch_array

(PHP 4 >= 4.0.2)

ingres_fetch_array -- Fetch a row of result into an array

Description

array **ingres_fetch_array** ([int result_type [, resource link]])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ingres_fetch_array() Returns an array that corresponds to the fetched row, or **FALSE** if there are no more rows.

This function is an extended version of **ingres_fetch_row()**. In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you must use the numeric index of the column or make an alias for the column.

```
ingres_query(select t1.f1 as foo t2.f1 as bar from t1, t2);
$result = ingres_fetch_array();
$foo = $result["foo"];
$bar = $result["bar"];
```

result_type can be INGRES_NUM for enumerated array, INGRES_ASSOC for associative array, or INGRES_BOTH (default).

Speed-wise, the function is identical to **ingres_fetch_object()**, and almost as quick as **ingres_fetch_row()** (the difference is insignificant).

Example 1. ingres_fetch_array() example

```
<?php
ingres_connect ($database, $user, $password);

ingres_query ("select * from table");
while ($row = ingres_fetch_array()) {
    echo $row["user_id"];  # using associative array
    echo $row["fullname"];
    echo $row[1];          # using enumerated array
    echo $row[2];
}
?>
```

See also ingres_query(), ingres_num_fields(), ingres_field_name(), ingres_fetch_object(), and ingres_fetch_row().

ingres_fetch_object

(PHP 4 >= 4.0.2)

ingres_fetch_object -- Fetch a row of result into an object.

Description

object **ingres_fetch_object** ([int result_type [, resource link]])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ingres_fetch_object() Returns an object that corresponds to the fetched row, or FALSE if there are no more rows.

This function is similar to ingres_fetch_array(), with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

The optional argument *result_type* is a constant and can take the following values: INGRES_ASSOC, INGRES_NUM, and INGRES_BOTH.

Speed-wise, the function is identical to ingres_fetch_array(), and almost as quick as ingres_fetch_row() (the difference is insignificant).

Example 1. ingres_fetch_object() example

```
<?php
ingres_connect ($database, $user, $password);
ingres_query ("select * from table");
while ($row = ingres_fetch_object()) {
    echo $row->user_id;
    echo $row->fullname;
}
?>
```

See also ingres_query(), ingres_num_fields(), ingres_field_name(), ingres_fetch_array(), and ingres_fetch_row().

ingres_fetch_row

(PHP 4 >= 4.0.2)

ingres_fetch_row -- Fetch a row of result into an enumerated array

Description

array **ingres_fetch_row** ([resource link])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ingres_fetch_row() returns an array that corresponds to the fetched row, or FALSE if there are no more rows. Each result column is stored in an array offset, starting at offset 1.

Subsequent call to **ingres_fetch_row()** would return the next row in the result set, or FALSE if there are no more rows.

Example 1. ingres_fetch_row() example

```
<?php
ingres_connect ($database, $user, $password);

ingres_query ("select * from table");
```

```
while ($row = ingres_fetch_row()) {
    echo $row[1];
    echo $row[2];
}
?>
```

See also **ingres_num_fields()**, **ingres_query()**, **ingres_fetch_array()**, and **ingres_fetch_object()**.

ingres_field_length

(PHP 4 >= 4.0.2)

ingres_field_length -- Get the length of a field

Description

int **ingres_field_length** (int index [, resource link])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ingres_field_length() returns the length of a field. This is the number of bytes used by the server to store the field. For detailed information, see the Ingres/OpenAPI User Guide - Appendix C.

index is the number of the field and must be between 1 and the value given by **ingres_num_fields()**.

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()**, and **ingres_fetch_row()**.

ingres_field_name

(PHP 4 >= 4.0.2)

ingres_field_name -- Get the name of a field in a query result.

Description

string **ingres_field_name** (int index [, resource link])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ingres_field_name() returns the name of a field in a query result, or **FALSE** on failure.

index is the number of the field and must be between 1 and the value given by **ingres_num_fields()**.

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()** and **ingres_fetch_row()**.

ingres_field_nullable

(PHP 4 >= 4.0.2)

ingres_field_nullable -- Test if a field is nullable

Description

bool **ingres_field_nullable** (int index [, resource link])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ingres_field_nullable() returns **TRUE** if the field can be set to the **NULL** value and **FALSE** if it can't.

index is the number of the field and must be between 1 and the value given by **ingres_num_fields()**.

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()**, and **ingres_fetch_row()**.

ingres_field_precision

(PHP 4 >= 4.0.2)

ingres_field_precision -- Get the precision of a field

Description

int **ingres_field_precision** (int index [, resource link])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ingres_field_precision() returns the precision of a field. This value is used only for decimal, float and money SQL data types. For detailed information, see the Ingres/OpenAPI User Guide - Appendix C.

index is the number of the field and must be between 1 and the value given by **ingres_num_fields()**.

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()**, and **ingres_fetch_row()**.

ingres_field_scale

(PHP 4 >= 4.0.2)

ingres_field_scale -- Get the scale of a field

Description

int **ingres_field_scale** (int index [, resource link])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ingres_field_scale() returns the scale of a field. This value is used only for the decimal SQL data type. For detailed information, see the Ingres/OpenAPI User Guide - Appendix C.

index is the number of the field and must be between 1 and the value given by **ingres_num_fields()**.

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()**, and **ingres_fetch_row()**.

ingres_field_type

(PHP 4 >= 4.0.2)

ingres_field_type -- Get the type of a field in a query result

Description

string **ingres_field_type** (int index [, resource link])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ingres_field_type() returns the type of a field in a query result, or **FALSE** on failure. Examples of types returned are "IIAPI_BYTE_TYPE", "IIAPI_CHA_TYPE", "IIAPI_DTE_TYPE", "IIAPI_FLT_TYPE", "IIAPI_INT_TYPE", "IIAPI_VCH_TYPE". Some of these types can map to more than one SQL type depending on the length of the field (see **ingres_field_length()**). For example "IIAPI_FLT_TYPE" can be a float4 or a float8. For detailed information, see the Ingres/OpenAPI User Guide - Appendix C.

index is the number of the field and must be between 1 and the value given by **ingres_num_fields()**.

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()**, and **ingres_fetch_row()**.

ingres_num_fields

(PHP 4 >= 4.0.2)

ingres_num_fields -- Get the number of fields returned by the last query

Description

int **ingres_num_fields** ([resource link])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ingres_num_fields() returns the number of fields in the results returned by the Ingres server after a call to **ingres_query()**

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()**, and **ingres_fetch_row()**.

ingres_num_rows

(PHP 4 >= 4.0.2)

ingres_num_rows -- Get the number of rows affected or returned by the last query

Description

int **ingres_num_rows** ([resource link])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

For delete, insert or update queries, **ingres_num_rows()** returns the number of rows affected by the query. For other queries, **ingres_num_rows()** returns the number of rows in the query's result.

> **Note:** This function is mainly meant to get the number of rows modified in the database. If this function is called before using **ingres_fetch_array()**, **ingres_fetch_object()** or **ingres_fetch_row()** the server will delete the result's data and the script won't be able to get them.
>
> You should instead retrieve the result's data using one of these fetch functions in a loop until it returns **FALSE**, indicating that no more results are available.

See also **ingres_query()**, **ingres_fetch_array()**, **ingres_fetch_object()**, and **ingres_fetch_row()**.

ingres_pconnect

(PHP 4 >= 4.0.2)

ingres_pconnect -- Open a persistent connection to an Ingres II database

Description

resource **ingres_pconnect** ([string database [, string username [, string password]]])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Returns a Ingres II link resource on success, or **FALSE** on failure.

See **ingres_connect()** for parameters details and examples. There are only 2 differences between **ingres_pconnect()** and **ingres_connect()** : First, when connecting, the function will first try to find a (persistent) link that's already opened with the same parameters. If one is found, an identifier for it will be returned instead of opening a new connection. Second, the connection to the Ingres server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (**ingres_close()** will not close links established by **ingres_pconnect()**). This type of link is therefore called 'persistent'.

See also **ingres_connect()** and **ingres_close()**.

ingres_query

(PHP 4 >= 4.0.2)

ingres_query -- Send a SQL query to Ingres II

Description

bool **ingres_query** (string query [, resource link])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Returns **TRUE** on success, or **FALSE** on failure.

ingres_query() sends the given *query* to the Ingres server. This query must be a valid SQL query (see the Ingres SQL reference guide)

The query becomes part of the currently open transaction. If there is no open transaction, **ingres_query()** opens a new transaction. To close the transaction, you can either call **ingres_commit()** to commit the changes made to the database or **ingres_rollback()** to cancel these changes. When the script ends, any open transaction is rolled back (by calling **ingres_rollback()**). You can also use **ingres_autocommit()** before opening a new transaction to have every SQL query immediatly commited.

Some types of SQL queries can't be sent with this function:

- close (see **ingres_close()**)
- commit (see **ingres_commit()**)
- connect (see **ingres_connect()**)
- disconnect (see **ingres_close()**)
- get dbevent
- prepare to commit
- rollback (see **ingres_rollback()**)
- savepoint
- set autocommit (see **ingres_autocommit()**)
- all cursor related queries are unsupported

Example 1. ingres_query() example

```
<?php
ingres_connect ($database, $user, $password);

ingres_query ("select * from table");
while ($row = ingres_fetch_row()) {
    echo $row[1];
```

```
    echo $row[2];
}
?>
```

See also **ingres_fetch_array()**, **ingres_fetch_object()**, **ingres_fetch_row()**, **ingres_commit()**, **ingres_rollback()**, and **ingres_autocommit()**.

ingres_rollback

(PHP 4 >= 4.0.2)

ingres_rollback -- Roll back a transaction

Description

bool **ingres_rollback** ([resource link])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ingres_rollback() rolls back the currently open transaction, actually canceling all changes made to the database during the transaction.

This closes the transaction. A new one can be open by sending a query with **ingres_query()**.

See also **ingres_query()**, **ingres_commit()**, and **ingres_autocommit()**.

XLVI. IRC Gateway Functions

Introduction

With IRCG you can rapidly stream XML data to thousands of concurrently connected users. This can be used to build powerful, extensible interactive platforms such as online games and webchats. IRCG also features support for a non-streaming mode where a helper application reformats incoming data and supplies static file snippets in special formats such as cHTML (i-mode) or WML (WAP). These static files are then delivered by the high-performance web server.

Up to v3, IRCG runs under these platforms:

- AIX
- FreeBSD
- HP-UX
- Irix
- Linux
- Solaris
- Tru64

Installation

Detailed installation instructions can be found here. We urge you to use the provided installation script.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

Predefined Constants

This extension has no constants defined.

Table of Contents

ircg_channel_mode

(PHP 4 >= 4.0.5)

ircg_channel_mode -- Set channel mode flags for user

Description

boolean **ircg_channel_mode** (resource connection, string channel, string mode_spec, string nick)

Set channel mode flags for *channel* on server connected to by *connection*. Mode flags are passed in *mode_spec* and are applied to the user specified by *nick*.

Mode flags are set or cleared by specifying a mode character and prepending it with a plus or minus character, respectively. E.g. operator mode is granted by '+o' and revoked by '-o', as passed as *mode_spec*.

ircg_disconnect

(PHP 4 >= 4.0.4)

ircg_disconnect -- Close connection to server

Description

boolean **ircg_disconnect** (resource connection, string reason)

ircg_disconnect() will close a *connection* to a server previously established with **ircg_pconnect()**.

See also: **ircg_pconnect()**.

ircg_fetch_error_msg

(PHP 4 >= 4.1.0)

ircg_fetch_error_msg -- Returns the error from previous IRCG operation

Description

array **ircg_fetch_error_msg** (resource connection)

ircg_fetch_error_msg() returns the error from a failed connection.

> **Note:** Error code is stored in first array element, error text in second. The error code is equivalent to IRC reply codes as defined by RFC 2812.

Example 1. ircg_fetch_error_msg() example

```
if (!ircg_join ($id, "#php")) {
    $error = ircg_fetch_error_msg($id);
    print ("Can't join channel #php. Error code:
            $error[0] Description: $error[1]");
}
```

ircg_get_username

(PHP 4 >= 4.1.0)

ircg_get_username -- Get username for connection

Description

string **ircg_get_username** (resource connection)

Function **ircg_get_username()** returns the username for the specified connection *connection*. Returns **FALSE** if *connection* died or is not valid.

ircg_html_encode

(PHP 4 >= 4.0.5)

ircg_html_encode -- Encodes HTML preserving output

Description

boolean **ircg_html_encode** (string html_string)

Encodes a HTML string *html_string* for output. This exposes the interface which the IRCG extension uses internally to reformat data coming from an IRC link. The function causes IRC color/font codes to be encoded in HTML and escapes certain entities.

ircg_ignore_add

(PHP 4 >= 4.0.5)

ircg_ignore_add -- Add a user to your ignore list on a server

Description

boolean **ircg_ignore_add** (resource connection, string nick)

This function adds user *nick* to the ignore list of connection *connection*. Afterwards, IRCG will suppress all messages from this user through the associated connection.

See also: **ircg_ignore_del()**.

ircg_ignore_del

(PHP 4 >= 4.0.5)

ircg_ignore_del -- Remove a user from your ignore list on a server

Description

boolean **ircg_ignore_del** (resource connection, string nick)

This function removes user *nick* from the IRCG ignore list associated with *connection*.

See also: **ircg_ignore_add()**.

ircg_is_conn_alive

(PHP 4 >= 4.0.5)

ircg_is_conn_alive -- Check connection status

Description

boolean **ircg_is_conn_alive** (resource connection)

ircg_is_conn_alive() returns **TRUE** if *connection* is still alive and working or **FALSE**, if the connection has died for some reason.

ircg_join

(PHP 4 >= 4.0.4)

ircg_join -- Join a channel on a connected server

Description

boolean **ircg_join** (resource connection, string channel [, string key])

Join the channel *channel* on the server connected to by *connection*. IRCG will optionally pass the room key *key*.

ircg_kick

(PHP 4 >= 4.0.5)

ircg_kick -- Kick a user out of a channel on server

Description

boolean **ircg_kick** (resource connection, string channel, string nick, string reason)

Kick user *nick* from *channel* on server connected to by *connection*. *reason* should give a short message describing why this action was performed.

ircg_lookup_format_messages

(PHP 4 >= 4.0.5)

ircg_lookup_format_messages -- Check for the existence of a format message set

Description

boolean **ircg_lookup_format_messages** (string name)

Check for the existence of the format message set *name*. Sets may be registered with **ircg_register_format_messages()**, a default set named `ircg` is always available. Returns **TRUE**, if the set exists and **FALSE** otherwise.

See also: **ircg_register_format_messages()**

ircg_msg

(PHP 4 >= 4.0.4)

ircg_msg -- Send message to channel or user on server

Description

boolean **ircg_msg** (resource connection, string recipient, string message [, boolean suppress])

ircg_msg() will send the message to a channel or user on the server connected to by *connection*. A *recipient* starting with `#` or `&` will send the *message* to a channel, anything else will be interpreted as a username.

Setting the optional parameter *suppress* to a **TRUE** value will suppress output of your message to your own *connection*. This so-called loopback is necessary, because the IRC server does not echo PRIVMSG commands back to us.

ircg_nick

(PHP 4 >= 4.0.5)

ircg_nick -- Change nickname on server

Description

boolean **ircg_nick** (resource connection, string nick)

Change your nickname on the given *connection* to the one given in *nick*, if possible.

Will return **TRUE** on success and **FALSE** on failure.

ircg_nickname_escape

(PHP 4 >= 4.0.6)

ircg_nickname_escape -- Encode special characters in nickname to be IRC-compliant

Description

string **ircg_nickname_escape** (string nick)

Function **ircg_nickname_escape()** returns an encoded nickname specified by *nick* wich is IRC-compliant.

See also: **ircg_nickname_unescape()**

ircg_nickname_unescape

(PHP 4 >= 4.0.6)

ircg_nickname_unescape -- Decodes encoded nickname

Description

string **ircg_nickname_unescape** (string nick)

Function **ircg_nickname_unescape()** returns a decoded nickname, which is specified in *nick*.

See also: **ircg_nickname_escape()**

ircg_notice

(PHP 4 >= 4.0.5)

ircg_notice -- Send a notice to a user on server

Description

boolean **ircg_notice** (resource connection, string , string message)

This function will send the *message* text to the user *nick* on the server connected to by *connection*. IRC servers and other software will not automatically generate replies to NOTICEs in contrast to other message types.

ircg_part

(PHP 4 >= 4.0.4)

ircg_part -- Leave a channel on server

Description

boolean **ircg_part** (resource connection, string channel)

Leave the channel *channel* on the server connected to by *connection*.

ircg_pconnect

(PHP 4 >= 4.0.4)

ircg_pconnect -- Connect to an IRC server

Description

resource **ircg_pconnect** (string username [, string server_ip [, int server_port [, string msg_format [, array ctcp_messages [, array user_settings]]]]])

ircg_pconnect() will try to establish a connection to an IRC server and return a connection resource handle for further use.

The only mandatory parameter is *username*, this will set your initial nickname on the server. *server_ip* and *server_port* are optional and default to `127.0.0.1` and `6667`.

> **Note:** For now parameter *server_ip* will not do any hostname lookups and will only accept IP addresses in numerical form. DNS lookups are expensive and should be done in the context of IRCG.

You can customize the output of IRC messages and events by selecting a format message set previously created with **ircg_register_format_messages()** by specifying the set's name in *msg_format*.

If you want to handle CTCP messages such as ACTION (/me), you need to define a mapping from CTCP type (e.g. ACTION) to a custom format string. Do this by passing an associative array as *ctcp_messages*. The keys of the array are the CTCP type and the respective value is the format message.

You can define "ident", "password", and "realname" tokens which are sent to the IRC server by setting these in an associative

array. Pass that array as *user_settings*.

See also: **ircg_disconnect()**, **ircg_is_conn_alive()**, **ircg_register_format_messages()**.

ircg_register_format_messages

(PHP 4 >= 4.0.5)

ircg_register_format_messages -- Register a format message set

Description

boolean **ircg_register_format_messages** (string name, array messages)

With **ircg_register_format_messages()** you can customize the way your IRC output looks like or which script functions are invoked on the client side.

- Plain channel message
- Private message received
- Private message sent
- Some user leaves channel
- Some user enters channel
- Some user was kicked from the channel
- Topic has been changed
- Error
- Fatal error
- Join list end(?)
- Self part(?)
- Some user changes his nick
- Some user quits his connection
- Mass join begin
- Mass join element
- Mass join end
- Whois user
- Whois server
- Whois idle
- Whois channel
- Whois end
- Voice status change on user
- Operator status change on user
- Banlist
- Banlist end
- %f - from
- %t - to
- %c - channel

- %r - plain message
- %m - encoded message
- %j - js encoded message
- 1 - mod encode
- 2 - nickname decode

See also: **ircg_lookup_format_messages()**.

ircg_set_current

(PHP 4 >= 4.0.4)

ircg_set_current -- Set current connection for output

Description

boolean **ircg_set_current** (resource connection)

Select the current HTTP connection for output in this execution context. Every output sent from the server connected to by *connection* will be copied to standard output while using default formatting or a format message set specified by **ircg_register_format_messages()**.

See also: **ircg_register_format_messages()**.

ircg_set_file

(PHP 4 >= 4.2.0)

ircg_set_file -- Set logfile for connection

Description

bool **ircg_set_file** (resource connection, string path)

Function **ircg_set_file()** specifies a logfile *path* in which all output from connection *connection* will be logged. Returns **TRUE** on success, otherwise **FALSE**.

ircg_set_on_die

(PHP 4 >= 4.2.0)

ircg_set_on_die -- Set action to be executed when connection dies

Description

bool **ircg_set_on_die** (resource connection, string host, int port, string data)

In case of the termination of connection *connection* IRCG will connect to *host* at *port* (Note: host must be an IPv4 address, IRCG does not resolve host-names due to blocking issues), send *data* to the new host connection and will wait until the remote part closes connection. This can be used to trigger a PHP script for example.

This feature requires IRCG 3.

ircg_topic

(PHP 4 >= 4.0.5)

ircg_topic -- Set topic for channel on server

Description

boolean **ircg_topic** (resource connection, string channel, string new_topic)

Change the topic for channel *channel* on the server connected to by *connection* to *new_topic*.

ircg_whois

(PHP 4 >= 4.0.5)

ircg_whois -- Query server for user information

Description

boolean **ircg_whois** (resource connection, string nick)

Sends a query to the connected server *connection* to ask for information about the specified user *nick*.

XLVII. PHP / Java Integration

Introduction

There are two possible ways to bridge PHP and Java: you can either integrate PHP into a Java Servlet environment, which is the more stable and efficient solution, or integrate Java support into PHP. The former is provided by a SAPI module that interfaces with the Servlet server, the latter by this Java extension.

The Java extension provides a simple and effective means for creating and invoking methods on Java objects from PHP. The JVM is created using JNI, and everything runs in-process.

Warning
This extension is *EXPERIMENTAL*. The behaviour of this extension -- including the names of its functions and anything else documented about this extension -- may change without notice in a future release of PHP. Use this extension at your own risk.

Requirements

You need a Java VM installed on your machine to use this extension.

Installation

To include Java support in your PHP build you must add the option `--with-java[=DIR]` where DIR points to the base install directory of your JDK. This extension can only be built as a shared dl. More build instructions for this extension can be found in `php4/ext/java/README`.

> **Note to Win32 Users:** In order to enable this module on a Windows environment, you must copy *jvm.dll* from the DLL folder of the PHP/Win32 binary package to the SYSTEM32 folder of your windows machine. (Ex:C:\WINNT\SYSTEM32 or C:\WINDOWS\SYSTEM32)

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. Java configuration options

Name	Default	Changeable
java.class.path	NULL	PHP_INI_ALL
java.home	NULL	PHP_INI_ALL
java.library.path	NULL	PHP_INI_ALL
java.library	JAVALIB	PHP_INI_ALL

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Examples

Example 1. Java Example

```
<?php
  // get instance of Java class java.lang.System in PHP
  $system = new Java('java.lang.System');

  // demonstrate property access
  print 'Java version='.$system->getProperty('java.version').' <br>';
  print 'Java vendor=' .$system->getProperty('java.vendor').' <br>';
  print 'OS='.$system->getProperty('os.name').' '.
             $system->getProperty('os.version').' on '.
             $system->getProperty('os.arch').' <br>';

  // java.util.Date example
  $formatter = new Java('java.text.SimpleDateFormat',
                        "EEEE, MMMM dd, yyyy 'at' h:mm:ss a zzzz");

  print $formatter->format(new Java('java.util.Date'));
?>
```

Example 2. AWT Example

```
<?php
  // This example is only intented to be run as a CGI.

  $frame  = new Java('java.awt.Frame', 'PHP');
  $button = new Java('java.awt.Button', 'Hello Java World!');

  $frame->add('North', $button);
  $frame->validate();
  $frame->pack();
  $frame->visible = True;

  $thread = new Java('java.lang.Thread');
  $thread->sleep(10000);

  $frame->dispose();
?>
```

Notes:

- `new Java()` will create an instance of a class if a suitable constructor is available. If no parameters are passed and the default constructor is useful as it provides access to classes like `java.lang.System` which expose most of their functionallity through static methods.
- Accessing a member of an instance will first look for bean properties then public fields. In other words, `print $date.time` will first attempt to be resolved as `$date.getTime()`, then as `$date.time`.
- Both static and instance members can be accessed on an object with the same syntax. Furthermore, if the java object is of type `java.lang.Class`, then static members of the class (fields and methods) can be accessed.

- Exceptions raised result in PHP warnings, and **NULL** results. The warnings may be eliminated by prefixing the method call with an "@" sign. The following APIs may be used to retrieve and reset the last error:
  - **java_last_exception_get()**
  - **java_last_exception_clear()**
- Overload resolution is in general a hard problem given the differences in types between the two languages. The PHP Java extension employs a simple, but fairly effective, metric for determining which overload is the best match.

  Additionally, method names in PHP are not case sensitive, potentially increasing the number of overloads to select from.

  Once a method is selected, the parameters are cooerced if necessary, possibly with a loss of data (example: double precision floating point numbers will be converted to boolean).
- In the tradition of PHP, arrays and hashtables may pretty much be used interchangably. Note that hashtables in PHP may only be indexed by integers or strings; and that arrays of primitive types in Java can not be sparse. Also note that these constructs are passed by value, so may be expensive in terms of memory and time.

Java Servlet SAPI

The Java Servlet SAPI builds upon the mechanism defined by the Java extension to enable the entire PHP processor to be run as a servlet. The primary advanatage of this from a PHP perspective is that web servers which support servlets typically take great care in pooling and reusing JVMs. Build instructions for the Servlet SAPI module can be found in `php4/sapi/README`. Notes:

- While this code is intended to be able to run on any servlet engine, it has only been tested on Apache's Jakarta/tomcat to date. Bug reports, success stories and/or patches required to get this code to run on other engines would be appreciated.
- PHP has a habit of changing the working directory. sapi/servlet will eventually change it back, but while PHP is running the servlet engine may not be able to load any classes from the CLASSPATH which are specified using a relative directory syntax, or find the work directory used for administration and JSP compilation tasks.

Table of Contents

java_last_exception_clear

(PHP 4 >= 4.0.2)

java_last_exception_clear -- Clear last Java exception

Description

void **java_last_exception_clear** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

See **java_last_exception_get()** for an example.

java_last_exception_get

(PHP 4 >= 4.0.2)

java_last_exception_get -- Get last Java exception

Description

exception **java_last_exception_get** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

The following example demonstrates the usage of Java's exception handler from within PHP:

Example 1. Java exception handler

```
<?php
  $stack = new Java('java.util.Stack');
  $stack->push(1);

  // This should succeed
  $result = $stack->pop();
  $ex = java_last_exception_get();
  if (!$ex) print "$result\n";

  // This should fail (error suppressed by @)
  $result = @$stack->pop();
  $ex = java_last_exception_get();
  if ($ex) print $ex->toString();

  // Clear last exception
  java_last_exception_clear();
?>
```

XLVIII. LDAP functions

Introduction

LDAP is the Lightweight Directory Access Protocol, and is a protocol used to access "Directory Servers". The Directory is a special kind of database that holds information in a tree structure.

The concept is similar to your hard disk directory structure, except that in this context, the root directory is "The world" and the first level subdirectories are "countries". Lower levels of the directory structure contain entries for companies, organisations or places, while yet lower still we find directory entries for people, and perhaps equipment or documents.

To refer to a file in a subdirectory on your hard disk, you might use something like:

```
/usr/local/myapp/docs
```

The forwards slash marks each division in the reference, and the sequence is read from left to right.

The equivalent to the fully qualified file reference in LDAP is the "distinguished name", referred to simply as "dn". An example dn might be:

```
cn=John Smith,ou=Accounts,o=My Company,c=US
```

The comma marks each division in the reference, and the sequence is read from right to left. You would read this dn as:

```
country = US
organization = My Company
organizationalUnit = Accounts
commonName = John Smith
```

In the same way as there are no hard rules about how you organise the directory structure of a hard disk, a directory server manager can set up any structure that is meaningful for the purpose. However, there are some conventions that are used. The message is that you can not write code to access a directory server unless you know something about its structure, any more than you can use a database without some knowledge of what is available.

Lots of information about LDAP can be found at

- Netscape
- University of Michigan
- OpenLDAP Project

- LDAP World

The Netscape SDK contains a helpful Programmer's Guide in HTML format.

Requirements

You will need to get and compile LDAP client libraries from either the University of Michigan ldap-3.3 package or the Netscape Directory SDK 3.0 to compile PHP with LDAP support.

Installation

LDAP support in PHP is not enabled by default. You will need to use the `--with-ldap[=DIR]` configuration option when compiling PHP to enable LDAP support. DIR is the LDAP base install directory.

> **Note to Win32 Users:** In order to enable this module on a Windows environment, you must copy *libsasl.dll* from the DLL folder of the PHP/Win32 binary package to the SYSTEM32 folder of your windows machine. (Ex: C:\WINNT\SYSTEM32 or C:\WINDOWS\SYSTEM32)

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. LDAP configuration options

Name	Default	Changeable
ldap.max_links	"-1"	PHP_INI_SYSTEM

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

LDAP_DEREF_NEVER (integer)

LDAP_DEREF_SEARCHING (integer)

LDAP_DEREF_FINDING (integer)

LDAP_DEREF_ALWAYS (integer)

LDAP_OPT_DEREF (integer)

LDAP_OPT_SIZELIMIT (integer)

LDAP_OPT_TIMELIMIT (integer)

LDAP_OPT_PROTOCOL_VERSION (integer)

LDAP_OPT_ERROR_NUMBER (integer)

LDAP_OPT_REFERRALS (integer)

LDAP_OPT_RESTART (integer)

LDAP_OPT_HOST_NAME (integer)

LDAP_OPT_ERROR_STRING (integer)

LDAP_OPT_MATCHED_DN (integer)

LDAP_OPT_SERVER_CONTROLS (integer)

LDAP_OPT_CLIENT_CONTROLS (integer)

LDAP_OPT_DEBUG_LEVEL (integer)

GSLC_SSL_NO_AUTH (integer)

GSLC_SSL_ONEWAY_AUTH (integer)

GSLC_SSL_TWOWAY_AUTH (integer)

Examples

Retrieve information for all entries where the surname starts with "S" from a directory server, displaying an extract with name and email address.

Example 1. LDAP search example

```
<?php
// basic sequence with LDAP is connect, bind, search, interpret search
// result, close connection

echo "<h3>LDAP query test</h3>";
echo "Connecting ...";
$ds=ldap_connect("localhost");  // must be a valid LDAP server!
echo "connect result is ".$ds."<p>";

if ($ds) {
    echo "Binding ...";
    $r=ldap_bind($ds);     // this is an "anonymous" bind, typically
                           // read-only access
    echo "Bind result is ".$r."<p>";

    echo "Searching for (sn=S*) ...";
    // Search surname entry
    $sr=ldap_search($ds,"o=My Company, c=US", "sn=S*");
    echo "Search result is ".$sr."<p>";

    echo "Number of entires returned is ".ldap_count_entries($ds,$sr)."<p>";

    echo "Getting entries ...<p>";
    $info = ldap_get_entries($ds, $sr);
    echo "Data for ".$info["count"]." items returned:<p>";

    for ($i=0; $i<$info["count"]; $i++) {
        echo "dn is: ". $info[$i]["dn"] ."<br>";
        echo "first cn entry is: ". $info[$i]["cn"][0] ."<br>";
        echo "first email entry is: ". $info[$i]["mail"][0] ."<p>";
    }

    echo "Closing connection";
    ldap_close($ds);

} else {
    echo "<h4>Unable to connect to LDAP server</h4>";
}
?>
```

Using the PHP LDAP calls

Before you can use the LDAP calls you will need to know ..

- The name or address of the directory server you will use
- The "base dn" of the server (the part of the world directory that is held on this server, which could be "o=My

Company,c=US")

- Whether you need a password to access the server (many servers will provide read access for an "anonymous bind" but require a password for anything else)

The typical sequence of LDAP calls you will make in an application will follow this pattern:

```
ldap_connect()    // establish connection to server
  |
ldap_bind()       // anonymous or authenticated "login"
  |
do something like search or update the directory
and display the results
  |
ldap_close()      // "logout"
```

Table of Contents

ldap_8859_to_t61

(PHP 4 >= 4.0.2)

ldap_8859_to_t61 -- Translate 8859 characters to t61 characters

Description

string **ldap_8859_to_t61** (string value)

Warning
This function is currently not documented; only the argument list is available.

ldap_add

(PHP 3, PHP 4)

ldap_add -- Add entries to LDAP directory

Description

bool **ldap_add** (resource link_identifier, string dn, array entry)

Returns **TRUE** on success or **FALSE** on failure.

The **ldap_add()** function is used to add entries in the LDAP directory. The DN of the entry to be added is specified by *dn*. Array *entry* specifies the information about the entry. The values in the entries are indexed by individual attributes. In case of multiple values for an attribute, they are indexed using integers starting with 0.

```
entry["attribute1"] = value
entry["attribute2"][0] = value1
entry["attribute2"][1] = value2
```

Example 1. Complete example with authenticated bind

```
<?php
$ds=ldap_connect("localhost");  // assuming the LDAP server is on this host

if ($ds) {
    // bind with appropriate dn to give update access
    $r=ldap_bind($ds,"cn=root, o=My Company, c=US", "secret");

    // prepare data
    $info["cn"]="John Jones";
    $info["sn"]="Jones";
    $info["mail"]="jonj@here.and.now";
    $info["objectclass"]="person";

    // add data to directory
    $r=ldap_add($ds, "cn=John Jones, o=My Company, c=US", $info);

    ldap_close($ds);
} else {
    echo "Unable to connect to LDAP server";
}
?>
```

ldap_bind

(PHP 3, PHP 4)

ldap_bind -- Bind to LDAP directory

Description

bool **ldap_bind** (resource link_identifier [, string bind_rdn [, string bind_password]])

Binds to the LDAP directory with specified RDN and password. Returns **TRUE** on success or **FALSE** on failure.

ldap_bind() does a bind operation on the directory. *bind_rdn* and *bind_password* are optional. If not specified, anonymous bind is attempted.

Example 1. Using LDAP Bind

```
<?php

// using ldap bind
$ldaprdn  = 'uname';     // ldap rdn or dn
$ldappass = 'password';  // associated password

// connect to ldap server
$ldapconn = ldap_connect("ldap.example.com")
    or die("Could not connect to LDAP server.");

if ($ldapconn) {

    // binding to ldap server
    $ldapbind = ldap_bind($ldapconn, $ldaprdn, $ldappass);

    // verify binding
    if ($ldapbind) {
        echo "LDAP bind successful...";
    } else {
        echo "LDAP bind failed...";
    }

}

?>
```

Example 2. Using LDAP Bind Anonymously

```
<?php

//using ldap bind anonymously

// connect to ldap server
$ldapconn = ldap_connect("ldap.example.com")
    or die("Could not connect to LDAP server.");

if ($ldapconn) {

    // binding anonymously
    $ldapbind = ldap_bind($ldapconn);

    if ($ldapbind) {
        echo "LDAP bind anonymous successful...";
    } else {
        echo "LDAP bind anonymous failed...";
    }

}

?>
```

ldap_close

(PHP 3, PHP 4)

ldap_close -- Close link to LDAP server

Description

bool **ldap_close** (resource link_identifier)

Returns **TRUE** on success or **FALSE** on failure.

ldap_close() closes the link to the LDAP server that's associated with the specified *link_identifier*.

This call is internally identical to **ldap_unbind()**. The LDAP API uses the call **ldap_unbind()**, so perhaps you should use this in preference to **ldap_close()**.

> **Note:** This function is an alias of **ldap_unbind()**.

ldap_compare

(PHP 4 >= 4.0.2)

ldap_compare -- Compare value of attribute found in entry specified with DN

Description

bool **ldap_compare** (resource link_identifier, string dn, string attribute, string value)

Returns **TRUE** if *value* matches otherwise returns **FALSE**. Returns -1 on error.

ldap_compare() is used to compare *value* of *attribute* to value of same attribute in LDAP directory entry specified with *dn*.

The following example demonstrates how to check whether or not given password matches the one defined in DN specified entry.

Example 1. Complete example of password check

```
<?php

$ds=ldap_connect("localhost");  // assuming the LDAP server is on this host

if ($ds) {

    // bind
    if(ldap_bind($ds)) {

        // prepare data
        $dn = "cn=Matti Meikku, ou=My Unit, o=My Company, c=FI";
        $value = "secretpassword";
        $attr = "password";

        // compare value
        $r=ldap_compare($ds, $dn, $attr, $value);

        if ($r === -1) {
            echo "Error: ".ldap_error($ds);
        } elseif ($r === TRUE) {
            echo "Password correct.";
        } elseif ($r === FALSE) {
            echo "Wrong guess! Password incorrect.";
        }

    } else {
        echo "Unable to bind to LDAP server.";
    }

    ldap_close($ds);

} else {
    echo "Unable to connect to LDAP server.";
}
?>
```

Warning
ldap_compare() can NOT be used to compare BINARY values!

Note: This function was added in 4.0.2.

ldap_connect

(PHP 3, PHP 4)

ldap_connect -- Connect to an LDAP server

Description

resource **ldap_connect** ([string hostname [, int port]])

Returns a positive LDAP link identifier on success, or **FALSE** on error.

ldap_connect() establishes a connection to a LDAP server on a specified *hostname* and *port*. Both the arguments are optional. If no arguments are specified then the link identifier of the already opened link will be returned. If only *hostname* is specified, then the port defaults to 389.

If you are using OpenLDAP 2.x.x you can specify a URL instead of the hostname. To use LDAP with SSL, compile OpenLDAP 2.x.x with SSL support, configure PHP with SSL, and use ldaps://hostname/ as host parameter. The port parameter is not used when using URLs.

Note: URL and SSL support were added in 4.0.4.

Example 1. Example of connecting to LDAP server.

```
<?php

// LDAP variables
$ldaphost = "ldap.example.com";  // your ldap servers
$ldapport = 389;                 // your ldap server's port number

// Connecting to LDAP
$ldapconn = ldap_connect( $ldaphost, $ldapport )
          or die( "Could not connect to {$ldaphost}" );

?>
```

Example 2. Example of connecting securely to LDAP server.

```
<?php

// make sure your host is the correct one
// that you issued your secure certificate to
$ldaphost = "ldaps://ldap.example.com/";

// Connecting to LDAP
$ldapconn = ldap_connect( $ldaphost )
          or die( "Could not connect to {$ldaphost}" );

?>
```

ldap_count_entries

(PHP 3, PHP 4)

ldap_count_entries -- Count the number of entries in a search

Description

int **ldap_count_entries** (resource link_identifier, resource result_identifier)

Returns number of entries in the result or **FALSE** on error.

ldap_count_entries() returns the number of entries stored in the result of previous search operations. *result_identifier* identifies the internal ldap result.

ldap_delete

(PHP 3, PHP 4)

ldap_delete -- Delete an entry from a directory

Description

bool **ldap_delete** (resource link_identifier, string dn)

Returns **TRUE** on success or **FALSE** on failure.

ldap_delete() function delete a particular entry in LDAP directory specified by *dn*.

ldap_dn2ufn

(PHP 3, PHP 4)

ldap_dn2ufn -- Convert DN to User Friendly Naming format

Description

string **ldap_dn2ufn** (string dn)

ldap_dn2ufn() function is used to turn a DN, specified by *dn*, into a more user-friendly form, stripping off type names.

ldap_err2str

(PHP 3>= 3.0.13, PHP 4)

ldap_err2str -- Convert LDAP error number into string error message

Description

string **ldap_err2str** (int errno)

Returns string error message.

This function returns the string error message explaining the error number *errno*. While LDAP errno numbers are standardized, different libraries return different or even localized textual error messages. Never check for a specific error message text, but always use an error number to check.

See also **ldap_errno()** and **ldap_error()**.

Example 1. Enumerating all LDAP error messages

```
<?php
  for($i=0; $i<100; $i++) {
    printf("Error $i: %s<br>\n", ldap_err2str($i));
  }
?>
```

ldap_errno

(PHP 3>= 3.0.12, PHP 4)

ldap_errno -- Return the LDAP error number of the last LDAP command

Description

int **ldap_errno** (resource link_identifier)

Return the LDAP error number of the last LDAP command for this link.

This function returns the standardized error number returned by the last LDAP command for the given *link_identifier*. This number can be converted into a textual error message using **ldap_err2str()**.

Unless you lower your warning level in your `php.ini` sufficiently or prefix your LDAP commands with @ (at) characters to suppress warning output, the errors generated will also show up in your HTML output.

Example 1. Generating and catching an error

```
<?php
// This example contains an error, which we will catch.
$ld = ldap_connect("localhost");
$bind = ldap_bind($ld);
// syntax error in filter expression (errno 87),
// must be "objectclass=*" to work.
$res =  @ldap_search($ld, "o=Myorg, c=DE", "objectclass");
if (!$res) {
    printf("LDAP-Errno: %s<br>\n", ldap_errno($ld));
    printf("LDAP-Error: %s<br>\n", ldap_error($ld));
    die("Argh!<br>\n");
}
$info = ldap_get_entries($ld, $res);
printf("%d matching entries.<br>\n", $info["count"]);
?>
```

See also **ldap_err2str()** and **ldap_error()**.

ldap_error

(PHP 3>= 3.0.12, PHP 4)

ldap_error -- Return the LDAP error message of the last LDAP command

Description

string **ldap_error** (resource link_identifier)

Returns string error message.

This function returns the string error message explaining the error generated by the last LDAP command for the given *link_identifier* While LDAP errno numbers are standardized, different libraries return different or even localized textual error messages. Never check for a specific error message text, but always use an error number to check.

Unless you lower your warning level in your `php.ini` sufficiently or prefix your LDAP commands with `@` (at) characters to suppress warning output, the errors generated will also show up in your HTML output.

See also **ldap_err2str()** and **ldap_errno()**.

ldap_explode_dn

(PHP 3, PHP 4)

ldap_explode_dn -- Splits DN into its component parts

Description

array **ldap_explode_dn** (string dn, int with_attrib)

ldap_explode_dn() function is used to split the DN returned by **ldap_get_dn()** and breaks it up into its component parts. Each part is known as Relative Distinguished Name, or RDN. **ldap_explode_dn()** returns an array of all those components. *with_attrib* is used to request if the RDNs are returned with only values or their attributes as well. To get RDNs with the attributes (i.e. in attribute=value format) set *with_attrib* to 0 and to get only values set it to 1.

ldap_first_attribute

(PHP 3, PHP 4)

ldap_first_attribute -- Return first attribute

Description

string **ldap_first_attribute** (resource link_identifier, resource result_entry_identifier, int ber_identifier)

Returns the first attribute in the entry on success and **FALSE** on error.

Similar to reading entries, attributes are also read one by one from a particular entry. **ldap_first_attribute()** returns the first attribute in the entry pointed by the *result_entry_identifier*. Remaining attributes are retrieved by calling **ldap_next_attribute()** successively. *ber_identifier* is the identifier to internal memory location pointer. It is passed by reference. The same *ber_identifier* is passed to the **ldap_next_attribute()** function, which modifies that pointer.

See also **ldap_get_attributes()**

ldap_first_entry

(PHP 3, PHP 4)

ldap_first_entry -- Return first result id

Description

resource **ldap_first_entry** (resource link_identifier, resource result_identifier)

Returns the result entry identifier for the first entry on success and **FALSE** on error.

Entries in the LDAP result are read sequentially using the **ldap_first_entry()** and **ldap_next_entry()** functions. **ldap_first_entry()**

returns the entry identifier for first entry in the result. This entry identifier is then supplied to **ldap_next_entry()** routine to get successive entries from the result.

See also **ldap_get_entries()**.

ldap_first_reference

(PHP 4 >= 4.0.5)

ldap_first_reference -- Return first reference

Description

resource **ldap_first_reference** (resource link, resource result)

Warning
This function is currently not documented; only the argument list is available.

ldap_free_result

(PHP 3, PHP 4)

ldap_free_result -- Free result memory

Description

bool **ldap_free_result** (resource result_identifier)

Returns **TRUE** on success or **FALSE** on failure.

ldap_free_result() frees up the memory allocated internally to store the result and pointed by the *result_identifier*. All result memory will be automatically freed when the script terminates.

Typically all the memory allocated for the ldap result gets freed at the end of the script. In case the script is making successive searches which return large result sets, **ldap_free_result()** could be called to keep the runtime memory usage by the script low.

ldap_get_attributes

(PHP 3, PHP 4)

ldap_get_attributes -- Get attributes from a search result entry

Description

array **ldap_get_attributes** (resource link_identifier, resource result_entry_identifier)

Returns a complete entry information in a multi-dimensional array on success and **FALSE** on error.

ldap_get_attributes() function is used to simplify reading the attributes and values from an entry in the search result. The return value is a multi-dimensional array of attributes and values.

Having located a specific entry in the directory, you can find out what information is held for that entry by using this call. You would use this call for an application which "browses" directory entries and/or where you do not know the structure of the directory entries. In many applications you will be searching for a specific attribute such as an email address or a surname, and won't care what other data is held.

return_value["count"] = number of attributes in the entry
return_value[0] = first attribute
return_value[n] = nth attribute

return_value["attribute"]["count"] = number of values for attribute

return_value["attribute"][0] = first value of the attribute
return_value["attribute"][i] = (i+1)th value of the attribute

Example 1. Show the list of attributes held for a particular directory entry

```
// $ds is the link identifier for the directory

// $sr is a valid search result from a prior call to
// one of the ldap directory search calls

$entry = ldap_first_entry($ds, $sr);

$attrs = ldap_get_attributes($ds, $entry);

echo $attrs["count"]." attributes held for this entry:<p>";

for ($i=0; $i<$attrs["count"]; $i++)
    echo $attrs[$i]."<br>";
```

See also ldap_first_attribute() and ldap_next_attribute()

ldap_get_dn

(PHP 3, PHP 4)

ldap_get_dn -- Get the DN of a result entry

Description

string **ldap_get_dn** (resource link_identifier, resource result_entry_identifier)

Returns the DN of the result entry and FALSE on error.

ldap_get_dn() function is used to find out the DN of an entry in the result.

ldap_get_entries

(PHP 3, PHP 4)

ldap_get_entries -- Get all result entries

Description

array **ldap_get_entries** (resource link_identifier, resource result_identifier)

Returns a complete result information in a multi-dimenasional array on success and FALSE on error.

ldap_get_entries() function is used to simplify reading multiple entries from the result, specified with *result_identifier*, and then reading the attributes and multiple values. The entire information is returned by one function call in a multi-dimensional array. The structure of the array is as follows.

The attribute index is converted to lowercase. (Attributes are case-insensitive for directory servers, but not when used as array indices.)

return_value["count"] = number of entries in the result
return_value[0] : refers to the details of first entry

return_value[i]["dn"] = DN of the ith entry in the result

return_value[i]["count"] = number of attributes in ith entry
return_value[i][j] = jth attribute in the ith entry in the result

return_value[i]["attribute"]["count"] = number of values for
attribute in ith entry
return_value[i]["attribute"][j] = jth value of attribute in ith entry

See also ldap_first_entry() and ldap_next_entry()

ldap_get_option

(PHP 4 >= 4.0.4)

ldap_get_option -- Get the current value for given option

Description

bool **ldap_get_option** (resource link_identifier, int option, mixed retval)

Sets *retval* to the value of the specified option. Returns **TRUE** on success or **FALSE** on failure.

The parameter *option* can be one of: LDAP_OPT_DEREF, LDAP_OPT_SIZELIMIT, LDAP_OPT_TIMELIMIT, LDAP_OPT_PROTOCOL_VERSION, LDAP_OPT_ERROR_NUMBER, LDAP_OPT_REFERRALS, LDAP_OPT_RESTART, LDAP_OPT_HOST_NAME, LDAP_OPT_ERROR_STRING, LDAP_OPT_MATCHED_DN. These are described in draft-ietf-ldapext-ldap-c-api-xx.txt

> **Note:** This function is only available when using OpenLDAP 2.x.x OR Netscape Directory SDK x.x, and was added in PHP 4.0.4

Example 1. Check protocol version

```
// $ds is a valid link identifier for a directory server
if (ldap_get_option($ds, LDAP_OPT_PROTOCOL_VERSION, $version))
    echo "Using protocol version $version";
else
    echo "Unable to determine protocol version";
```

See also **ldap_set_option()**.

ldap_get_values_len

(PHP 3>= 3.0.13, PHP 4)

ldap_get_values_len -- Get all binary values from a result entry

Description

array **ldap_get_values_len** (resource link_identifier, resource result_entry_identifier, string attribute)

Returns an array of values for the attribute on success and **FALSE** on error.

ldap_get_values_len() function is used to read all the values of the attribute in the entry in the result. entry is specified by the *result_entry_identifier*. The number of values can be found by indexing "count" in the resultant array. Individual values are accessed by integer index in the array. The first index is 0.

This function is used exactly like **ldap_get_values()** except that it handles binary data and not string data.

> **Note:** This function was added in 4.0.

ldap_get_values

(PHP 3, PHP 4)

ldap_get_values -- Get all values from a result entry

Description

array **ldap_get_values** (resource link_identifier, resource result_entry_identifier, string attribute)

Returns an array of values for the attribute on success and **FALSE** on error.

ldap_get_values() function is used to read all the values of the attribute in the entry in the result. entry is specified by the *result_entry_identifier*. The number of values can be found by indexing "count" in the resultant array. Individual values are

accessed by integer index in the array. The first index is 0.

This call needs a *result_entry_identifier*, so needs to be preceded by one of the ldap search calls and one of the calls to get an individual entry.

You application will either be hard coded to look for certain attributes (such as "surname" or "mail") or you will have to use the **ldap_get_attributes()** call to work out what attributes exist for a given entry.

LDAP allows more than one entry for an attribute, so it can, for example, store a number of email addresses for one person's directory entry all labeled with the attribute "mail"

return_value["count"] = number of values for attribute
return_value[0] = first value of attribute
return_value[i] = ith value of attribute

Example 1. List all values of the "mail" attribute for a directory entry

```
// $ds is a valid link identifier for a directory server

// $sr is a valid search result from a prior call to
//     one of the ldap directory search calls

// $entry is a valid entry identifier from a prior call to
//        one of the calls that returns a directory entry

$values = ldap_get_values($ds, $entry,"mail");

echo $values["count"]." email addresses for this entry.<p>";

for ($i=0; $i < $values["count"]; $i++)
    echo $values[$i]."<br>";
```

ldap_list

(PHP 3, PHP 4)

ldap_list -- Single-level search

Description

resource **ldap_list** (resource link_identifier, string base_dn, string filter [, array attributes [, int attrsonly [, int sizelimit [, int timelimit [, int deref]]]]])

Returns a search result identifier or **FALSE** on error.

ldap_list() performs the search for a specified *filter* on the directory with the scope LDAP_SCOPE_ONELEVEL.

LDAP_SCOPE_ONELEVEL means that the search should only return information that is at the level immediately below the *base_dn* given in the call. (Equivalent to typing "ls" and getting a list of files and folders in the current working directory.)

This call takes 5 optional parameters. See **ldap_search()** notes.

> **Note:** These optional parameters were added in 4.0.2: *attrsonly*, *sizelimit*, *timelimit*, *deref*.

Example 1. Produce a list of all organizational units of an organization

```
// $ds is a valid link identifier for a directory server

$basedn = "o=My Company, c=US";
$justthese = array("ou");

$sr=ldap_list($ds, $basedn, "ou=*", $justthese);

$info = ldap_get_entries($ds, $sr);

for ($i=0; $i<$info["count"]; $i++)
    echo $info[$i]["ou"][0] ;
```

> **Note:** From 4.0.5 on it's also possible to do parallel searches. See **ldap_search()** for details.

ldap_mod_add

(PHP 3>= 3.0.8, PHP 4)

ldap_mod_add -- Add attribute values to current attributes

Description

bool **ldap_mod_add** (resource link_identifier, string dn, array entry)

Returns **TRUE** on success or **FALSE** on failure.

This function adds attribute(s) to the specified *dn*. It performs the modification at the attribute level as opposed to the object level. Object-level additions are done by the **ldap_add()** function.

ldap_mod_del

(PHP 3>= 3.0.8, PHP 4)

ldap_mod_del -- Delete attribute values from current attributes

Description

bool **ldap_mod_del** (resource link_identifier, string dn, array entry)

Returns **TRUE** on success or **FALSE** on failure.

This function removes attribute(s) from the specified *dn*. It performs the modification at the attribute level as opposed to the object level. Object-level deletions are done by the **ldap_delete()** function.

ldap_mod_replace

(PHP 3>= 3.0.8, PHP 4)

ldap_mod_replace -- Replace attribute values with new ones

Description

bool **ldap_mod_replace** (resoure link_identifier, string dn, array entry)

Returns **TRUE** on success or **FALSE** on failure.

This function replaces attribute(s) from the specified *dn*. It performs the modification at the attribute level as opposed to the object level. Object-level modifications are done by the **ldap_modify()** function.

ldap_modify

(PHP 3, PHP 4)

ldap_modify -- Modify an LDAP entry

Description

bool **ldap_modify** (resource link_identifier, string dn, array entry)

Returns **TRUE** on success or **FALSE** on failure.

ldap_modify() function is used to modify the existing entries in the LDAP directory. The structure of the entry is same as in **ldap_add()**.

ldap_next_attribute

(PHP 3, PHP 4)

ldap_next_attribute -- Get the next attribute in result

Description

string **ldap_next_attribute** (resource link_identifier, resource result_entry_identifier, resource ber_identifier)

Returns the next attribute in an entry on success and **FALSE** on error.

ldap_next_attribute() is called to retrieve the attributes in an entry. The internal state of the pointer is maintained by the *ber_identifier*. It is passed by reference to the function. The first call to **ldap_next_attribute()** is made with the *result_entry_identifier* returned from **ldap_first_attribute()**.

See also **ldap_get_attributes()**

ldap_next_entry

(PHP 3, PHP 4)

ldap_next_entry -- Get next result entry

Description

resource **ldap_next_entry** (resource link_identifier, resource result_entry_identifier)

Returns entry identifier for the next entry in the result whose entries are being read starting with **ldap_first_entry()**. If there are no more entries in the result then it returns **FALSE**.

ldap_next_entry() function is used to retrieve the entries stored in the result. Successive calls to the **ldap_next_entry()** return entries one by one till there are no more entries. The first call to **ldap_next_entry()** is made after the call to **ldap_first_entry()** with the *result_entry_identifier* as returned from the **ldap_first_entry()**.

See also **ldap_get_entries()**

ldap_next_reference

(PHP 4 >= 4.0.5)

ldap_next_reference -- Get next reference

Description

resource **ldap_next_reference** (resource link, resource entry)

Warning
This function is currently not documented; only the argument list is available.

ldap_parse_reference

(PHP 4 >= 4.0.5)

ldap_parse_reference -- Extract information from reference entry

Description

bool **ldap_parse_reference** (resource link, resource entry, array referrals)

Warning
This function is currently not documented; only the argument list is available.

ldap_parse_result

(PHP 4 >= 4.0.5)

ldap_parse_result -- Extract information from result

Description

bool **ldap_parse_result** (resource link, resource result, int errcode, string matcheddn, string errmsg, array referrals)

Warning
This function is currently not documented; only the argument list is available.

ldap_read

(PHP 3, PHP 4)

ldap_read -- Read an entry

Description

resource **ldap_read** (resource link_identifier, string base_dn, string filter [, array attributes [, int attrsonly [, int sizelimit [, int timelimit [, int deref]]]]])

Returns a search result identifier or **FALSE** on error.

ldap_read() performs the search for a specified *filter* on the directory with the scope LDAP_SCOPE_BASE. So it is equivalent to reading an entry from the directory.

An empty filter is not allowed. If you want to retrieve absolutely all information for this entry, use a filter of "objectClass=*". If you know which entry types are used on the directory server, you might use an appropriate filter such as "objectClass=inetOrgPerson".

This call takes 5 optional parameters. See **ldap_search()** notes.

> **Note:** These optional parameters were added in 4.0.2: *attrsonly*, *sizelimit*, *timelimit*, *deref*.

From 4.0.5 on it's also possible to do parallel searches. See **ldap_search()** for details.

ldap_rename

(PHP 4 >= 4.0.5)

ldap_rename -- Modify the name of an entry

Description

bool **ldap_rename** (resource link_identifier, string dn, string newrdn, string newparent, bool deleteoldrdn)

The entry specified by *dn* is renamed/moved. The new RDN is specified by *newrdn* and the new parent/superior entry is specified by *newparent*. If the parameter *deleteoldrdn* is **TRUE** the old RDN value(s) is removed, else the old RDN value(s) is retained as non-distinguished values of the entry. Returns **TRUE** on success or **FALSE** on failure.

> **Note:** This function currently only works with LDAPv3. You may have to use **ldap_set_option()** prior to binding to use LDAPv3. This function is only available when using OpenLDAP 2.x.x OR Netscape Directory SDK x.x, and was added in PHP 4.0.5.

ldap_search

(PHP 3, PHP 4)

ldap_search -- Search LDAP tree

Description

resource **ldap_search** (resource link_identifier, string base_dn, string filter [, array attributes [, int attrsonly [, int sizelimit [, int timelimit [, int deref]]]]])

Returns a search result identifier or **FALSE** on error.

ldap_search() performs the search for a specified filter on the directory with the scope of LDAP_SCOPE_SUBTREE. This is equivalent to searching the entire directory. *base_dn* specifies the base DN for the directory.

There is a optional fourth parameter, that can be added to restrict the attributes and values returned by the server to just those required. This is much more efficient than the default action (which is to return all attributes and their associated values). The use of the fourth parameter should therefore be considered good practice.

The fourth parameter is a standard PHP string array of the required attributes, eg array("mail","sn","cn") Note that the "dn" is always returned irrespective of which attributes types are requested.

Note too that some directory server hosts will be configured to return no more than a preset number of entries. If this occurs, the server will indicate that it has only returned a partial results set. This occurs also if the sixth parameter *sizelimit* has been used to limit the count of fetched entries.

The fifth parameter *attrsonly* should be set to 1 if only attribute types are wanted. If set to 0 both attributes types and attribute values are fetched which is the default behaviour.

With the sixth parameter *sizelimit* it is possible to limit the count of entries fetched. Setting this to 0 means no limit. NOTE: This parameter can NOT override server-side preset sizelimit. You can set it lower though.

The seventh parameter *timelimit* sets the number of seconds how long is spend on the search. Setting this to 0 means no limit. NOTE: This parameter can NOT override server-side preset timelimit. You can set it lower though.

The eigth parameter *deref* specifies how aliases should be handled during the search. It can be one of the following:

- LDAP_DEREF_NEVER - (default) aliases are never dereferenced.
- LDAP_DEREF_SEARCHING - aliases should be dereferenced during the search but not when locating the base object of the search.
- LDAP_DEREF_FINDING - aliases should be dereferenced when locating the base object but not during the search.
- LDAP_DEREF_ALWAYS - aliases should be dereferenced always.

Note: These optional parameters were added in 4.0.2: *attrsonly*, *sizelimit*, *timelimit*, *deref*.

The search filter can be simple or advanced, using boolean operators in the format described in the LDAP doumentation (see the Netscape Directory SDK for full information on filters).

The example below retrieves the organizational unit, surname, given name and email address for all people in "My Company" where the surname or given name contains the substring $person. This example uses a boolean filter to tell the server to look for information in more than one attribute.

Example 1. LDAP search

```
// $ds is a valid link identifier for a directory server

// $person is all or part of a person's name, eg "Jo"

$dn = "o=My Company, c=US";
$filter="(|(sn=$person*)(givenname=$person*))";
$justthese = array( "ou", "sn", "givenname", "mail");

$sr=ldap_search($ds, $dn, $filter, $justthese);

$info = ldap_get_entries($ds, $sr);

print $info["count"]." entries returned<p>";
```

From 4.0.5 on it's also possible to do parallel searches. To do this you use an array of link identifiers, rather than a single identifier, as the first argument. If you don't want the same base DN and the same filter for all the searches, you can also use an array of base DNs and/or an array of filters. Those arrays must be of the same size as the link identifier array since the first entries of the arrays are used for one search, the second entries are used for another, and so on. When doing parallel searches an array of search result identifiers is returned, except in case of error, then the entry corresponding to the search will be **FALSE**. This is very much like the value normally returned, except that a result identifier is always returned when a search was made. There are some rare cases where the normal search returns **FALSE** while the parallel search returns an identifier.

ldap_set_option

(PHP 4 >= 4.0.4)

ldap_set_option -- Set the value of the given option

Description

bool **ldap_set_option** (resource link_identifier, int option, mixed newval)

Sets the value of the specified option to be *newval*. Returns **TRUE** on success or **FALSE** on failure. on error.

The parameter *option* can be one of: LDAP_OPT_DEREF, LDAP_OPT_SIZELIMIT, LDAP_OPT_TIMELIMIT, LDAP_OPT_PROTOCOL_VERSION, LDAP_OPT_ERROR_NUMBER, LDAP_OPT_REFERRALS, LDAP_OPT_RESTART, LDAP_OPT_HOST_NAME, LDAP_OPT_ERROR_STRING, LDAP_OPT_MATCHED_DN, LDAP_OPT_SERVER_CONTROLS, LDAP_OPT_CLIENT_CONTROLS. Here's a brief description, see draft-ietf-ldapext-ldap-c-api-xx.txt for details.

The options LDAP_OPT_DEREF, LDAP_OPT_SIZELIMIT, LDAP_OPT_TIMELIMIT, LDAP_OPT_PROTOCOL_VERSION and LDAP_OPT_ERROR_NUMBER have integer value, LDAP_OPT_REFERRALS and LDAP_OPT_RESTART have boolean value, and the options LDAP_OPT_HOST_NAME, LDAP_OPT_ERROR_STRING and LDAP_OPT_MATCHED_DN have string value. The first example illustrates their use. The options LDAP_OPT_SERVER_CONTROLS and LDAP_OPT_CLIENT_CONTROLS require a list of controls, this means that the value must be an array of controls. A control consists of an *oid* identifying the control, an optional *value*, and an optional flag for *criticality*. In PHP a control is given by an array containing an element with the key *oid* and string value, and two optional elements. The optional elements are key *value* with string value and key *iscritical* with boolean value. *iscritical* defaults to **FALSE** if not supplied. See also the second example below.

> **Note:** This function is only available when using OpenLDAP 2.x.x OR Netscape Directory SDK x.x, and was added in PHP 4.0.4.

Example 1. Set protocol version

```
// $ds is a valid link identifier for a directory server
if (ldap_set_option($ds, LDAP_OPT_PROTOCOL_VERSION, 3))
    echo "Using LDAPv3";
else
    echo "Failed to set protocol version to 3";
```

Example 2. Set server controls

```
// $ds is a valid link identifier for a directory server
// control with no value
$ctrl1 = array("oid" => "1.2.752.58.10.1", "iscritical" => TRUE);
// iscritical defaults to FALSE
$ctrl2 = array("oid" => "1.2.752.58.1.10", "value" => "magic");
// try to set both controls
if (!ldap_set_option($ds, LDAP_OPT_SERVER_CONTROLS, array($ctrl1, $ctrl2)))
    echo "Failed to set server controls";
```

See also **ldap_get_option()**.

ldap_set_rebind_proc

(PHP 4 >= 4.2.0)

ldap_set_rebind_proc -- Set a callback function to do re-binds on referral chasing.

Description

bool **ldap_set_rebind_proc** (resource link, string callback)

Warning
This function is currently not documented; only the argument list is available.

ldap_sort

(PHP 4 >= 4.2.0)

ldap_sort -- Sort LDAP result entries

Description

bool **ldap_sort** (resource link, resource result, string sortfilter)

Warning
This function is currently not documented; only the argument list is available.

ldap_start_tls

(PHP 4 >= 4.2.0)

ldap_start_tls -- Start TLS

Description

bool **ldap_start_tls** (resource link)

Warning
This function is currently not documented; only the argument list is available.

ldap_t61_to_8859

(PHP 4 >= 4.0.2)

ldap_t61_to_8859 -- Translate t61 characters to 8859 characters

Description

string **ldap_t61_to_8859** (string value)

Warning
This function is currently not documented; only the argument list is available.

ldap_unbind

(PHP 3, PHP 4)

ldap_unbind -- Unbind from LDAP directory

Description

bool **ldap_unbind** (resource link_identifier)

Returns **TRUE** on success or **FALSE** on failure.

ldap_unbind() function unbinds from the LDAP directory.

XLIX. Mail functions

Introduction

The **mail()** function allows you to send mail.

Requirements

No external libraries are needed to build this extension.

Installation

There is no installation needed to use these functions; they are part of the PHP core.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. Mail configuration options

Name	Default	Changeable
SMTP	"localhost"	PHP_INI_ALL
smtp_port	"25"	PHP_INI_ALL
sendmail_from	NULL	PHP_INI_ALL
sendmail_path	DEFAULT_SENDMAIL_PATH	PHP_INI_SYSTEM

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Here is a short explanation of the configuration directives.

SMTP **string**

Used under Windows only: DNS name or IP address of the SMTP server PHP should use for mail sent with the **mail()** function.

SMTP **int**

Used under Windows only: Number of the port to connect to the server specified with the `SMTP` setting when sending mail with **mail()**; defaults to 25. Only available since `PHP` 4.3.0.

sendmail_from **string**

Which "From:" mail address should be used in mail sent from PHP under Windows.

sendmail_path **string**

Where the **sendmail** program can be found, usually `/usr/sbin/sendmail` or `/usr/lib/sendmail`. **configure** does an honest attempt of locating this one for you and set a default, but if it fails, you can set it here.

Systems not using sendmail should set this directive to the sendmail wrapper/replacement their mail system offers, if any. For example, Qmail users can normally set it to `/var/qmail/bin/sendmail` or `/var/qmail/bin/qmail-inject`.

qmail-inject does not require any option to process mail correctly.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Table of Contents

ezmlm_hash

(PHP 3>= 3.0.17, PHP 4 >= 4.0.2)

ezmlm_hash -- Calculate the hash value needed by EZMLM

Description

int **ezmlm_hash** (string addr)

ezmlm_hash() calculates the hash value needed when keeping EZMLM mailing lists in a MySQL database.

Example 1. Calculating the hash and subscribing a user

```
$user = "joecool@example.com";
$hash = ezmlm_hash ($user);
$query = sprintf ("INSERT INTO sample VALUES (%s, '%s')", $hash, $user);
$db->query($query); // using PHPLIB db interface
```

mail

(PHP 3, PHP 4)

mail -- send mail

Description

bool **mail** (string to, string subject, string message [, string additional_headers [, string additional_parameters]])

mail() automatically mails the message specified in *message* to the receiver specified in *to*. Multiple recipients can be specified by putting a comma between each address in *to*. Email with attachments and special types of content can be sent using this function. This is accomplished via MIME-encoding - for more information, see this Zend article or the PEAR Mime Classes.

The following RFC's may also be useful: RFC 1896, RFC 2045, RFC 2046, RFC 2047, RFC 2048, and RFC 2049.

mail() returns **TRUE** if the mail was successfully accepted for delivery, **FALSE** otherwise.

Warning
The Windows implementation of **mail()** differs in many ways from the Unix implementation. First, it doesn't use a local binary for composing messages but only operates on direct sockets which means a MTA is needed listening on a network socket (which can either on the localhost or a remote machine). Second, the custom headers like From:, Cc:, Bcc: and Date: are **not** interpreted by the MTA in the first place, but are parsed by PHP. PHP < 4.3 only supported the Cc: header element (and was case-sensitive). PHP >= 4.3 supports all the mentioned header elements and is no longer case-sensitive.

Example 1. Sending mail.

```
mail("joecool@example.com", "My Subject", "Line 1\nLine 2\nLine 3");
```

If a fourth string argument is passed, this string is inserted at the end of the header. This is typically used to add extra headers. Multiple extra headers are separated with a carriage return and newline.

> **Note:** You must use \r\n to separate headers, although some Unix mail transfer agents may work with just a single newline (\n).

Example 2. Sending mail with extra headers.

```
mail("nobody@example.com", "the subject", $message,
     "From: webmaster@$SERVER_NAME\r\n"
    ."Reply-To: webmaster@$SERVER_NAME\r\n"
    ."X-Mailer: PHP/" . phpversion());
```

The *additional_parameters* parameter can be used to pass an additional parameter to the program configured to use when sending mail using the sendmail_path configuration setting. For example, this can be used to set the envelope sender address when using sendmail. You may need to add the user that your web server runs as to your sendmail configuration to prevent a 'X-Warning' header from being added to the message when you set the envelope sender using this method.

Example 3. Sending mail with extra headers and setting an additional command line parameter.

```
mail("nobody@example.com", "the subject", $message,
     "From: webmaster@$SERVER_NAME", "-fwebmaster@$SERVER_NAME");
```

> **Note:** This fifth parameter was added in PHP 4.0.5. Since PHP 4.2.3 this parameter is disabled in safe_mode and the **mail()** function will expose a warning message and return **FALSE** if you're trying to use it.

You can also use simple string building techniques to build complex email messages.

Example 4. Sending complex email.

```
/* recipients */
$to  = "Mary <mary@example.com>" . ", " ; // note the comma
$to .= "Kelly <kelly@example.com>";

/* subject */
$subject = "Birthday Reminders for August";

/* message */
$message = '
<html>
<head>
 <title>Birthday Reminders for August</title>
</head>
<body>
<p>Here are the birthdays upcoming in August!</p>
<table>
 <tr>
  <th>Person</th><th>Day</th><th>Month</th><th>Year</th>
 </tr>
 <tr>
  <td>Joe</td><td>3rd</td><td>August</td><td>1970</td>
 </tr>
 <tr>
  <td>Sally</td><td>17th</td><td>August</td><td>1973</td>
 </tr>
</table>
</body>
</html>
';

/* To send HTML mail, you can set the Content-type header. */
$headers  = "MIME-Version: 1.0\r\n";
$headers .= "Content-type: text/html; charset=iso-8859-1\r\n";

/* additional headers */
$headers .= "From: Birthday Reminder <birthday@example.com>\r\n";

$headers .= "Cc: birthdayarchive@example.com\r\n";
$headers .= "Bcc: birthdaycheck@example.com\r\n";

/* and now mail it */
mail($to, $subject, $message, $headers);
```

> **Note:** Make sure you do not have any newline characters in the *to* or *subject*, or the mail may not be sent properly.

> **Note:** The *to* parameter cannot be an address in the form of "Something <someone@example.com>". The mail command will not parse this properly while talking with the MTA.

See also **imap_mail()**.

L. mailparse functions

Introduction

Warning
This extension is *EXPERIMENTAL*. The behaviour of this extension -- including the names of its functions and anything else documented about this extension -- may change without notice in a future release of PHP. Use this extension at your own risk.

Table of Contents

mailparse_msg_extract_part_file -- Extracts/decodes a message section, decoding the transfer encoding
mailparse_msg_extract_part -- Extracts/decodes a message section. If callbackfunc is not specified, the contents will be sent to "stdout"
mailparse_msg_free -- Frees a handle allocated by mailparse_msg_crea
mailparse_msg_get_part_data -- Returns an associative array of info about the message
mailparse_msg_get_part -- Returns a handle on a given section in a mimemessage
mailparse_msg_get_structure -- Returns an array of mime section names in the supplied message
mailparse_msg_parse_file -- Parse file and return a resource representing the structure
mailparse_msg_parse -- Incrementally parse data into buffer
mailparse_rfc822_parse_addresses -- Parse addresses and returns a hash containing that data
mailparse_stream_encode -- Streams data from source file pointer, apply encoding and write to destfp
mailparse_uudecode_all -- Scans the data from fp and extract each embedded uuencoded file. Returns an array listing filename information

mailparse_determine_best_xfer_encoding

(4.1.0 - 4.1.2 only)

mailparse_determine_best_xfer_encoding -- Figures out the best way of encoding the content read from the file pointer fp, which must be seek-able

Description

int **mailparse_determine_best_xfer_encoding** (resource fp)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

mailparse_msg_create

(4.1.0 - 4.1.2 only)

mailparse_msg_create -- Returns a handle that can be used to parse a message

Description

int **mailparse_msg_create** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

mailparse_msg_extract_part_file

(4.1.0 - 4.1.2 only)

mailparse_msg_extract_part_file -- Extracts/decodes a message section, decoding the transfer encoding

Description

string **mailparse_msg_extract_part_file** (resource rfc2045, string filename [, string callbackfunc])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

mailparse_msg_extract_part

(4.1.0 - 4.1.2 only)

mailparse_msg_extract_part -- Extracts/decodes a message section. If callbackfunc is not specified, the contents will be sent to "stdout"

Description

void **mailparse_msg_extract_part** (resource rfc2045, string msgbody [, string callbackfunc])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

mailparse_msg_free

(4.1.0 - 4.1.2 only)

mailparse_msg_free -- Frees a handle allocated by mailparse_msg_crea

Description

void **mailparse_msg_free** (resource rfc2045buf)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

mailparse_msg_get_part_data

(4.1.0 - 4.1.2 only)

mailparse_msg_get_part_data -- Returns an associative array of info about the message

Description

array **mailparse_msg_get_part_data** (resource rfc2045)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

mailparse_msg_get_part

(4.1.0 - 4.1.2 only)

mailparse_msg_get_part -- Returns a handle on a given section in a mimemessage

Description

int **mailparse_msg_get_part** (resource rfc2045, string mimesection)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

mailparse_msg_get_structure

(4.1.0 - 4.1.2 only)

mailparse_msg_get_structure -- Returns an array of mime section names in the supplied message

Description

array **mailparse_msg_get_structure** (resource rfc2045)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

mailparse_msg_parse_file

(4.1.0 - 4.1.2 only)

mailparse_msg_parse_file -- Parse file and return a resource representing the structure

Description

resource **mailparse_msg_parse_file** (string filename)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

mailparse_msg_parse

(4.1.0 - 4.1.2 only)

mailparse_msg_parse -- Incrementally parse data into buffer

Description

void **mailparse_msg_parse** (resource rfc2045buf, string data)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

mailparse_rfc822_parse_addresses

(4.1.0 - 4.1.2 only)

mailparse_rfc822_parse_addresses -- Parse addresses and returns a hash containing that data

Description

array **mailparse_rfc822_parse_addresses** (string addresses)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

mailparse_stream_encode

(4.1.0 - 4.1.2 only)

mailparse_stream_encode -- Streams data from source file pointer, apply encoding and write to destfp

Description

bool **mailparse_stream_encode** (resource sourcefp, resource destfp, string encoding)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

mailparse_uudecode_all

(no version information, might be only in CVS)

mailparse_uudecode_all -- Scans the data from fp and extract each embedded uuencoded file. Returns an array listing filename information

Description

array **mailparse_uudecode_all** (resource fp)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

LI. Mathematical Functions

Introduction

These math functions will only handle values within the range of the **integer** and **float** types on your computer (this corresponds currently to the C types long resp. double). If you need to handle bigger numbers, take a look at the arbitrary precision math functions.

Requirements

No external libraries are needed to build this extension.

Installation

There is no installation needed to use these functions; they are part of the PHP core.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are always available as part of the PHP core.

Table 1. Math constants

Constant	Value	Description
M_PI	3.14159265358979323846	Pi
M_E	2.7182818284590452354	e
M_LOG2E	1.4426950408889634074	log_2 e
M_LOG10E	0.43429448190325182765	log_10 e
M_LN2	0.69314718055994530942	log_e 2

Constant	Value	Description
M_LN10	2.302585092994045684O2	log_e 10
M_PI_2	1.5707963267948966192	pi/2
M_PI_4	0.7853981633974830962	pi/4
M_1_PI	0.31830988618379067154	1/pi
M_2_PI	0.63661977236758134308	2/pi
M_SQRTPI	1.77245385090551602729	sqrt(pi) [4.0.2]
M_2_SQRTPI	1.12837916709551257390	2/sqrt(pi)
M_SQRT2	1.41421356237309504880	sqrt(2)
M_SQRT3	1.73205080756887729352	sqrt(3) [4.0.2]
M_SQRT1_2	0.70710678118654752440	1/sqrt(2)
M_LNPI	1.14472988584940017414	log_e(pi) [4.0.2]
M_EULER	0.57721566490153286061	Euler constant [4.0.2]

Only M_PI is available in PHP versions up to and including PHP 4.0.0. All other constants are available starting with PHP 4.0.0. Constants labeled [4.0.2] were added in PHP 4.0.2.

Table of Contents

abs

(PHP 3, PHP 4)

abs -- Absolute value

Description

mixed **abs** (mixed number)

Returns the absolute value of *number*. If the argument *number* is of type **float**, the return type is also **float**, otherwise it is **integer** (as **float** usually has a bigger value range than **integer**).

Example 1. abs() example

```
$abs = abs(-4.2); // $abs = 4.2; (double/float)
$abs2 = abs(5);   // $abs2 = 5; (integer)
$abs3 = abs(-5);  // $abs3 = 5; (integer)
```

acos

(PHP 3, PHP 4)

acos -- Arc cosine

Description

float **acos** (float arg)

Returns the arc cosine of *arg* in radians. **acos()** is the complementary function of **cos()**, which means that `a==cos(acos(a))` for every value of a that is within **acos()**' range.

See also **acosh()**, **asin()** and **atan()**.

acosh

(PHP 4 >= 4.1.0)

acosh -- Inverse hyperbolic cosine

Description

float **acosh** (float arg)

Returns the inverse hyperbolic cosine of *arg*, i.e. the value whose hyperbolic cosine is *arg*.

> **Note:** This function is not implemented on Windows platforms.

See also **acos()**, **asinh()** and **atanh()**.

asin

(PHP 3, PHP 4)

asin -- Arc sine

Description

float **asin** (float arg)

Returns the arc sine of *arg* in radians. **asin()** is the complementary function of **sin()**, which means that `a==sin(asin(a))` for every value of a that is within **asin()**'s range.

See also **asinh()**, **acos()** and **atan()**.

asinh

(PHP 4 >= 4.1.0)

asinh -- Inverse hyperbolic sine

Description

float **asinh** (float arg)

Returns the inverse hyperbolic sine of *arg*, i.e. the value whose hyperbolic sine is *arg*.

> **Note:** This function is not implemented on Windows platforms.

See also **asin()**, **acosh()** and **atanh()**.

atan2

(PHP 3>= 3.0.5, PHP 4)

atan2 -- arc tangent of two variables

Description

float **atan2** (float y, float x)

This function calculates the arc tangent of the two variables *x* and *y*. It is similar to calculating the arc tangent of *y* / *x*, except that the signs of both arguments are used to determine the quadrant of the result.

The function returns the result in radians, which is between -PI and PI (inclusive).

See also **acos()** and **atan()**.

atan

(PHP 3, PHP 4)

atan -- Arc tangent

Description

float **atan** (float arg)

Returns the arc tangent of *arg* in radians. **atan()** is the complementary function of **tan()**, which means that `a==tan(atan(a))` for every value of a that is within **atan()**'s range.

See also **atanh()**, **asin()** and **acos()**.

atanh

(PHP 4 >= 4.1.0)

atanh -- Inverse hyperbolic tangent

Description

float **atanh** (float arg)

Returns the inverse hyperbolic tangent of *arg*, i.e. the value whose hyperbolic tangent is *arg*.

Note: This function is not implemented on Windows platforms.

See also **atan()**, **asinh()** and **acosh()**.

base_convert

(PHP 3>= 3.0.6, PHP 4)

base_convert -- Convert a number between arbitrary bases

Description

string **base_convert** (string number, int frombase, int tobase)

Returns a string containing *number* represented in base *tobase*. The base in which *number* is given is specified in *frombase*. Both *frombase* and *tobase* have to be between 2 and 36, inclusive. Digits in numbers with a base higher than 10 will be represented with the letters a-z, with a meaning 10, b meaning 11 and z meaning 35.

Example 1. base_convert()

```
$binary = base_convert ($hexadecimal, 16, 2);
```

bindec

(PHP 3, PHP 4)

bindec -- Binary to decimal

Description

int **bindec** (string binary_string)

Returns the decimal equivalent of the binary number represented by the *binary_string* argument.

bindec() converts a binary number to an **integer**. The largest number that can be converted is 31 bits of 1's or 2147483647 in decimal.

See also: **decbin()**.

ceil

(PHP 3, PHP 4)

ceil -- Round fractions up

Description

float **ceil** (float value)

Returns the next highest integer value by rounding up *value* if necessary. The return value of **ceil()** is still of type **float** as the value range of **float** is usually bigger than that of **integer**.

Example 1. ceil() example

```
echo ceil(4.3);    // 5
echo ceil(9.999);  // 10
```

See also **floor()** and **round()**.

cos

(PHP 3, PHP 4)

cos -- Cosine

Description

float **cos** (float arg)

cos() returns the cosine of the *arg* parameter. The *arg* parameter is in radians.

See also **acos()**, **sin()** **tan()** and **deg2rad()**.

cosh

(PHP 4 >= 4.1.0)

cosh -- Hyperbolic cosine

Description

float **cosh** (float arg)

Returns the hyperbolic cosine of *arg*, defined as `(exp(arg) + exp(-arg))/2`.

See also **cos()**, **acosh()**, **sin()** and **tan()**.

decbin

(PHP 3, PHP 4)

decbin -- Decimal to binary

Description

string **decbin** (int number)

Returns a string containing a binary representation of the given *number* argument. The largest number that can be converted is 4294967295 in decimal resulting to a string of 32 1's.

See also: **bindec()**.

dechex

(PHP 3, PHP 4)

dechex -- Decimal to hexadecimal

Description

string **dechex** (int number)

Returns a string containing a hexadecimal representation of the given *number* argument. The largest number that can be converted is 2147483647 in decimal resulting to "7fffffff".

See also **hexdec()**.

decoct

(PHP 3, PHP 4)

decoct -- Decimal to octal

Description

string **decoct** (int number)

Returns a string containing an octal representation of the given *number* argument. The largest number that can be converted is 2147483647 in decimal resulting to "17777777777".

See also **octdec()**.

deg2rad

(PHP 3>= 3.0.4, PHP 4)

deg2rad -- Converts the number in degrees to the radian equivalent

Description

float **deg2rad** (float number)

This function converts *number* from degrees to the radian equivalent.

See also **rad2deg()**.

exp

(PHP 3, PHP 4)

exp -- Calculates the exponent of e (the Neperian or Natural logarithm base)

Description

float **exp** (float arg)

Returns e raised to the power of *arg*.

See also **pow()**.

expm1

(PHP 4 >= 4.1.0)

expm1 -- Returns exp(number) - 1, computed in a way that is accurate even when the value of number is close to zero

Description

float **expm1** (float number)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

floor

(PHP 3, PHP 4)

floor -- Round fractions down

Description

float **floor** (float value)

Returns the next lowest integer value by rounding down *value* if necessary. The return value of **floor()** is still of type **float** because the value range of **float** is usually bigger than that of **integer**.

Example 1. floor() example

```
echo floor(4.3);   // 4
echo floor(9.999); // 9
```

See also **ceil()** and **round()**.

fmod

(PHP 4 >= 4.2.0)

fmod -- Returns the floating point remainder (modulo) of the division of the arguments

Description

float **fmod** (float x, float y)

Returns the floating point remainder of dividing the dividend (*x*) by the divisor (*y*). The reminder (r) is defined as: x = i * y + r, for some integer i. If *y* is non-zero, r has the same sign as *x* and a magnitude less than the magnitude of *y*.

Example 1. Using fmod()

```
$x = 5.7;
$y = 1.3;
$r = fmod($x, $y);
// $r equals 0.5, because 4 * 1.3 + 0.5 = 5.7
```

getrandmax

(PHP 3, PHP 4)

getrandmax -- Show largest possible random value

Description

int **getrandmax** (void)

Returns the maximum value that can be returned by a call to **rand()**.

See also **rand()**, **srand()** and **mt_getrandmax()**.

hexdec

(PHP 3, PHP 4)

hexdec -- Hexadecimal to decimal

Description

int **hexdec** (string hex_string)

Returns the decimal equivalent of the hexadecimal number represented by the *hex_string* argument. **hexdec()** converts a hexadecimal string to a decimal number. The largest number that can be converted is 7fffffff or 2147483647 in decimal.

hexdec() will replace of any non-hexadecimal characters it encounters by `0`. This way, all left zeros are ignored, but right zeros will be valued.

Example 1. hexdec() example

```
var_dump(hexdec("See"));
var_dump(hexdec("ee"));
// both prints "int(238)"

var_dump(hexdec("that"));
var_dump(hexdec("a0"));
// both prints int(160)
```

See also **dechex()**.

hypot

(PHP 4 >= 4.1.0)

hypot -- Returns sqrt(num1*num1 + num2*num2)

Description

float **hypot** (float num1, float num2)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

is_finite

(PHP 4 >= 4.2.0)

is_finite --

Description

bool **is_finite** (float val)

Returns **TRUE** if *val* is a legal finite number within the allowed range for a PHP float on this platform.

is_infinite

(PHP 4 >= 4.2.0)

is_infinite --

Description

bool **is_infinite** (float val)

Returns **TRUE** if *val* is infinite (positive or negative), like the result of `log(0)` or any value too big to fit into a float on this platform.

is_nan

(PHP 4 >= 4.2.0)

is_nan --

Description

bool **is_nan** (float val)

Returns **TRUE** if *val* is 'not a number', like the result of `acos(1.01)`.

lcg_value

(PHP 4)

lcg_value -- Combined linear congruential generator

Description

float **lcg_value** (void)

lcg_value() returns a pseudo random number in the range of (0, 1). The function combines two CGs with periods of 2^31 - 85 and 2^31 - 249. The period of this function is equal to the product of both primes.

log10

(PHP 3, PHP 4)

log10 -- Base-10 logarithm

Description

float **log10** (float arg)

Returns the base-10 logarithm of *arg*.

log1p

(PHP 4 >= 4.1.0)

log1p -- Returns log(1 + number), computed in a way that accurate even when the val ue of number is close to zero

Description

float **log1p** (float number)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

log

(PHP 3, PHP 4)

log -- Natural logarithm

Description

float **log** (float arg [, float base])

If the optional *base* parameter is specified, **log()** returns $\log_{base} arg$, otherwise **log()** returns the natural logarithm of *arg*.

> **Note:** The *base* parameter became available with PHP version 4.3.0.
>
> As always you can calculate the logarithm in base b of a number n, but using the mathematical identity: $\log_b(n) = \log(n)/\log(b)$, where log is the neperian (or natural) logarithm.

max

(PHP 3, PHP 4)

max -- Find highest value

Description

number **max** (mixed arg1, mixed arg2, mixed argn)

max() returns the numerically highest of the parameter values.

If the first parameter is an array, **max()** returns the highest value in that array. If the first parameter is an integer, string or float, you need at least two parameters and **max()** returns the biggest of these values. You can compare an unlimited number of values.

If one or more of the values is a float, all the values will be treated as floats, and a float is returned. If none of the values is a float, all of them will be treated as integers, and an integer is returned.

See also **min()**.

min

(PHP 3, PHP 4)

min -- Find lowest value

Description

number **min** (number arg1, number arg2 [, ...])

number **min** (array numbers)

min() returns the numerically lowest of the parameter values.

In the first variant, you need at least two parameters and **min()** returns the lowest of these values. You can compare an unlimited number of values. If one of the variables is undefined, **min()** will fail.

In the second variant, **min()** returns the lowest value in *numbers*.

If one or more of the values is a **float**, all the values will be treated as floats, and a float is returned. If none of the values is a float, all of them will be treated as **integer**s, and an integer is returned. Upon failure, **min()** returns NULL and an error of level **E_WARNING** is generated.

```
<?php
$a = 4;
$b = 9;
$c = 3;
$arr = array(99, 34, 11);

// You may want to implement your own error checking in
// case of failure (a variable may not be set)
if (!$min_value = @min($a, $b, $c)) {
    echo "Could not get min value, please try again.";
} else {
    echo "min value is $min_value";
}
```

```
print min($arr);  // 11
?>
```

See also **max()**.

mt_getrandmax

(PHP 3>= 3.0.6, PHP 4)

mt_getrandmax -- Show largest possible random value

Description

int **mt_getrandmax** (void)

Returns the maximum value that can be returned by a call to **mt_rand()**.

See also **mt_rand()**, **mt_srand()** and **getrandmax()**.

mt_rand

(PHP 3>= 3.0.6, PHP 4)

mt_rand -- Generate a better random value

Description

int **mt_rand** ([int min, int max])

Many random number generators of older libcs have dubious or unknown characteristics and are slow. By default, PHP uses the libc random number generator with the **rand()** function. The **mt_rand()** function is a drop-in replacement for this. It uses a random number generator with known characteristics using the Mersenne Twister, which will produce random numbers that should be suitable for seeding some kinds of cryptography (see the home page for details) and is four times faster than what the average libc provides.

If called without the optional *min*, *max* arguments **mt_rand()** returns a pseudo-random value between 0 and **RAND_MAX**. If you want a random number between 5 and 15 (inclusive), for example, use `mt_rand (5, 15)`.

In older versions of PHP, you had to seed the random number generator before use with **mt_srand()**. Since 4.2.0 this is no longer necessary.

> **Note:** In versions before 3.0.7 the meaning of *max* was *range*. To get the same results in these versions the short example should be `mt_rand (5, 11)` to get a random number between 5 and 15.

See also **mt_srand()**, **mt_getrandmax()** and **rand()**.

mt_srand

(PHP 3>= 3.0.6, PHP 4)

mt_srand -- Seed the better random number generator

Description

void **mt_srand** (int seed)

Seeds the random number generator with *seed*.

```
// seed with microseconds
function make_seed() {
    list($usec, $sec) = explode(' ', microtime());
    return (float) $sec + ((float) $usec * 100000);
}
```

```
mt_srand(make_seed());
$randval = mt_rand();
```

Note: Since PHP 4.2.0 it's no longer necessary to seed the random number generator before using it.

See also **mt_rand()**, **mt_getrandmax()** and **srand()**.

octdec

(PHP 3, PHP 4)

octdec -- Octal to decimal

Description

int **octdec** (string octal_string)

Returns the decimal equivalent of the octal number represented by the *octal_string* argument. The largest number that can be converted is 17777777777 or 2147483647 in decimal.

See also **decoct()**.

pi

(PHP 3, PHP 4)

pi -- Get value of pi

Description

float **pi** (void)

Returns an approximation of pi. The returned **float** has a precision based on the precision directive in `php.ini`, which defaults to `14`. Also, you can use the `M_PI` constant which yields identical results to **pi()**.

```
echo pi(); // 3.1415926535898
echo M_PI; // 3.1415926535898
```

pow

(PHP 3, PHP 4)

pow -- Exponential expression

Description

number **pow** (number base, number exp)

Returns *base* raised to the power of *exp*. If possible, this function will return an **integer**.

If the power cannot be computed, a warning will be issued, and **pow()** will return **FALSE**.

Example 1. Some examples of pow()

```
<?php

var_dump( pow(2,8) ); // int(256)
echo pow(-1,20); // 1
echo pow(0, 0); // 1

echo pow(-1, 5.5); // error

?>
```

Warning
In PHP 4.0.6 and earlier **pow()** always returned a **float**, and did not issue warnings.

See also **exp()** and **sqrt()**.

rad2deg

(PHP 3>= 3.0.4, PHP 4)

rad2deg -- Converts the radian number to the equivalent number in degrees

Description

float **rad2deg** (float number)

This function converts *number* from radian to degrees.

See also **deg2rad()**.

rand

(PHP 3, PHP 4)

rand -- Generate a random value

Description

int **rand** ([int min, int max])

If called without the optional *min*, *max* arguments **rand()** returns a pseudo-random value between 0 and **RAND_MAX**. If you want a random number between 5 and 15 (inclusive), for example, use `rand (5, 15)`.

In older versions of PHP, you had to seed the random number generator before use with **srand()**. Since 4.2.0 this is no longer necessary.

> **Note:** In versions before 3.0.7 the meaning of *max* was *range*. To get the same results in these versions the short example should be `rand (5, 11)` to get a random number between 5 and 15.

See also **srand()**, **getrandmax()**, and **mt_rand()**.

round

(PHP 3, PHP 4)

round -- Rounds a float

Description

float **round** (float val [, int precision])

Returns the rounded value of *val* to specified *precision* (number of digits after the decimal point). *precision* can also be negative or zero (default).

Caution
PHP doesn't handle strings like `"12,300.2"` correctly by default. See converting from strings.

```
echo round(3.4);         // 3
echo round(3.5);         // 4
echo round(3.6);         // 4
echo round(3.6, 0);      // 4
echo round(1.95583, 2);  // 1.96
echo round(1241757, -3); // 1242000
```

Note: The *precision* parameter is only available in PHP 4.

See also **ceil()** and **floor()**.

sin

(PHP 3, PHP 4)

sin -- Sine

Description

float **sin** (float arg)

sin() returns the sine of the *arg* parameter. The *arg* parameter is in radians.

```
<?php

// Precision depends on your precision directive
print sin(deg2rad(60));  //  0.866025403 ...
print sin(60);           // -0.304810621 ...

?>
```

See also **asin()**, **cos()**, **tan()** and **deg2rad()**.

sinh

(PHP 4 >= 4.1.0)

sinh -- Hyperbolic sine

Description

float **sinh** (float arg)

Returns the hyperbolic sine of *arg*, defined as `(exp(arg) - exp(-arg))/2`.

See also **sin()**, **asinh()**, **cos()** and **tan()**.

sqrt

(PHP 3, PHP 4)

sqrt -- Square root

Description

float **sqrt** (float arg)

Returns the square root of *arg*.

```
<?php
// Precision depends on your precision directive
echo sqrt(9); // 3
echo sqrt(10); // 3.16227766 ...
?>
```

See also **pow()**.

srand

(PHP 3, PHP 4)

srand -- Seed the random number generator

Description

void **srand** (int seed)

Seeds the random number generator with *seed*.

```
// seed with microseconds
function make_seed() {
    list($usec, $sec) = explode(' ', microtime());
    return (float) $sec + ((float) $usec * 100000);
}
srand(make_seed());
$randval = rand();
```

Note: Since PHP 4.2.0 it's no longer necessary to seed the random number generator before using it.

See also **rand()**, **getrandmax()** and **mt_srand()**.

tan

(PHP 3, PHP 4)

tan -- Tangent

Description

float **tan** (float arg)

tan() returns the tangent of the *arg* parameter. The *arg* parameter is in radians.

See also **atan()**, **sin()**, **cos()** and **deg2rad()**.

tanh

(PHP 4 >= 4.1.0)

tanh -- Hyperbolic tangent

Description

float **tanh** (float arg)

Returns the hyperbolic tangent of *arg*, defined as `sinh(arg)/cosh(arg)`.

See also **tan()**, **atanh()**, **sin()** and **cos()**.

LII. Multi-Byte String Functions

Introduction

There are many languages in which all characters can be expressed by single byte. Multi-byte character codes are used to express many characters for many languages. `mbstring` is developed to handle Japanese characters. However, many `mbstring` functions are able to handle character encoding other than Japanese.

A multi-byte character encoding represents single character with consecutive bytes. Some character encoding has shift(escape) sequences to start/end multi-byte character strings. Therefore, a multi-byte character string may be destroyed when it is divided and/or counted unless multi-byte character encoding safe method is used. This module provides multi-byte character safe string functions and other utility functions such as conversion functions.

Since PHP is basically designed for ISO-8859-1, some multi-byte character encoding does not work well with PHP. Therefore, it is important to set `mbstring.internal_encoding` to a character encoding that works with PHP.

PHP4 Character Encoding Requirements

- Per byte encoding
- Single byte characters in range of `00h-7fh` which is compatible with `ASCII`
- Multi-byte characters without `00h-7fh`

These are examples of internal character encoding that works with PHP and does NOT work with PHP.

```
Character encodings work with PHP:
ISO-8859-*, EUC-JP, UTF-8

Character encodings do NOT work with PHP:
JIS, SJIS
```

Character encoding, that does not work with PHP, may be converted with `mbstring`'s HTTP input/output conversion feature/function.

> **Note:** SJIS should not be used for internal encoding unless the reader is familiar with parser/compiler, character encoding and character encoding issues.
>
> **Note:** If you use databases with PHP, it is recommended that you use the same character encoding for both database and `internal encoding` for ease of use and better performance.
>
> If you are using PostgreSQL, it supports character encoding that is different from backend character encoding. See the PostgreSQL manual for details.

Installation

`mbstring` is an extended module. You must enable the module with the `configure` script. Refer to the Install section for details.

The following configure options are related to the `mbstring` module.

- `--enable-mbstring`: Enable `mbstring` functions. This option is required to use `mbstring` functions.

  > **Note:** As of PHP 4.3.0, the option `--enable-mbstring` will be enabled by default and replaced with `--with-mbstring[=LANG]` to support Chinese, Korean and Russian language support. Japanese character encoding is supported by default. If `--with-mbstring=cn` is used, simplified chinese encoding will be supported. If `--with-mbstring=tw` is used, traditional chinese encoding will be supported. If `--with-mbstring=kr` is used, korean encoding will be supported. If `--with-mbstring=ru` is used, russian encoding will be supported. If `--with-mbstring=all` is added, all supported character encoding in mbstring will be enabled, but the binary size of PHP will be maximized because of huge Unicode character maps. Note that Chinese, Korean and Russian encoding is experimentally supported in PHP 4.3.0.

- `--enable-mbstr-enc-trans` : Enable HTTP input character encoding conversion using `mbstring` conversion engine. If this feature is enabled, HTTP input character encoding may be converted to `mbstring.internal_encoding` automatically.

  > **Note:** As of PHP 4.3.0, the option `--enable-mbstr-enc-trans` will be eliminated and replaced with `mbstring.encoding_translation`. HTTP input character encoding conversion is enabled when this is set to `On` (the default is `Off`).

- `--enable-mbregex`: Enable regular expression functions with multibyte character support.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. Multi-Byte String configuration options

Name	Default	Changeable
mbstring.language	NULL	PHP_INI_ALL
mbstring.detect_order	NULL	PHP_INI_ALL
mbstring.http_input	NULL	PHP_INI_ALL
mbstring.http_output	NULL	PHP_INI_ALL

Name	Default	Changeable
mbstring.internal_encoding	NULL	PHP_INI_ALL
mbstring.script_encoding	NULL	PHP_INI_ALL
mbstring.substitute_character	NULL	PHP_INI_ALL
mbstring.func_overload	"0"	PHP_INI_SYSTEM
mbstring.encoding_translation	"0"	PHP_INI_ALL

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Here is a short explanation of the configuration directives.

- `mbstring.language` defines default language used in mbstring. Note that this option defines `mbstring.interanl_encoding` and `mbstring.interanl_encoding` should be placed after `mbstring.language` in `php.ini`
- `mbstring.encoding_translation` enables HTTP input character encoding detection and translation into internal chatacter encoding.
- `mbstring.internal_encoding` defines default internal character encoding.
- `mbstring.http_input` defines default HTTP input character encoding.
- `mbstring.http_output` defines default HTTP output character encoding.
- `mbstring.detect_order` defines default character code detection order. See also **mb_detect_order()**.
- `mbstring.substitute_character` defines character to substitute for invalid character encoding.
- `mbstring.func_overload`overload(replace) single byte functions by mbstring functions. **mail()**, **ereg()**, etc. are overloaded by **mb_send_mail()**, **mb_ereg()**, etc. Possible values are 0, 1, 2, 4 or a combination of them. For example, 7 for overload everything. 0: No overload, 1: Overload **mail()** function, 2: Overload str*() functions, 4: Overload ereg*() functions.

Web Browsers are supposed to use the same character encoding when submitting form. However, browsers may not use the same character encoding. See **mb_http_input()** to detect character encoding used by browsers.

If `enctype` is set to `multipart/form-data` in HTML forms, `mbstring` does not convert character encoding in POST data. The user must convert them in the script, if conversion is needed.

Although, browsers are smart enough to detect character encoding in HTML. `charset` is better to be set in HTTP header. Change `default_charset` according to character encoding.

Example 1. `php.ini` setting example

```
; Set default language
mbstring.language        = English; Set default language to English (default)
mbstring.language        = Japanese; Set default language to Japanese

;; Set default internal encoding
;; Note: Make sure to use character encoding works with PHP
mbstring.internal_encoding    = UTF-8  ; Set internal encoding to UTF-8

;; HTTP input encoding translation is enabled.
mbstring.encoding_translation = On

;; Set default HTTP input character encoding
;; Note: Script cannot change http_input setting.
mbstring.http_input           = pass    ; No conversion.
mbstring.http_input           = auto    ; Set HTTP input to auto
                                ; "auto" is expanded to "ASCII,JIS,UTF-8,EUC-JP,SJIS"
mbstring.http_input           = SJIS    ; Set HTTP2 input to  SJIS
mbstring.http_input           = UTF-8,SJIS,EUC-JP ; Specify order

;; Set default HTTP output character encoding
mbstring.http_output          = pass    ; No conversion
mbstring.http_output          = UTF-8   ; Set HTTP output encoding to UTF-8

;; Set default character encoding detection order
mbstring.detect_order         = auto    ; Set detect order to auto
mbstring.detect_order         = ASCII,JIS,UTF-8,SJIS,EUC-JP ; Specify order

;; Set default substitute character
mbstring.substitute_character = 12307   ; Specify Unicode value
mbstring.substitute_character = none    ; Do not print character
mbstring.substitute_character = long    ; Long Example: U+3000,JIS+7E7E
```

Example 2. `php.ini` setting for `EUC-JP` users

```
;; Disable Output Buffering
```

```
output_buffering      = Off

;; Set HTTP header charset
default_charset       = EUC-JP

;; Set default language to Japanese
mbstring.language = Japanese

;; HTTP input encoding translation is enabled.
mbstring.encoding_translation = On

;; Set HTTP input encoding conversion to auto
mbstring.http_input   = auto

;; Convert HTTP output to EUC-JP
mbstring.http_output  = EUC-JP

;; Set internal encoding to EUC-JP
mbstring.internal_encoding = EUC-JP

;; Do not print invalid characters
mbstring.substitute_character = none
```

Example 3. `php.ini` setting for `SJIS` users

```
;; Enable Output Buffering
output_buffering     = On

;; Set mb_output_handler to enable output conversion
output_handler       = mb_output_handler

;; Set HTTP header charset
default_charset      = Shift_JIS

;; Set default language to Japanese
mbstring.language = Japanese

;; Set http input encoding conversion to auto
mbstring.http_input  = auto

;; Convert to SJIS
mbstring.http_output = SJIS

;; Set internal encoding to EUC-JP
mbstring.internal_encoding = EUC-JP

;; Do not print invalid characters
mbstring.substitute_character = none
```

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

`MB_OVERLOAD_MAIL` (integer)

`MB_OVERLOAD_STRING` (integer)

`MB_OVERLOAD_REGEX` (integer)

HTTP Input and Output

HTTP input/output character encoding conversion may convert binary data also. Users are supposed to control character encoding conversion if binary data is used for HTTP input/output.

If `enctype` for HTML form is set to `multipart/form-data`, `mbstring` does not convert character encoding in POST data. If it is the case, strings are needed to be converted to internal character encoding.

- HTTP Input

  There is no way to control HTTP input character conversion from PHP script. To disable HTTP input character conversion, it has to be done in `php.ini`.

  Example 4. Disable HTTP input conversion in `php.ini`

```
;; Disable HTTP Input conversion
mbstring.http_input = pass
;; Disable HTTP Input conversion (PHP 4.3.0 or higher)
mbstring.encoding_translation = Off
```

  When using PHP as an Apache module, it is possible to override PHP ini setting per Virtual Host in `httpd.conf` or per directory with `.htaccess`. Refer to the Configuration section and Apache Manual for details.

- HTTP Output

  There are several ways to enable output character encoding conversion. One is using `php.ini`, another is using **ob_start()** with **mb_output_handler()** as `ob_start` callback function.

  > **Note:** For PHP3-i18n users, `mbstring`'s output conversion differs from PHP3-i18n. Character encoding is converted using output buffer.

Example 5. `php.ini` setting example

```
;; Enable output character encoding conversion for all PHP pages

;; Enable Output Buffering
output_buffering    = On

;; Set mb_output_handler to enable output conversion
output_handler      = mb_output_handler
```

Example 6. Script example

```
<?php

// Enable output character encoding conversion only for this page

// Set HTTP output character encoding to SJIS
mb_http_output('SJIS');

// Start buffering and specify "mb_output_handler" as
// callback function
ob_start('mb_output_handler');

?>
```

Supported Character Encodings

Currently, the following character encoding is supported by the `mbstring` module. Character encoding may be specified for `mbstring` functions' `encoding` parameter.

The following character encoding is supported in this PHP extension:

`UCS-4`, `UCS-4BE`, `UCS-4LE`, `UCS-2`, `UCS-2BE`, `UCS-2LE`, `UTF-32`, `UTF-32BE`, `UTF-32LE`, `UCS-2LE`, `UTF-16`, `UTF-16BE`, `UTF-16LE`, `UTF-8`, `UTF-7`, `ASCII`, `EUC-JP`, `SJIS`, `eucJP-win`, `SJIS-win`, `ISO-2022-JP`, `JIS`, `ISO-8859-1`, `ISO-8859-2`, `ISO-8859-3`, `ISO-8859-4`, `ISO-8859-5`, `ISO-8859-6`, `ISO-8859-7`, `ISO-8859-8`, `ISO-8859-9`, `ISO-8859-10`, `ISO-8859-13`, `ISO-8859-14`, `ISO-8859-15`, `byte2be`, `byte2le`, `byte4be`, `byte4le`, `BASE64`, `7bit`, `8bit` and `UTF7-IMAP`.

As of PHP 4.3.0, the following character encoding support will be added experimentaly : `EUC-CN`, `CP936`, `HZ`, `EUC-TW`, `CP950`, `BIG-5`, `EUC-KR`, `UHC` (`CP949`), `ISO-2022-KR`, `Windows-1251` (`CP1251`), `Windows-1252` (`CP1252`), `CP866`, `KOI8-R`.

`php.ini` entry, which accepts encoding name, accepts "`auto`" and "`pass`" also. `mbstring` functions, which accepts encoding name, and accepts "`auto`".

If "`pass`" is set, no character encoding conversion is performed.

If "`auto`" is set, it is expanded to "`ASCII,JIS,UTF-8,EUC-JP,SJIS`".

See also **mb_detect_order()**

Note: "Supported character encoding" does not mean that it works as internal character code.

Overloading PHP string functions with multi byte string functions

Because almost PHP application written for language using single-byte character encoding, there are some difficulties for multibyte string handling including japanese. Almost PHP string functions such as **substr()** do not support multibyte string.

Multibyte extension (mbstring) has some PHP string functions with multibyte support (ex. **substr()** supports **mb_substr()**).

Multibyte extension (mbstring) also supports 'function overloading' to add multibyte string functionality without code modification. Using function overloading, some PHP string functions will be oveloaded multibyte string functions. For example, **mb_substr()** is called instead of **substr()** if function overloading is enabled. Function overload makes easy to port application supporting only single-byte encoding for multibyte application.

`mbstring.func_overload` in `php.ini` should be set some positive value to use function overloading. The value should specify the category of overloading functions, sbould be set 1 to enable mail function overloading. 2 to enable string functions, 4 to regular expression functions. For example, if is set for 7, mail, strings, regex functions should be overloaded. The list of overloaded functions are shown in below.

Table 2. Functions to be overloaded

value of mbstring.func_overload	original function	overloaded function
1	**mail()**	**mb_send_mail()**
2	**strlen()**	**mb_strlen()**
2	**strpos()**	**mb_strpos()**
2	**strrpos()**	**mb_strrpos()**
2	**substr()**	**mb_substr()**
2	**strtolower()**	**mb_strtolower()**
2	**strtoupper()**	**mb_strtoupper()**
2	**substr_count()**	**mb_substr_count()**
4	**ereg()**	**mb_ereg()**
4	**eregi()**	**mb_eregi()**
4	**ereg_replace()**	**mb_ereg_replace()**
4	**eregi_replace()**	**mb_eregi_replace()**
4	**split()**	**mb_split()**

Basics of Japanese multi-byte characters

Most Japanese characters need more than 1 byte per character. In addition, several character encoding schemas are used under a Japanese environment. There are EUC-JP, Shift_JIS(SJIS) and ISO-2022-JP(JIS) character encoding. As Unicode becomes popular, UTF-8 is used also. To develop Web applications for a Japanese environment, it is important to use the character set for the task in hand, whether HTTP input/output, RDBMS and E-mail.

- Storage for a character can be up to six bytes
- A multi-byte character is usually twice of the width compared to single-byte characters. Wider characters are called "zen-kaku" - meaning full width, narrower characters are called "han-kaku" - meaning half width. "zen-kaku" characters are usually fixed width.
- Some character encoding defines shift(escape) sequence for entering/exiting multi-byte character strings.
- ISO-2022-JP must be used for SMTP/NNTP.
- "i-mode" web site is supposed to use SJIS.

References

Multi-byte character encoding and its related issues are very complex. It is impossible to cover in sufficient detail here. Please refer to the following URLs and other resources for further readings.

- Unicode/UTF/UCS/etc

  ```
  http://www.unicode.org/
  ```

- Japanese/Korean/Chinese character information

  ```
  ftp://ftp.ora.com/pub/examples/nutshell/ujip/doc/cjk.inf
  ```

Table of Contents

mb_convert_case

(PHP 4 >= 4.3.0)

mb_convert_case -- Perform case folding on a string

Description

string **mb_convert_case** (string str, int mode [, string encoding])

mb_convert_case() returns case folded version of *string* converted in the way specified by *mode*.

mode can be one of MB_CASE_UPPER, MB_CASE_LOWER or MB_CASE_TITLE.

encoding specifies the encoding of *str*; if omitted, the internal character encoding value will be used.

The return value is str with the appropriate case folding applied.

By contrast to the standard case folding functions such as **strtolower()** and **strtoupper()**, case folding is performed on the basis of the Unicode character properties. Thus the behaviour of this function is not affected by locale settings and it can convert any characters that have 'alphabetic' property, such as A-umlaut (Ä).

For more information about the Unicode properties, please see http://www.unicode.org/unicode/reports/tr21/.

Example 1. mb_convert_case() example

```
$str = "mary had a Little lamb and she loved it so";
$str = mb_convert_case($str, MB_CASE_UPPER, "UTF-8");
print $str; # Prints MARY HAD A LITTLE LAMB AND SHE LOVED IT SO
$str = mb_convert_case($str, MB_CASE_TITLE, "UTF-8");
print $str; # Prints Mary Had A Little Lamb And She Loved It So
```

See also **mb_strtolower()**, **mb_strtoupper()**, **strtolower()**, **strtoupper()**.

mb_convert_encoding

(PHP 4 >= 4.0.6)

mb_convert_encoding -- Convert character encoding

Description

string **mb_convert_encoding** (string str, string to-encoding [, mixed from-encoding])

mb_convert_encoding() converts character encoding of string *str* from *from-encoding* to *to-encoding*.

str : String to be converted.

from-encoding is specified by character code name before conversion. it can be array or string - comma separated enumerated list. If it is not specified, the internal encoding will be used.

Example 1. mb_convert_encoding() example

```
/* Convert internal character encoding to SJIS */
$str = mb_convert_encoding($str, "SJIS");

/* Convert EUC-JP to UTF-7 */
$str = mb_convert_encoding($str, "UTF-7", "EUC-JP");

/* Auto detect encoding from JIS, eucjp-win, sjis-win, then convert str to UCS-2LE */
$str = mb_convert_encoding($str, "UCS-2LE", "JIS, eucjp-win, sjis-win");

/* "auto" is expanded to "ASCII,JIS,UTF-8,EUC-JP,SJIS" */
$str = mb_convert_encoding($str, "EUC-JP", "auto");
```

See also: **mb_detect_order()**.

mb_convert_kana

(PHP 4 >= 4.0.6)

mb_convert_kana -- Convert "kana" one from another ("zen-kaku" ,"han-kaku" and more)

Description

string **mb_convert_kana** (string str, string option [, mixed encoding])

mb_convert_kana() performs "han-kaku" - "zen-kaku" conversion for string *str*. It returns converted string. This function is only useful for Japanese.

option is conversion option. Default value is "KV".

encoding is character encoding. If it is omitted, internal character encoding is used.

```
Applicable Conversion Options

       option : Specify with conversion of following options. Default "KV"
       "r" :  Convert "zen-kaku" alphabets to "han-kaku"
       "R" :  Convert "han-kaku" alphabets to "zen-kaku"
       "n" :  Convert "zen-kaku" numbers to "han-kaku"
       "N" :  Convert "han-kaku" numbers to "zen-kaku"
       "a" :  Convert "zen-kaku" alphabets and numbers to "han-kaku"
       "A" :  Convert "han-kaku" alphabets and numbers to "zen-kaku"
       (Characters included in "a", "A" options are
       U+0021 - U+007E excluding U+0022, U+0027, U+005C, U+007E)
       "s" :  Convert "zen-kaku" space to "han-kaku" (U+3000 -> U+0020)
       "S" :  Convert "han-kaku" space to "zen-kaku" (U+0020 -> U+3000)
       "k" :  Convert "zen-kaku kata-kana" to "han-kaku kata-kana"
       "K" :  Convert "han-kaku kata-kana" to "zen-kaku kata-kana"
       "h" :  Convert "zen-kaku hira-gana" to "han-kaku kata-kana"
       "H" :  Convert "han-kaku kata-kana" to "zen-kaku hira-gana"
       "c" :  Convert "zen-kaku kata-kana" to "zen-kaku hira-gana"
       "C" :  Convert "zen-kaku hira-gana" to "zen-kaku kata-kana"
       "V" :  Collapse voiced sound notation and convert them into a character. Use with "K","H"
```

Example 1. mb_convert_kana() example

```
/* Convert all "kana" to "zen-kaku" "kata-kana" */
$str = mb_convert_kana($str, "KVC");

/* Convert "han-kaku" "kata-kana" to "zen-kaku" "kata-kana"
   and "zen-kaku" alpha-numeric to "han-kaku" */
$str = mb_convert_kana($str, "KVa");
```

mb_convert_variables

(PHP 4 >= 4.0.6)

mb_convert_variables -- Convert character code in variable(s)

Description

string **mb_convert_variables** (string to-encoding, mixed from-encoding, mixed vars)

mb_convert_variables() convert character encoding of variables *vars* in encoding *from-encoding* to encoding *to-encoding*. It returns character encoding before conversion for success, **FALSE** for failure.

mb_convert_variables() join strings in Array or Object to detect encoding, since encoding detection tends to fail for short strings. Therefore, it is impossible to mix encoding in single array or object.

It *from-encoding* is specified by array or comma separated string, it tries to detect encoding from *from-coding*. When *encoding* is omitted, detect_order is used.

vars (3rd and larger) is reference to variable to be converted. String, Array and Object are accepted. **mb_convert_variables()** assumes all parameters have the same encoding.

Example 1. mb_convert_variables() example

```
/* Convert variables $post1, $post2 to internal encoding */
$interenc = mb_internal_encoding();
$inputenc = mb_convert_variables($interenc, "ASCII,UTF-8,SJIS-win", $post1, $post2);
```

mb_decode_mimeheader

(PHP 4 >= 4.0.6)

mb_decode_mimeheader -- Decode string in MIME header field

Description

string **mb_decode_mimeheader** (string str)

mb_decode_mimeheader() decodes encoded-word string *str* in MIME header.

It returns decoded string in internal character encoding.

See also **mb_encode_mimeheader()**.

mb_decode_numericentity

(PHP 4 >= 4.0.6)

mb_decode_numericentity -- Decode HTML numeric string reference to character

Description

string **mb_decode_numericentity** (string str, array convmap [, string encoding])

Convert numeric string reference of string *str* in specified block to character. It returns converted string.

array is array to specifies code area to convert.

encoding is character encoding. If it is omitted, internal character encoding is used.

Example 1. *convmap* example

```
$convmap = array (
   int start_code1, int end_code1, int offset1, int mask1,
   int start_code2, int end_code2, int offset2, int mask2,
   ........
   int start_codeN, int end_codeN, int offsetN, int maskN );
// Specify Unicode value for start_codeN and end_codeN
// Add offsetN to value and take bit-wise 'AND' with maskN,
// then convert value to numeric string reference.
```

See also: **mb_encode_numericentity()**.

mb_detect_encoding

(PHP 4 >= 4.0.6)

mb_detect_encoding -- Detect character encoding

Description

string **mb_detect_encoding** (string str [, mixed encoding-list])

mb_detect_encoding() detects character encoding in string *str*. It returns detected character encoding.

encoding-list is list of character encoding. Encoding order may be specified by array or comma separated list string.

If *encoding_list* is omitted, detect_order is used.

Example 1. mb_detect_encoding() example

```
/* Detect character encoding with current detect_order */
echo mb_detect_encoding($str);

/* "auto" is expanded to "ASCII,JIS,UTF-8,EUC-JP,SJIS" */
echo mb_detect_encoding($str, "auto");
```

```
/* Specify encoding_list character encoding by comma separated list */
echo mb_detect_encoding($str, "JIS, eucjp-win, sjis-win");

/* Use array to specify encoding_list  */
$ary[] = "ASCII";
$ary[] = "JIS";
$ary[] = "EUC-JP";
echo mb_detect_encoding($str, $ary);
```

See also: **mb_detect_order()**.

mb_detect_order

(PHP 4 >= 4.0.6)

mb_detect_order -- Set/Get character encoding detection order

Description

array **mb_detect_order** ([mixed encoding-list])

mb_detect_order() sets automatic character encoding detection order to *encoding-list*. It returns **TRUE** for success, **FALSE** for failure.

encoding-list is array or comma separated list of character encoding. ("auto" is expanded to "ASCII, JIS, UTF-8, EUC-JP, SJIS")

If *encoding-list* is omitted, it returns current character encoding detection order as array.

This setting affects **mb_detect_encoding()** and **mb_send_mail()**.

> **Note:** mbstring currently implements following encoding detection filters. If there is a invalid byte sequence for following encoding, encoding detection will fail.
>
> **Note:** UTF-8, UTF-7, ASCII, EUC-JP,SJIS, eucJP-win, SJIS-win, JIS, ISO-2022-JP
>
> For ISO-8859-*, mbstring always detects as ISO-8859-*.
>
> For UTF-16, UTF-32, UCS2 and UCS4, encoding detection will fail always.
>
> **Example 1. Useless detect order example**
>
> ```
> ; Always detect as ISO-8859-1
> detect_order = ISO-8859-1, UTF-8
>
> ; Always detect as UTF-8, since ASCII/UTF-7 values are
> ; valid for UTF-8
> detect_order = UTF-8, ASCII, UTF-7
> ```

Example 2. mb_detect_order() examples

```
/* Set detection order by enumerated list */
mb_detect_order("eucjp-win,sjis-win,UTF-8");

/* Set detection order by array */
$ary[] = "ASCII";
$ary[] = "JIS";
$ary[] = "EUC-JP";
mb_detect_order($ary);

/* Display current detection order */
echo implode(", ", mb_detect_order());
```

See also **mb_internal_encoding()**, **mb_http_input()**, **mb_http_output()** **mb_send_mail()**.

mb_encode_mimeheader

(PHP 4 >= 4.0.6)

mb_encode_mimeheader -- Encode string for MIME header

Description

string **mb_encode_mimeheader** (string str [, string charset [, string transfer-encoding [, string linefeed]]])

mb_encode_mimeheader() converts string *str* to encoded-word for header field. It returns converted string in ASCII encoding.

charset is character encoding name. Default is ISO-2022-JP.

transfer-encoding is transfer encoding. It should be one of "B" (Base64) or "Q" (Quoted-Printable). Default is "B".

linefeed is end of line marker. Default is "\r\n" (CRLF).

Example 1. mb_convert_kana() example

```
$name = ""; // kanji
$mbox = "kru";
$doma = "gtinn.mon";
$addr = mb_encode_mimeheader($name, "UTF-7", "Q") . " <" . $mbox . "@" . $doma . ">";
echo $addr;
```

See also **mb_decode_mimeheader()**.

mb_encode_numericentity

(PHP 4 >= 4.0.6)

mb_encode_numericentity -- Encode character to HTML numeric string reference

Description

string **mb_encode_numericentity** (string str, array convmap [, string encoding])

mb_encode_numericentity() converts specified character codes in string *str* from HTML numeric character reference to character code. It returns converted string.

array is array specifies code area to convert.

encoding is character encoding.

Example 1. *convmap* example

```
$convmap = array (
 int start_code1, int end_code1, int offset1, int mask1,
 int start_code2, int end_code2, int offset2, int mask2,
 ........
 int start_codeN, int end_codeN, int offsetN, int maskN );
// Specify Unicode value for start_codeN and end_codeN
// Add offsetN to value and take bit-wise 'AND' with maskN, then
// it converts value to numeric string reference.
```

Example 2. mb_encode_numericentity() example

```
/* Convert Left side of ISO-8859-1 to HTML numeric character reference */
$convmap = array(0x80, 0xff, 0, 0xff);
$str = mb_encode_numericentity($str, $convmap, "ISO-8859-1");

/* Convert user defined SJIS-win code in block 95-104 to numeric
   string reference */
$convmap = array(
       0xe000, 0xe03e, 0x1040, 0xffff,
       0xe03f, 0xe0bb, 0x1041, 0xffff,
       0xe0bc, 0xe0fa, 0x1084, 0xffff,
       0xe0fb, 0xe177, 0x1085, 0xffff,
       0xe178, 0xe1b6, 0x10c8, 0xffff,
       0xe1b7, 0xe233, 0x10c9, 0xffff,
       0xe234, 0xe272, 0x110c, 0xffff,
       0xe273, 0xe2ef, 0x110d, 0xffff,
       0xe2f0, 0xe32e, 0x1150, 0xffff,
       0xe32f, 0xe3ab, 0x1151, 0xffff );
$str = mb_encode_numericentity($str, $convmap, "sjis-win");
```

See also: **mb_decode_numericentity()**.

mb_ereg_match

(4.2.0 - 4.3.0 only)

mb_ereg_match -- Regular expression match for multibyte string

Description

bool **mb_ereg_match** (string pattern, string string [, string option])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

mb_ereg_match() returns **TRUE** if *string* matches regular expression *pattern*, **FALSE** if not.

The internal encoding or the character encoding specified in **mb_regex_encoding()** will be used as character encoding.

> **Note:** This function is supported in PHP 4.2.0 or higher.

See also: **mb_regex_encoding()**, **mb_ereg()**.

mb_ereg_replace

(4.2.0 - 4.3.0 only)

mb_ereg_replace -- Replace regular expression with multibyte support

Description

string **mb_ereg_replace** (string pattern, string replacement, string string [, array option])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

mb_ereg_replace() scans *string* for matches to *pattern*, then replaces the matched text with *replacement* and returns the result string or **FALSE** on error. Multibyte character can be used in *pattern*.

Matching condition can be set by *option* parameter. If `i` is specified for this parameter, the case will be ignored. If `x` is specified, white space will be ignored. If `m` is specified, match will be executed in multiline mode and line break will be included in '.'. If `p` is specified, match will be executed in POSIX mode, line break will be considered as normal character. If `e` is specified, *replacement* string will be evaluated as PHP expression.

The internal encoding or the character encoding specified in **mb_regex_encoding()** will be used as character encoding.

> **Note:** This function is supported in PHP 4.2.0 or higher.

See also: **mb_regex_encoding()**, **mb_eregi_replace()**.

mb_ereg_search_getpos

(4.2.0 - 4.3.0 only)

mb_ereg_search_getpos -- Returns start point for next regular expression match

Description

array **mb_ereg_search_getpos** (void)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

mb_ereg_search_getpos() returns the point to start regular expression match for **mb_ereg_search()**, **mb_ereg_search_pos()**, **mb_ereg_search_regs()**. The position is represented by bytes from the head of string.

The internal encoding or the character encoding specified in **mb_regex_encoding()** will be used as character encoding.

> **Note:** This function is supported in PHP 4.2.0 or higher.

See also: **mb_regex_encoding()**, **mb_ereg_search_setpos()**.

mb_ereg_search_getregs

(4.2.0 - 4.3.0 only)

mb_ereg_search_getregs -- Retrive the result from the last multibyte regular expression match

Description

array **mb_ereg_search_getregs** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

mb_ereg_search_getregs() returns an array including the sub-string of matched part by last **mb_ereg_search()**, **mb_ereg_search_pos()**, **mb_ereg_search_regs()**. If there are some maches, the first element will have the matched sub-string, the second element will have the first part grouped with brackets, the third element will have the second part grouped with brackets, and so on. It returns **FALSE** on error;

The internal encoding or the character encoding specified in **mb_regex_encoding()** will be used as character encoding.

> **Note:** This function is supported in PHP 4.2.0 or higher.

See also: **mb_regex_encoding()**, **mb_ereg_search_init()**.

mb_ereg_search_init

(4.2.0 - 4.3.0 only)

mb_ereg_search_init -- Setup string and regular expression for multibyte regular expression match

Description

array **mb_ereg_search_init** (string string [, string pattern [, string option]])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

mb_ereg_search_init() sets *string* and *pattern* for multibyte regular expression. These values are used for **mb_ereg_search()**, **mb_ereg_search_pos()**, **mb_ereg_search_regs()**. It returns **TRUE** for success, **FALSE** for error.

The internal encoding or the character encoding specified in **mb_regex_encoding()** will be used as character encoding.

> **Note:** This function is supported in PHP 4.2.0 or higher.

See also: **mb_regex_encoding()**, **mb_ereg_search_regs()**.

mb_ereg_search_pos

(4.2.0 - 4.3.0 only)

mb_ereg_search_pos -- Return position and length of matched part of multibyte regular expression for predefined multibyte string

Description

array **mb_ereg_search_pos** ([string pattern [, string option]])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

mb_ereg_search_pos() returns an array including position of matched part for multibyte regular expression. The first element of the array will be the beggining of matched part, the second element will be length (bytes) of matched part. It returns **FALSE** on error.

The string for match is specified by **mb_ereg_search_init()**. It it is not specified, the previous one will be used.

The internal encoding or the character encoding specified in **mb_regex_encoding()** will be used as character encoding.

> **Note:** This function is supported in PHP 4.2.0 or higher.

See also: **mb_regex_encoding()**, **mb_ereg_search_init()**.

mb_ereg_search_regs

(4.2.0 - 4.3.0 only)

mb_ereg_search_regs -- Returns the matched part of multibyte regular expression

Description

array **mb_ereg_search_regs** ([string pattern [, string option]])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

mb_ereg_search_regs() executes the multibyte regular expression match, and if there are some matched part, it returns an array including substring of matched part as first element, the first grouped part with brackets as second element, the second grouped part as third element, and so on. It returns **FALSE** on error.

The internal encoding or the character encoding specified in **mb_regex_encoding()** will be used as character encoding.

> **Note:** This function is supported in PHP 4.2.0 or higher.

See also: **mb_regex_encoding()**, **mb_ereg_search_init()**.

mb_ereg_search_setpos

(4.2.0 - 4.3.0 only)

mb_ereg_search_setpos -- Set start point of next regular expression match

Description

array **mb_ereg_search_setpos** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

mb_ereg_search_setpos() sets the starting point of match for **mb_ereg_search()**.

The internal encoding or the character encoding specified in **mb_regex_encoding()** will be used as character encoding.

> **Note:** This function is supported in PHP 4.2.0 or higher.

See also: **mb_regex_encoding()**, **mb_ereg_search_init()**.

mb_ereg_search

(4.2.0 - 4.3.0 only)

mb_ereg_search -- Multibyte regular expression match for predefined multibyte string

Description

bool **mb_ereg_search** ([string pattern [, string option]])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

mb_ereg_search() returns **TRUE** if the multibyte string matches with the regular expression, **FALSE** for otherwise. The string for matching is set by **mb_ereg_search_init()**. If *pattern* is not specified, the previous one is used.

The internal encoding or the character encoding specified in **mb_regex_encoding()** will be used as character encoding.

> **Note:** This function is supported in PHP 4.2.0 or higher.

See also: **mb_regex_encoding()**, **mb_ereg_search_init()**.

mb_ereg

(4.2.0 - 4.3.0 only)

mb_ereg -- Regular expression match with multibyte support

Description

int **mb_ereg** (string pattern, string string [, array regs])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

mb_ereg() executes the regular expression match with multibyte support, and returns 1 if matches are found. If the optional third parameter was specified, the function returns the byte length of matched part, and therarray *regs* will contain the substring of matched string. The functions returns 1 if it matches with the empty string. It no matche found or error happend, **FALSE** will be returned.

The internal encoding or the character encoding specified in **mb_regex_encoding()** will be used as character encoding.

> **Note:** This function is supported in PHP 4.2.0 or higher.

See also: **mb_regex_encoding()**, **mb_eregi()**

mb_eregi_replace

(4.2.0 - 4.3.0 only)

mb_eregi_replace -- Replace regular expression with multibyte support ignoring case

Description

string **mb_eregi_replace** (string pattern, string replace, string string)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

mb_ereg_replace() scans *string* for matches to *pattern*, then replaces the matched text with *replacement* and returns the result string or **FALSE** on error. Multibyte character can be used in *pattern*. The case will be ignored.

The internal encoding or the character encoding specified in **mb_regex_encoding()** will be used as character encoding.

Note: This function is supported in PHP 4.2.0 or higher.

See also: **mb_regex_encoding()**, **mb_ereg_replace()**.

mb_eregi

(4.2.0 - 4.3.0 only)

mb_eregi -- Regular expression match ignoring case with multibyte support

Description

int **mb_eregi** (string pattern, string string [, array regs])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

mb_eregi() executes the regular expression match with multibyte support, and returns 1 if matches are found. This function ignore case. If the optional third parameter was specified, the function returns the byte length of matched part, and therarray *regs* will contain the substring of matched string. The functions returns 1 if it matches with the empty string. It no matche found or error happend, **FALSE** will be returned.

The internal encoding or the character encoding specified in **mb_regex_encoding()** will be used as character encoding.

Note: This function is supported in PHP 4.2.0 or higher.

See also: **mb_regex_encoding()**, **mb_ereg()**.

mb_get_info

(PHP 4 >= 4.2.0)

mb_get_info -- Get internal settings of mbstring

Description

string **mb_get_info** ([string type])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

mb_get_info() returns internal setting parameter of mbstring.

If *type* isn't specified or is specified to "all", an array having the elements "internal_encoding", "http_output", "http_input", "func_overload" will be returned.

If *type* is specified for "http_output", "http_input", "internal_encoding", "func_overload", the specified setting parameter will be returned.

See also **mb_internal_encoding()**, **mb_http_output()**.

mb_http_input

(PHP 4 >= 4.0.6)

mb_http_input -- Detect HTTP input character encoding

Description

string **mb_http_input** ([string type])

mb_http_input() returns result of HTTP input character encoding detection.

type: Input string specifies input type. "G" for GET, "P" for POST, "C" for COOKIE. If type is omitted, it returns last input type processed.

Return Value: Character encoding name. If **mb_http_input()** does not process specified HTTP input, it returns **FALSE**.

See also **mb_internal_encoding()**, **mb_http_output()**, **mb_detect_order()**.

mb_http_output

(PHP 4 >= 4.0.6)

mb_http_output -- Set/Get HTTP output character encoding

Description

string **mb_http_output** ([string encoding])

If *encoding* is set, **mb_http_output()** sets HTTP output character encoding to *encoding*. Output after this function is converted to *encoding*. **mb_http_output()** returns **TRUE** for success and **FALSE** for failure.

If *encoding* is omitted, **mb_http_output()** returns current HTTP output character encoding.

See also **mb_internal_encoding()**, **mb_http_input()**, **mb_detect_order()**.

mb_internal_encoding

(PHP 4 >= 4.0.6)

mb_internal_encoding -- Set/Get internal character encoding

Description

string **mb_internal_encoding** ([string encoding])

mb_internal_encoding() sets internal character encoding to *encoding* If parameter is omitted, it returns current internal encoding.

encoding is used for HTTP input character encoding conversion, HTTP output character encoding conversion and default character encoding for string functions defined by mbstring module.

encoding: Character encoding name

Return Value: If *encoding* is set,**mb_internal_encoding()** returns **TRUE** for success, otherwise returns **FALSE**. If *encoding* is omitted, it returns current character encoding name.

Example 1. mb_internal_encoding() example

```
/* Set internal character encoding to UTF-8 */
mb_internal_encoding("UTF-8");
```

```
/* Display current internal character encoding */
echo mb_internal_encoding();
```

See also mb_http_input(), mb_http_output(), mb_detect_order().

mb_language

(PHP 4 >= 4.0.6)

mb_language -- Set/Get current language

Description

string **mb_language** ([string language])

mb_language() sets language. If *language* is omitted, it returns current language as string.

language setting is used for encoding e-mail messages. Valid languages are "Japanese", "ja","English","en" and "uni" (UTF-8). mb_send_mail() uses this setting to encode e-mail.

Language and its setting is ISO-2022-JP/Base64 for Japanese, UTF-8/Base64 for uni, ISO-8859-1/quoted printable for English.

Return Value: If *language* is set and *language* is valid, it returns TRUE. Otherwise, it returns FALSE. When *language* is omitted, it returns language name as string. If no language is set previously, it returns FALSE.

See also mb_send_mail().

mb_output_handler

(PHP 4 >= 4.0.6)

mb_output_handler -- Callback function converts character encoding in output buffer

Description

string **mb_output_handler** (string contents, int status)

mb_output_handler() is ob_start() callback function. **mb_output_handler()** converts characters in output buffer from internal character encoding to HTTP output character encoding.

4.1.0 or later version, this hanlder adds charset HTTP header when following conditions are met:

- Does not set `Content-Type` by header()
- Default MIME type begins with `text/`
- `http_output` setting is other than pass

contents : Output buffer contents

status : Output buffer status

Return Value: String converted

Example 1. mb_output_handler() example

```
mb_http_output("UTF-8");
ob_start("mb_output_handler");
```

> **Note:** If you want to output some binary data such as image from PHP script with PHP 4.3.0 or later, Content-Type: header must be send using header() before any binary data was send to client (e.g. header("Content-Type: image/png")). If Content-Type: header was send, output character encoding conversion will not be performed.
>
> Note that if 'Content-Type: text/*' was send using header(), the sending data is regarded as text, encoding conversion will be performed using character encoding settings.

If you want to output some binary data such as image from PHP script with PHP 4.2.x or earlier, you must set output encoding to "pass" using **mb_http_output()**.

See also **ob_start()**.

mb_parse_str

(PHP 4 >= 4.0.6)

mb_parse_str -- Parse GET/POST/COOKIE data and set global variable

Description

boolean **mb_parse_str** (string encoded_string [, array result])

mb_parse_str() parses GET/POST/COOKIE data and sets global variables. Since PHP does not provide raw POST/COOKIE data, it can only used for GET data for now. It preses URL encoded data, detects encoding, converts coding to internal encoding and set values to *result* array or global variables.

encoded_string: URL encoded data.

result: Array contains decoded and character encoding converted values.

Return Value: It returns **TRUE** for success or **FALSE** for failure.

See also **mb_detect_order()**, **mb_internal_encoding()**.

mb_preferred_mime_name

(PHP 4 >= 4.0.6)

mb_preferred_mime_name -- Get MIME charset string

Description

string **mb_preferred_mime_name** (string encoding)

mb_preferred_mime_name() returns MIME `charset` string for character encoding *encoding*. It returns `charset` string.

Example 1. mb_preferred_mime_string() example

```
$outputenc = "sjis-win";
mb_http_output($outputenc);
ob_start("mb_output_handler");
header("Content-Type: text/html; charset=" . mb_preferred_mime_name($outputenc));
```

mb_regex_encoding

(4.2.0 - 4.3.0 only)

mb_regex_encoding -- Returns current encoding for multibyte regex as string

Description

string **mb_regex_encoding** ([string encoding])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

mb_regex_encoding() returns the character encoding used by multibyte regex functions.

If the optional parameter *encoding* is specified, it is set to the character encoding for multibyte regex. The default value is the

internal character encoding.

Note: This function is supported in PHP 4.2.0 or higher.

See also: **mb_internal_encoding()**, **mb_ereg()**

mb_regex_set_options

(PHP 4 4.3.0 only)

mb_regex_set_options -- Set/Get the default options for mbregex functions

Description

string **mb_regex_set_options** ([string options])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

mb_regex_set_options() sets the default options described by *options* for multibyte regex functions.

Returns the previous options. If *options* is omitted, it returns the string that describes the current options.

Note: This function is supported in PHP 4.3.0 or higher.

See also: **mb_split()**, **mb_ereg()** **mb_eregi()**

mb_send_mail

(PHP 4 >= 4.0.6)

mb_send_mail -- Send encoded mail.

Description

boolean **mb_send_mail** (string to, string subject, string message [, string additional_headers [, string additional_parameter]])

mb_send_mail() sends email. Headers and message are converted and encoded according to **mb_language()** setting. **mb_send_mail()** is wrapper function of **mail()**. See **mail()** for details.

to is mail addresses send to. Multiple recipients can be specified by putting a comma between each address in to. This parameter is not automatically encoded.

subject is subject of mail.

message is mail message.

additional_headers is inserted at the end of the header. This is typically used to add extra headers. Multiple extra headers are separated with a newline ("\n").

additional_parameter is a MTA command line parameter. It is useful when setting the correct Return-Path header when using sendmail.

Returns **TRUE** on success or **FALSE** on failure.

See also **mail()**, **mb_encode_mimeheader()**, and **mb_language()**.

mb_split

(4.2.0 - 4.3.0 only)

mb_split -- Split multibyte string using regular expression

Description

array **mb_split** (string pattern, string string [, int limit])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

mb_split() split multibyte *string* using regular expression *pattern* and returns the result as an array.

If optional parameter *limit* is specified, it will be split in *limit* elements as maximum.

The internal encoding or the character encoding specified in **mb_regex_encoding()** will be used as character encoding.

Note: This function is supported in PHP 4.2.0 or higher.

See also: **mb_regex_encoding()**, **mb_ereg()**.

mb_strcut

(PHP 4 >= 4.0.6)

mb_strcut -- Get part of string

Description

string **mb_strcut** (string str, int start [, int length [, string encoding]])

mb_strcut() returns the portion of *str* specified by the *start* and *length* parameters.

mb_strcut() performs equivalent operation as **mb_substr()** with different method. If *start* position is multi-byte character's second byte or larger, it starts from first byte of multi-byte character.

It subtracts string from *str* that is shorter than *length* AND character that is not part of multi-byte string or not being middle of shift sequence.

encoding is character encoding. If it is not set, internal character encoding is used.

See also **mb_substr()**, **mb_internal_encoding()**.

mb_strimwidth

(PHP 4 >= 4.0.6)

mb_strimwidth -- Get truncated string with specified width

Description

string **mb_strimwidth** (string str, int start, int width, string trimmarker [, string encoding])

mb_strimwidth() truncates string *str* to specified *width*. It returns truncated string.

If *trimmarker* is set, *trimmarker* is appended to return value.

start is start position offset. Number of characters from the beginning of string. (First character is 0)

trimmarker is string that is added to the end of string when string is truncated.

encoding is character encoding. If it is omitted, internal encoding is used.

Example 1. mb_strimwidth() example

```
$str = mb_strimwidth($str, 0, 40, "..>");
```

See also: **mb_strwidth()**, **mb_internal_encoding()**.

mb_strlen

(PHP 4 >= 4.0.6)

mb_strlen -- Get string length

Description

string **mb_strlen** (string str [, string encoding])

mb_strlen() returns number of characters in string *str* having character encoding *encoding*. A multi-byte character is counted as 1.

encoding is character encoding for *str*. If *encoding* is omitted, internal character encoding is used.

See also **mb_internal_encoding()**, **strlen()**.

mb_strpos

(PHP 4 >= 4.0.6)

mb_strpos -- Find position of first occurrence of string in a string

Description

int **mb_strpos** (string haystack, string needle [, int offset [, string encoding]])

mb_strpos() returns the numeric position of the first occurrence of *needle* in the *haystack* string. If *needle* is not found, it returns **FALSE**.

mb_strpos() performs multi-byte safe **strpos()** operation based on number of characters. *needle* position is counted from the beginning of the *haystack*. First character's position is 0. Second character position is 1, and so on.

If *encoding* is omitted, internal character encoding is used. **mb_strrpos()** accepts `string` for *needle* where **strrpos()** accepts only character.

offset is search offset. If it is not specified, 0 is used.

encoding is character encoding name. If it is omitted, internal character encoding is used.

See also **mb_strpos()**, **mb_internal_encoding()**, **strpos()**

mb_strrpos

(PHP 4 >= 4.0.6)

mb_strrpos -- Find position of last occurrence of a string in a string

Description

int **mb_strrpos** (string haystack, string needle [, string encoding])

mb_strrpos() returns the numeric position of the last occurrence of *needle* in the *haystack* string. If *needle* is not found, it returns **FALSE**.

mb_strrpos() performs multi-byte safe **strrpos()** operation based on number of characters. *needle* position is counted from the beginning of *haystack*. First character's position is 0. Second character position is 1.

If *encoding* is omitted, internal encoding is assumed. **mb_strrpos()** accepts `string` for *needle* where **strrpos()** accepts only character.

encoding is character encoding. If it is not specified, internal character encoding is used.

See also **mb_strpos()**, **mb_internal_encoding()**, **strrpos()**.

mb_strtolower

(PHP 4 >= 4.3.0)

mb_strtolower -- Make a string lowercase

Description

string **mb_strtolower** (string str [, string encoding])

mb_strtolower() returns *str* with all alphabetic characters converted to lowercase.

encoding specifies the encoding of *str*; if omitted, the internal character encoding value will be used.

For more information about the Unicode properties, please see http://www.unicode.org/unicode/reports/tr21/.

By contrast to **strtolower()**, 'alphabetic' is determined by the Unicode character properties. Thus the behaviour of this function is not affected by locale settings and it can convert any characters that have 'alphabetic' property, such as A-umlaut (Ä).

Example 1. mb_strtolower() example

```
$str = "Mary Had A Little Lamb and She LOVED It So";
$str = mb_strtolower($str);
print $str; # Prints mary had a little lamb and she loved it so
```

See also **strtolower()**, **mb_strtoupper()**, **mb_convert_case()**.

mb_strtoupper

(PHP 4 >= 4.3.0)

mb_strtoupper -- Make a string uppercase

Description

string **mb_strtoupper** (string str [, string encoding])

mb_strtoupper() returns *str* with all alphabetic characters converted to uppercase.

encoding specifies the encoding of *str*; if omitted, the internal character encoding value will be used.

By contrast to **strtoupper()**, 'alphabetic' is determined by the Unicode character properties. Thus the behaviour of this function is not affected by locale settings and it can convert any characters that have 'alphabetic' property, such as a-umlaut (ä).

For more information about the Unicode properties, please see http://www.unicode.org/unicode/reports/tr21/.

Example 1. mb_strtoupper() example

```
$str = "Mary Had A Little Lamb and She LOVED It So";
$str = mb_strtoupper($str);
print $str; # Prints MARY HAD A LITTLE LAMB AND SHE LOVED IT SO
```

See also **strtoupper()**, **mb_strtolower()**, **mb_convert_case()**.

mb_strwidth

(PHP 4 >= 4.0.6)

mb_strwidth -- Return width of string

Description

int **mb_strwidth** (string str [, string encoding])

mb_strwidth() returns width of string *str*.

Multi-byte character usually twice of width compare to single byte character.

```
Character width

       U+0000 - U+0019   0
       U+0020 - U+1FFF   1
       U+2000 - U+FF60   2
       U+FF61 - U+FF9F   1
       U+FFA0 -          2
```

encoding is character encoding. If it is omitted, internal encoding is used.

See also: **mb_strimwidth()**, **mb_internal_encoding()**.

mb_substitute_character

(PHP 4 >= 4.0.6)

mb_substitute_character -- Set/Get substitution character

Description

mixed **mb_substitute_character** ([mixed substrchar])

mb_substitute_character() specifies substitution character when input character encoding is invalid or character code is not exist in output character encoding. Invalid characters may be substituted **NULL**(no output), string or integer value (Unicode character code value).

This setting affects **mb_detect_encoding()** and **mb_send_mail()**.

substchar : Specify Unicode value as integer or specify as string as follows

- "none" : no output
- "long" : Output character code value (Example: U+3000,JIS+7E7E)

Return Value: If *substchar* is set, it returns **TRUE** for success, otherwise returns **FALSE**. If *substchar* is not set, it returns Unicode value or "none"/"long".

Example 1. mb_substitute_character() example

```
/* Set with Unicode U+3013 (GETA MARK) */
mb_substitute_character(0x3013);

/* Set hex format */
mb_substitute_character("long");

/* Display current setting */
echo mb_substitute_character();
```

mb_substr_count

(PHP 4 >= 4.3.0)

mb_substr_count -- Count the number of substring occurrences

Description

int **mb_substr_count** (string haystack, string needle [, string encoding])

mb_substr_count() returns the number of times the *needle* substring occurs in the *haystack* string.

encoding specifies the encoding for *needle* and *haystack*. If omitted, internal character encoding is used.

Example 1. mb_substr_count() example

```
<?php
print mb_substr_count("This is a test", "is"); // prints out 2
?>
```

See also **substr_count()**, **mb_strpos()**, **mb_substr()**.

mb_substr

(PHP 4 >= 4.0.6)

mb_substr -- Get part of string

Description

string **mb_substr** (string str, int start [, int length [, string encoding]])

mb_substr() returns the portion of *str* specified by the *start* and *length* parameters.

mb_substr() performs multi-byte safe **substr()** operation based on number of characters. Position is counted from the beginning of *str*. First character's position is 0. Second character position is 1, and so on.

If *encoding* is omitted, internal encoding is assumed.

encoding is character encoding. If it is omitted, internal character encoding is used.

See also **mb_strcut()**, **mb_internal_encoding()**.

LIII. MCAL functions

Introduction

MCAL stands for Modular Calendar Access Library.

Libmcal is a C library for accessing calendars. It's written to be very modular, with pluggable drivers. MCAL is the calendar equivalent of the IMAP module for mailboxes.

With mcal support, a calendar stream can be opened much like the mailbox stream with the IMAP support. Calendars can be local file stores, remote ICAP servers, or other formats that are supported by the mcal library.

Calendar events can be pulled up, queried, and stored. There is also support for calendar triggers (alarms) and recurring events.

With libmcal, central calendar servers can be accessed, removing the need for any specific database or local file programming.

Most of the functions use an internal event structure that is unique for each stream. This alleviates the need to pass around large objects between functions. There are convenience functions for setting, initializing, and retrieving the event structure values.

> **Note:** PHP had an ICAP extension previously, but the original library and the PHP extension is not supported anymore. The suggested replacement is MCAL.
>
> **Note:** This extension is not available on Windows platforms.

Requirements

This extension requires the mcal library to be installed. Grab the latest version from http://mcal.chek.com/ and compile and install it.

Installation

After you installed the mcal library, to get these functions to work, you have to compile PHP `-with-mcal[=DIR]`.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

`MCAL_SUNDAY` (integer)

`MCAL_MONDAY` (integer)

`MCAL_TUESDAY` (integer)

`MCAL_WEDNESDAY` (integer)

`MCAL_THURSDAY` (integer)

`MCAL_FRIDAY` (integer)

`MCAL_SATURDAY` (integer)

`MCAL_JANUARY` (integer)

`MCAL_FEBRUARY` (integer)

`MCAL_MARCH` (integer)

`MCAL_APRIL` (integer)

`MCAL_MAY` (integer)

`MCAL_JUNE` (integer)

`MCAL_JULY` (integer)

`MCAL_AUGUST` (integer)

`MCAL_SEPTEMBER` (integer)

`MCAL_OCTOBER` (integer)

`MCAL_NOVEMBER` (integer)

`MCAL_DECEMBER` (integer)

`MCAL_RECUR_NONE` (integer)

`MCAL_RECUR_DAILY` (integer)

`MCAL_RECUR_WEEKLY` (integer)

`MCAL_RECUR_MONTHLY_MDAY` (integer)

`MCAL_RECUR_MONTHLY_WDAY` (integer)

`MCAL_RECUR_YEARLY` (integer)

MCAL_M_SUNDAY (integer)

MCAL_M_MONDAY (integer)

MCAL_M_TUESDAY (integer)

MCAL_M_WEDNESDAY (integer)

MCAL_M_THURSDAY (integer)

MCAL_M_FRIDAY (integer)

MCAL_M_SATURDAY (integer)

MCAL_M_WEEKDAYS (integer)

MCAL_M_WEEKEND (integer)

MCAL_M_ALLDAYS (integer)

Table of Contents

mcal_append_event

(PHP 4)

mcal_append_event -- Store a new event into an MCAL calendar

Description

int **mcal_append_event** (int mcal_stream)

mcal_append_event() Stores the global event into an MCAL calendar for the given stream.

Returns the id of the newly inserted event.

mcal_close

(PHP 3>= 3.0.13, PHP 4)

mcal_close -- Close an MCAL stream

Description

int **mcal_close** (int mcal_stream, int flags)

Closes the given mcal stream.

mcal_create_calendar

(PHP 3>= 3.0.13, PHP 4)

mcal_create_calendar -- Create a new MCAL calendar

Description

string **mcal_create_calendar** (int stream, string calendar)

Creates a new calendar named *calendar*.

mcal_date_compare

(PHP 3>= 3.0.13, PHP 4)

mcal_date_compare -- Compares two dates

Description

int **mcal_date_compare** (int a_year, int a_month, int a_day, int b_year, int b_month, int b_day)

mcal_date_compare() Compares the two given dates, returns <o, o, >o if a<b, a==b, a>b respectively.

mcal_date_valid

(PHP 3>= 3.0.13, PHP 4)

mcal_date_valid -- Returns **TRUE** if the given year, month, day is a valid date

Description

int **mcal_date_valid** (int year, int month, int day)

mcal_date_valid() Returns **TRUE** if the given year, month and day is a valid date, **FALSE** if not.

mcal_day_of_week

(PHP 3>= 3.0.13, PHP 4)

mcal_day_of_week -- Returns the day of the week of the given date

Description

int **mcal_day_of_week** (int year, int month, int day)

mcal_day_of_week() returns the day of the week of the given date. Possible return values range from o for Sunday through 6 for Saturday.

mcal_day_of_year

(PHP 3>= 3.0.13, PHP 4)

mcal_day_of_year -- Returns the day of the year of the given date

Description

int **mcal_** (int year, int month, int day)

mcal_day_of_year() returns the day of the year of the given date.

mcal_days_in_month

(PHP 3>= 3.0.13, PHP 4)

mcal_days_in_month -- Returns the number of days in the given month

Description

int **mcal_days_in_month** (int month, int leap year)

mcal_days_in_month() Returns the number of days in the given month, taking into account if the given year is a leap year or not.

mcal_delete_calendar

(PHP 3>= 3.0.13, PHP 4)

mcal_delete_calendar -- Delete an MCAL calendar

Description

string **mcal_delete_calendar** (int stream, string calendar)

Deletes the calendar named *calendar*.

mcal_delete_event

(PHP 3>= 3.0.13, PHP 4)

mcal_delete_event -- Delete an event from an MCAL calendar

Description

int **mcal_delete_event** (int mcal_stream [, int event_id])

mcal_delete_event() deletes the calendar event specified by the event_id.

Returns **TRUE**.

mcal_event_add_attribute

(PHP 3>= 3.0.15, PHP 4)

mcal_event_add_attribute -- Adds an attribute and a value to the streams global event structure

Description

void **mcal_event_add_attribute** (int stream, string attribute, string value)

mcal_event_add_attribute() adds an attribute to the stream's global event structure with the value given by "value".

mcal_event_init

(PHP 3>= 3.0.13, PHP 4)

mcal_event_init -- Initializes a streams global event structure

Description

int **mcal_event_init** (int stream)

mcal_event_init() initializes a streams global event structure. this effectively sets all elements of the structure to o, or the default settings.

Returns **TRUE**.

mcal_event_set_alarm

(PHP 3>= 3.0.13, PHP 4)

mcal_event_set_alarm -- Sets the alarm of the streams global event structure

Description

int **mcal_event_set_alarm** (int stream, int alarm)

mcal_event_set_alarm() sets the streams global event structure's alarm to the given minutes before the event.

Returns **TRUE**.

mcal_event_set_category

(PHP 3>= 3.0.13, PHP 4)

mcal_event_set_category -- Sets the category of the streams global event structure

Description

int **mcal_event_set_category** (int stream, string category)

mcal_event_set_category() sets the streams global event structure's category to the given string.

Returns **TRUE**.

mcal_event_set_class

(PHP 3>= 3.0.13, PHP 4)

mcal_event_set_class -- Sets the class of the streams global event structure

Description

int **mcal_event_set_class** (int stream, int class)

mcal_event_set_class() sets the streams global event structure's class to the given value. The class is either 1 for public, or 0 for private.

Returns **TRUE**.

mcal_event_set_description

(PHP 3>= 3.0.13, PHP 4)

mcal_event_set_description -- Sets the description of the streams global event structure

Description

int **mcal_event_set_description** (int stream, string description)

mcal_event_set_description() sets the streams global event structure's description to the given string.

Returns **TRUE**.

mcal_event_set_end

(PHP 3>= 3.0.13, PHP 4)

mcal_event_set_end -- Sets the end date and time of the streams global event structure

Description

int **mcal_event_set_end** (int stream, int year, int month [, int day [, int hour [, int min [, int sec]]]])

mcal_event_set_end() sets the streams global event structure's end date and time to the given values.

Returns **TRUE**.

mcal_event_set_recur_daily

(PHP 3>= 3.0.13, PHP 4)

mcal_event_set_recur_daily -- Sets the recurrence of the streams global event structure

Description

int **mcal_event_set_recur_daily** (int stream, int year, int month, int day, int interval)

mcal_event_set_recur_daily() sets the streams global event structure's recurrence to the given value to be reoccuring on a daily basis, ending at the given date.

mcal_event_set_recur_monthly_mday

(PHP 3>= 3.0.13, PHP 4)

mcal_event_set_recur_monthly_mday -- Sets the recurrence of the streams global event structure

Description

int **mcal_event_set_recur_monthly_mday** (int stream, int year, int month, int day, int interval)

mcal_event_set_recur_monthly_mday() sets the streams global event structure's recurrence to the given value to be reoccuring on a monthly by month day basis, ending at the given date.

mcal_event_set_recur_monthly_wday

(PHP 3>= 3.0.13, PHP 4)

mcal_event_set_recur_monthly_wday -- Sets the recurrence of the streams global event structure

Description

int **mcal_event_set_recur_monthly_wday** (int stream, int year, int month, int day, int interval)

mcal_event_set_recur_monthly_wday() sets the streams global event structure's recurrence to the given value to be reoccuring on a monthly by week basis, ending at the given date.

mcal_event_set_recur_none

(PHP 3>= 3.0.15, PHP 4)

mcal_event_set_recur_none -- Sets the recurrence of the streams global event structure

Description

int **mcal_event_set_recur_none** (int stream)

mcal_event_set_recur_none() sets the streams global event structure to not recur (event->recur_type is set to **MCAL_RECUR_NONE**).

mcal_event_set_recur_weekly

(PHP 3>= 3.0.13, PHP 4)

mcal_event_set_recur_weekly -- Sets the recurrence of the streams global event structure

Description

int **mcal_event_set_recur_weekly** (int stream, int year, int month, int day, int interval, int weekdays)

mcal_event_set_recur_weekly() sets the streams global event structure's recurrence to the given value to be reoccuring on a weekly basis, ending at the given date.

mcal_event_set_recur_yearly

(PHP 3>= 3.0.13, PHP 4)

mcal_event_set_recur_yearly -- Sets the recurrence of the streams global event structure

Description

int **mcal_event_set_recur_yearly** (int stream, int year, int month, int day, int interval)

mcal_event_set_recur_yearly() sets the streams global event structure's recurrence to the given value to be reoccuring on a yearly basis,ending at the given date.

mcal_event_set_start

(PHP 3>= 3.0.13, PHP 4)

mcal_event_set_start -- Sets the start date and time of the streams global event structure

Description

int **mcal_event_set_start** (int stream, int year, int month [, int day [, int hour [, int min [, int sec]]]])

mcal_event_set_start() sets the streams global event structure's start date and time to the given values.

Returns **TRUE**.

mcal_event_set_title

(PHP 3>= 3.0.13, PHP 4)

mcal_event_set_title -- Sets the title of the streams global event structure

Description

int **mcal_event_set_title** (int stream, string title)

mcal_event_set_title() sets the streams global event structure's title to the given string.

Returns **TRUE**.

mcal_expunge

(no version information, might be only in CVS)

mcal_expunge -- Deletes all events marked for being expunged.

Description

int **mcal_expunge** (int stream)

mcal_expunge() Deletes all events which have been previously marked for deletion.

mcal_fetch_current_stream_event

(PHP 3>= 3.0.13, PHP 4)

mcal_fetch_current_stream_event -- Returns an object containing the current streams event structure

Description

object **mcal_fetch_current_stream_event** (int stream)

mcal_fetch_current_stream_event() returns the current stream's event structure as an object containing:

- int id - ID of that event.
- int public - **TRUE** if the event if public, **FALSE** if it is private.
- string category - Category string of the event.
- string title - Title string of the event.
- string description - Description string of the event.
- int alarm - number of minutes before the event to send an alarm/reminder.
- object start - Object containing a datetime entry.

- object end - Object containing a datetime entry.
- int recur_type - recurrence type
- int recur_interval - recurrence interval
- datetime recur_enddate - recurrence end date
- int recur_data - recurrence data

All datetime entries consist of an object that contains:

- int year - year
- int month - month
- int mday - day of month
- int hour - hour
- int min - minutes
- int sec - seconds
- int alarm - minutes before event to send an alarm

mcal_fetch_event

(PHP 3>= 3.0.13, PHP 4)

mcal_fetch_event -- Fetches an event from the calendar stream

Description

object **mcal_fetch_event** (int mcal_stream, int event_id [, int options])

mcal_fetch_event() fetches an event from the calendar stream specified by *id*.

Returns an event object consisting of:

- int id - ID of that event.
- int public - **TRUE** if the event if public, **FALSE** if it is private.
- string category - Category string of the event.
- string title - Title string of the event.
- string description - Description string of the event.
- int alarm - number of minutes before the event to send an alarm/reminder.
- object start - Object containing a datetime entry.
- object end - Object containing a datetime entry.
- int recur_type - recurrence type
- int recur_interval - recurrence interval
- datetime recur_enddate - recurrence end date
- int recur_data - recurrence data

All datetime entries consist of an object that contains:

- int year - year
- int month - month
- int mday - day of month

- int hour - hour
- int min - minutes
- int sec - seconds
- int alarm - minutes before event to send an alarm

The possible values for recur_type are:

- 0 - Indicates that this event does not recur
- 1 - This event recurs daily
- 2 - This event recurs on a weekly basis
- 3 - This event recurs monthly on a specific day of the month (e.g. the 10th of the month)
- 4 - This event recurs monthly on a sequenced day of the week (e.g. the 3rd Saturday)
- 5 - This event recurs on an annual basis

mcal_is_leap_year

(PHP 3>= 3.0.13, PHP 4)

mcal_is_leap_year -- Returns if the given year is a leap year or not

Description

int **mcal_is_leap_year** (int year)

mcal_is_leap_year() returns 1 if the given year is a leap year, 0 if not.

mcal_list_alarms

(PHP 3>= 3.0.13, PHP 4)

mcal_list_alarms -- Return a list of events that has an alarm triggered at the given datetime

Description

array **mcal_list_alarms** (int mcal_stream [, int begin_year [, int begin_month [, int begin_day [, int end_year [, int end_month [, int end_day]]]]]])

Returns an array of event ID's that has an alarm going off between the start and end dates, or if just a stream is given, uses the start and end dates in the global event structure.

mcal_list_events() function takes in an optional beginning date and an end date for a calendar stream. An array of event id's that are between the given dates or the internal event dates are returned.

mcal_list_events

(PHP 3>= 3.0.13, PHP 4)

mcal_list_events -- Return a list of IDs for a date or a range of dates.

Description

array **mcal_list_events** (int mcal_stream, object begin_date [, object end_date])

Returns an array of ID's that are between the start and end dates, or if just a stream is given, uses the start and end dates in the global event structure.

mcal_list_events() function takes in an beginning date and an optional end date for a calendar stream. An array of event id's that are between the given dates or the internal event dates are returned.

mcal_next_recurrence

(PHP 3>= 3.0.13, PHP 4)

mcal_next_recurrence -- Returns the next recurrence of the event

Description

int **mcal_next_recurrence** (int stream, int weekstart, array next)

mcal_next_recurrence() returns an object filled with the next date the event occurs, on or after the supplied date. Returns empty date field if event does not occur or something is invalid. Uses weekstart to determine what day is considered the beginning of the week.

mcal_open

(PHP 3>= 3.0.13, PHP 4)

mcal_open -- Opens up an MCAL connection

Description

int **mcal_open** (string calendar, string username, string password [, int options])

Returns an MCAL stream on success, **FALSE** on error.

mcal_open() opens up an MCAL connection to the specified *calendar* store. If the optional *options* is specified, passes the *options* to that mailbox also. The streams internal event structure is also initialized upon connection.

mcal_popen

(PHP 3>= 3.0.13, PHP 4)

mcal_popen -- Opens up a persistent MCAL connection

Description

int **mcal_popen** (string calendar, string username, string password [, int options])

Returns an MCAL stream on success, **FALSE** on error.

mcal_popen() opens up an MCAL connection to the specified *calendar* store. If the optional *options* is specified, passes the *options* to that mailbox also. The streams internal event structure is also initialized upon connection.

mcal_rename_calendar

(PHP 3>= 3.0.13, PHP 4)

mcal_rename_calendar -- Rename an MCAL calendar

Description

string **mcal_rename_calendar** (int stream, string old_name, string new_name)

Renames the calendar *old_name* to *new_name*.

mcal_reopen

(PHP 3>= 3.0.13, PHP 4)

mcal_reopen -- Reopens an MCAL connection

Description

int **mcal_reopen** (string calendar [, int options])

Reopens an MCAL stream to a new calendar.

mcal_reopen() reopens an MCAL connection to the specified *calendar* store. If the optional *options* is specified, passes the *options* to that mailbox also.

mcal_snooze

(PHP 3>= 3.0.13, PHP 4)

mcal_snooze -- Turn off an alarm for an event

Description

int **mcal_snooze** (int id)

mcal_snooze() turns off an alarm for a calendar event specified by the id.

Returns **TRUE**.

mcal_store_event

(PHP 3>= 3.0.13, PHP 4)

mcal_store_event -- Modify an existing event in an MCAL calendar

Description

int **mcal_store_event** (int mcal_stream)

mcal_store_event() Stores the modifications to the current global event for the given stream.

Returns the event id of the modified event on success and **FALSE** on error.

mcal_time_valid

(PHP 3>= 3.0.13, PHP 4)

mcal_time_valid -- Returns **TRUE** if the given year, month, day is a valid time

Description

int **mcal_time_valid** (int hour, int minutes, int seconds)

mcal_time_valid() Returns **TRUE** if the given hour, minutes and seconds is a valid time, **FALSE** if not.

mcal_week_of_year

(PHP 4)

mcal_week_of_year -- Returns the week number of the given date

Description

int **mcal_week_of_year** (int day, int month, int year)

LIV. Mcrypt Encryption Functions

Introduction

This is an interface to the mcrypt library, which supports a wide variety of block algorithms such as DES, TripleDES, Blowfish (default), 3-WAY, SAFER-SK64, SAFER-SK128, TWOFISH, TEA, RC2 and GOST in CBC, OFB, CFB and ECB cipher modes. Additionally, it supports RC6 and IDEA which are considered "non-free".

Requirements

These functions work using mcrypt. To use it, download libmcrypt-x.x.tar.gz from here and follow the included installation instructions. Windows users will find all the needed compiled mcrypt binaries here.

If you linked against libmcrypt 2.4.x or higher, the following additional block algorithms are supported: CAST, LOKI97, RIJNDAEL, SAFERPLUS, SERPENT and the following stream ciphers: ENIGMA (crypt), PANAMA, RC4 and WAKE. With libmcrypt 2.4.x or higher another cipher mode is also available; nOFB.

Installation

You need to compile PHP with the `--with-mcrypt[=DIR]` parameter to enable this extension. DIR is the mcrypt install directory. Make sure you compile libmcrypt with the option `--disable-posix-threads`.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. Mcrypt configuration options

Name	Default	Changeable
mcrypt.algorithms_dir	NULL	PHP_INI_ALL
mcrypt.modes_dir	NULL	PHP_INI_ALL

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

Mcrypt can operate in four block cipher modes (CBC, OFB, CFB, and ECB). If linked against libmcrypt-2.4.x or higher the

functions can also operate in the block cipher mode nOFB and in STREAM mode. Below you find a list with all supported encryption modes together with the constants that are defines for the encryption mode. For a more complete reference and discussion see Applied Cryptography by Schneier (ISBN 0-471-11709-9).

- MCRYPT_MODE_ECB (electronic codebook) is suitable for random data, such as encrypting other keys. Since data there is short and random, the disadvantages of ECB have a favorable negative effect.
- MCRYPT_MODE_CBC (cipher block chaining) is especially suitable for encrypting files where the security is increased over ECB significantly.
- MCRYPT_MODE_CFB (cipher feedback) is the best mode for encrypting byte streams where single bytes must be encrypted.
- MCRYPT_MODE_OFB (output feedback, in 8bit) is comparable to CFB, but can be used in applications where error propagation cannot be tolerated. It's insecure (because it operates in 8bit mode) so it is not recommended to use it.
- MCRYPT_MODE_NOFB (output feedback, in nbit) is comparable to OFB, but more secure because it operates on the block size of the algorithm.
- MCRYPT_MODE_STREAM is an extra mode to include some stream algorithms like WAKE or RC4.

Some other mode and random device constants:

MCRYPT_ENCRYPT (integer)

MCRYPT_DECRYPT (integer)

MCRYPT_DEV_RANDOM (integer)

MCRYPT_DEV_URANDOM (integer)

MCRYPT_RAND (integer)

Mcrypt ciphers

Here is a list of ciphers which are currently supported by the mcrypt extension. For a complete list of supported ciphers, see the defines at the end of `mcrypt.h`. The general rule with the mcrypt-2.2.x API is that you can access the cipher from PHP with MCRYPT_ciphername. With the libmcrypt-2.4.x and libmcrypt-2.5.x API these constants also work, but it is possible to specify the name of the cipher as a string with a call to **mcrypt_module_open()**.

- MCRYPT_3DES
- MCRYPT_ARCFOUR_IV (libmcrypt > 2.4.x only)
- MCRYPT_ARCFOUR (libmcrypt > 2.4.x only)
- MCRYPT_BLOWFISH
- MCRYPT_CAST_128
- MCRYPT_CAST_256
- MCRYPT_CRYPT
- MCRYPT_DES
- MCRYPT_DES_COMPAT (libmcrypt 2.2.x only)
- MCRYPT_ENIGMA (libmcrypt > 2.4.x only, alias for MCRYPT_CRYPT)
- MCRYPT_GOST
- MCRYPT_IDEA (non-free)
- MCRYPT_LOKI97 (libmcrypt > 2.4.x only)
- MCRYPT_MARS (libmcrypt > 2.4.x only, non-free)
- MCRYPT_PANAMA (libmcrypt > 2.4.x only)
- MCRYPT_RIJNDAEL_128 (libmcrypt > 2.4.x only)

- MCRYPT_RIJNDAEL_192 (libmcrypt > 2.4.x only)
- MCRYPT_RIJNDAEL_256 (libmcrypt > 2.4.x only)
- MCRYPT_RC2
- MCRYPT_RC4 (libmcrypt 2.2.x only)
- MCRYPT_RC6 (libmcrypt > 2.4.x only)
- MCRYPT_RC6_128 (libmcrypt 2.2.x only)
- MCRYPT_RC6_192 (libmcrypt 2.2.x only)
- MCRYPT_RC6_256 (libmcrypt 2.2.x only)
- MCRYPT_SAFER64
- MCRYPT_SAFER128
- MCRYPT_SAFERPLUS (libmcrypt > 2.4.x only)
- MCRYPT_SERPENT(libmcrypt > 2.4.x only)
- MCRYPT_SERPENT_128 (libmcrypt 2.2.x only)
- MCRYPT_SERPENT_192 (libmcrypt 2.2.x only)
- MCRYPT_SERPENT_256 (libmcrypt 2.2.x only)
- MCRYPT_SKIPJACK (libmcrypt > 2.4.x only)
- MCRYPT_TEAN (libmcrypt 2.2.x only)
- MCRYPT_THREEWAY
- MCRYPT_TRIPLEDES (libmcrypt > 2.4.x only)
- MCRYPT_TWOFISH (for older mcrypt 2.x versions, or mcrypt > 2.4.x)
- MCRYPT_TWOFISH128 (TWOFISHxxx are available in newer 2.x versions, but not in the 2.4.x versions)
- MCRYPT_TWOFISH192
- MCRYPT_TWOFISH256
- MCRYPT_WAKE (libmcrypt > 2.4.x only)
- MCRYPT_XTEA (libmcrypt > 2.4.x only)

You must (in CFB and OFB mode) or can (in CBC mode) supply an initialization vector (IV) to the respective cipher function. The IV must be unique and must be the same when decrypting/encrypting. With data which is stored encrypted, you can take the output of a function of the index under which the data is stored (e.g. the MD5 key of the filename). Alternatively, you can transmit the IV together with the encrypted data (see chapter 9.3 of Applied Cryptography by Schneier (ISBN 0-471-11709-9) for a discussion of this topic).

Examples

Mcrypt can be used to encrypt and decrypt using the above mentioned ciphers. If you linked against libmcrypt-2.2.x, the four important mcrypt commands (**mcrypt_cfb()**, **mcrypt_cbc()**, **mcrypt_ecb()**, and **mcrypt_ofb()**) can operate in both modes which are named MCRYPT_ENCRYPT and MCRYPT_DECRYPT, respectively.

Example 1. Encrypt an input value with TripleDES under 2.2.x in ECB mode

```
<?php
$key = "this is a secret key";
$input = "Let us meet at 9 o'clock at the secret place.";

$encrypted_data = mcrypt_ecb (MCRYPT_3DES, $key, $input, MCRYPT_ENCRYPT);
?>
```

This example will give you the encrypted data as a string in `$encrypted_data`.

If you linked against libmcrypt 2.4.x or 2.5.x, these functions are still available, but it is recommended that you use the advanced functions.

Example 2. Encrypt an input value with TripleDES under 2.4.x and higher in ECB mode

```
<?php
    $key = "this is a secret key";
    $input = "Let us meet at 9 o'clock at the secret place.";

    $td = mcrypt_module_open ('tripledes', '', 'ecb', '');
    $iv = mcrypt_create_iv (mcrypt_enc_get_iv_size ($td), MCRYPT_RAND);
    mcrypt_generic_init ($td, $key, $iv);
    $encrypted_data = mcrypt_generic ($td, $input);
    mcrypt_generic_deinit ($td);
    mcrypt_module_close ($td);
?>
```

This example will give you the encrypted data as a string in `$encrypted_data`. For a full example see **mcrypt_module_open()**.

Table of Contents

mcrypt_cbc

(PHP 3>= 3.0.8, PHP 4)

mcrypt_cbc -- Encrypt/decrypt data in CBC mode

Description

string **mcrypt_cbc** (int cipher, string key, string data, int mode [, string iv])

string **mcrypt_cbc** (string cipher, string key, string data, int mode [, string iv])

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x or higher.

This function should not be used anymore, see **mcrypt_generic()** and **mdecrypt_generic()** for replacements.

mcrypt_cfb

(PHP 3>= 3.0.8, PHP 4)

mcrypt_cfb -- Encrypt/decrypt data in CFB mode

Description

string **mcrypt_cfb** (int cipher, string key, string data, int mode, string iv)

string **mcrypt_cfb** (string cipher, string key, string data, int mode [, string iv])

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x or higher.

This function should not be used anymore, see **mcrypt_generic()** and **mdecrypt_generic()** for replacements.

mcrypt_create_iv

(PHP 3>= 3.0.8, PHP 4)

mcrypt_create_iv -- Create an initialization vector (IV) from a random source

Description

string **mcrypt_create_iv** (int size, int source)

mcrypt_create_iv() is used to create an IV.

mcrypt_create_iv() takes two arguments, *size* determines the size of the IV, *source* specifies the source of the IV.

The source can be MCRYPT_RAND (system random number generator), MCRYPT_DEV_RANDOM (read data from /dev/random) and MCRYPT_DEV_URANDOM (read data from /dev/urandom). If you use MCRYPT_RAND, make sure to call srand() before to initialize the random number generator.

Example 1. mcrypt_create_iv() example

```
<?php
    $size = mcrypt_get_iv_size (MCRYPT_CAST_256, MCRYPT_MODE_CFB);
    $iv = mcrypt_create_iv ($size, MCRYPT_DEV_RANDOM);
?>
```

The IV is only meant to give an alternative seed to the encryption routines. This IV does not need to be secret at all, though it can be desirable. You even can send it along with your ciphertext without loosing security.

More information can be found at http://www.ciphersbyritter.com/GLOSSARY.HTM#IV, http://fn2.freenet.edmonton.ab.ca/~jsavard/crypto/co0409.htm and in chapter 9.3 of Applied Cryptography by Schneier (ISBN 0-471-11709-9) for a discussion of this topic.

mcrypt_decrypt

(PHP 4 >= 4.0.2)

mcrypt_decrypt -- Decrypts crypttext with given parameters

Description

string **mcrypt_decrypt** (string cipher, string key, string data, string mode [, string iv])

mcrypt_decrypt() decrypts the data and returns the unencrypted data.

Cipher is one of the MCRYPT_ciphername constants of the name of the algorithm as string.

Key is the key with which the data is encrypted. If it's smaller that the required keysize, it is padded with '\0'.

Data is the data that will be decrypted with the given cipher and mode. If the size of the data is not n * blocksize, the data will be padded with '\0'.

Mode is one of the MCRYPT_MODE_modename constants of one of "ecb", "cbc", "cfb", "ofb", "nofb" or "stream".

The *IV* parameter is used for the initialisation in CBC, CFB, OFB modes, and in some algorithms in STREAM mode. If you do not supply an IV, while it is needed for an algorithm, the function issues a warning and uses an IV with all bytes set to '\0'.

mcrypt_ecb

(PHP 3>= 3.0.8, PHP 4)

mcrypt_ecb -- Encrypt/decrypt data in ECB mode

Description

string **mcrypt_ecb** (int cipher, string key, string data, int mode)

string **mcrypt_ecb** (string cipher, string key, string data, int mode [, string iv])

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x or higher.

This function should not be used anymore, see **mcrypt_generic()** and **mdecrypt_generic()** for replacements.

mcrypt_enc_get_algorithms_name

(PHP 4 >= 4.0.2)

mcrypt_enc_get_algorithms_name -- Returns the name of the opened algorithm

Description

string **mcrypt_enc_get_algorithms_name** (resource td)

This function returns the name of the algorithm.

Example 1. mcrypt_enc_get_algorithms_name() example

```
<?php
    $td = mcrypt_module_open (MCRYPT_CAST_256, '', MCRYPT_MODE_CFB, '');
    echo mcrypt_enc_get_algorithms_name($td). "\n";

    $td = mcrypt_module_open ('cast-256', '', MCRYPT_MODE_CFB, '');
    echo mcrypt_enc_get_algorithms_name($td). "\n";
?>

Prints:
CAST-256
CAST-256
```

mcrypt_enc_get_block_size

(PHP 4 >= 4.0.2)

mcrypt_enc_get_block_size -- Returns the blocksize of the opened algorithm

Description

int **mcrypt_enc_get_block_size** (resource td)

This function returns the block size of the algorithm specified by the encryption descriptor *td* in bytes.

mcrypt_enc_get_iv_size

(PHP 4 >= 4.0.2)

mcrypt_enc_get_iv_size -- Returns the size of the IV of the opened algorithm

Description

int **mcrypt_enc_get_iv_size** (resource td)

This function returns the size of the iv of the algorithm specified by the encryption descriptor in bytes. If it returns 'o' then the IV is ignored in the algorithm. An IV is used in cbc, cfb and ofb modes, and in some algorithms in stream mode.

mcrypt_enc_get_key_size

(PHP 4 >= 4.0.2)

mcrypt_enc_get_key_size -- Returns the maximum supported keysize of the opened mode

Description

int **mcrypt_enc_get_key_size** (resource td)

This function returns the maximum supported key size of the algorithm specified by the encryption descriptor *td* in bytes.

mcrypt_enc_get_modes_name

(PHP 4 >= 4.0.2)

mcrypt_enc_get_modes_name -- Returns the name of the opened mode

Description

string **mcrypt_enc_get_modes_name** (resource td)

This function returns the name of the mode.

Example 1. mcrypt_enc_get_modes_name() example

```
<?php
    $td = mcrypt_module_open (MCRYPT_CAST_256, '', MCRYPT_MODE_CFB, '');
    echo mcrypt_enc_get_modes_name($td). "\n";

    $td = mcrypt_module_open ('cast-256', '', 'ecb', '');
    echo mcrypt_enc_get_modes_name($td). "\n";
?>

Prints:
CFB
ECB
```

mcrypt_enc_get_supported_key_sizes

(PHP 4 >= 4.0.2)

mcrypt_enc_get_supported_key_sizes -- Returns an array with the supported keysizes of the opened algorithm

Description

array **mcrypt_enc_get_supported_key_sizes** (resource td)

Returns an array with the key sizes supported by the algorithm specified by the encryption descriptor. If it returns an empty

array then all key sizes between 1 and **mcrypt_enc_get_key_size()** are supported by the algorithm.

Example 1. mcrypt_enc_get_supported_key_sizes() example

```
<?php
    $td = mcrypt_module_open ('rijndael-256', '', 'ecb', '');
    var_dump (mcrypt_enc_get_supported_key_sizes($td));
?>

This will print:

array(3) {
  [0]=>
  int(16)
  [1]=>
  int(24)
  [2]=>
  int(32)
}
?>
```

mcrypt_enc_is_block_algorithm_mode

(PHP 4 >= 4.0.2)

mcrypt_enc_is_block_algorithm_mode -- Checks whether the encryption of the opened mode works on blocks

Description

bool **mcrypt_enc_is_block_algorithm_mode** (resource td)

This function returns **TRUE** if the mode is for use with block algorithms, otherwise it returns **FALSE**. (eg. **FALSE** for stream, and **TRUE** for cbc, cfb, ofb).

mcrypt_enc_is_block_algorithm

(PHP 4 >= 4.0.2)

mcrypt_enc_is_block_algorithm -- Checks whether the algorithm of the opened mode is a block algorithm

Description

bool **mcrypt_enc_is_block_algorithm** (resource td)

This function returns **TRUE** if the algorithm is a block algorithm, or **FALSE** if it is a stream algorithm.

mcrypt_enc_is_block_mode

(PHP 4 >= 4.0.2)

mcrypt_enc_is_block_mode -- Checks whether the opened mode outputs blocks

Description

bool **mcrypt_enc_is_block_mode** (resource td)

This function returns **TRUE** if the mode outputs blocks of bytes or **FALSE** if it outputs bytes. (eg. **TRUE** for cbc and ecb, and **FALSE** for cfb and stream).

mcrypt_enc_self_test

(PHP 4 >= 4.0.2)

mcrypt_enc_self_test -- This function runs a self test on the opened module

Description

bool **mcrypt_enc_self_test** (resource td)

This function runs the self test on the algorithm specified by the descriptor *td*. If the self test succeeds it returns **FALSE**. In case of an error, it returns **TRUE**.

mcrypt_encrypt

(PHP 4 >= 4.0.2)

mcrypt_encrypt -- Encrypts plaintext with given parameters

Description

string **mcrypt_encrypt** (string cipher, string key, string data, string mode [, string iv])

mcrypt_encrypt() encrypts the data and returns the encrypted data.

Cipher is one of the MCRYPT_ciphername constants of the name of the algorithm as string.

Key is the key with which the data will be encrypted. If it's smaller that the required keysize, it is padded with '\0'. It is better not to use ASCII strings for keys. It is recommended to use the mhash functions to create a key from a string.

Data is the data that will be encrypted with the given cipher and mode. If the size of the data is not n * blocksize, the data will be padded with '\0'. The returned crypttext can be larger that the size of the data that is given by *data*.

Mode is one of the MCRYPT_MODE_modename constants of one of "ecb", "cbc", "cfb", "ofb", "nofb" or "stream".

The *IV* parameter is used for the initialisation in CBC, CFB, OFB modes, and in some algorithms in STREAM mode. If you do not supply an IV, while it is needed for an algorithm, the function issues a warning and uses an IV with all bytes set to '\0'.

Example 1. mcrypt_encrypt() Example

```
<?php
    $iv = mcrypt_create_iv (mcrypt_get_iv_size (MCRYPT_RIJNDAEL_256, MCRYPT_MODE_ECB), MCRYPT_RAND);
    $key = "This is a very secret key";
    $text = "Meet me at 11 o'clock behind the monument.";
    echo strlen ($text)."\n";

    $crypttext = mcrypt_encrypt (MCRYPT_RIJNDAEL_256, $key, $text, MCRYPT_MODE_ECB, $iv);
    echo strlen ($crypttext)."\n";
?>
```

The above example will print out:

```
42
64
```

See also **mcrypt_module_open()** for a more advanced API and an example.

mcrypt_generic_deinit

(PHP 4 >= 4.1.1)

mcrypt_generic_deinit -- This function deinitializes an encryption module

Description

bool **mcrypt_generic_deinit** (resource td)

This function terminates encryption specified by the encryption descriptor (*td*). It clears all buffers, but does not close the module. You need to call **mcrypt_module_close()** yourself. (But PHP does this for you at the end of the script. Returns **FALSE** on error, or **TRUE** on success.

See for an example **mcrypt_module_open()** and the entry on **mcrypt_generic_init()**.

mcrypt_generic_end

(PHP 4 >= 4.0.2)

mcrypt_generic_end -- This function terminates encryption

Description

bool **mcrypt_generic_end** (resource td)

This function is deprecated, use **mcrypt_generic_deinit()** instead. It can cause crashes when used with **mcrypt_module_close()** due to multiple buffer frees.

This function terminates encryption specified by the encryption descriptor (*td*). Actually it clears all buffers, and closes all the modules used. Returns **FALSE** on error, or **TRUE** on success.

mcrypt_generic_init

(PHP 4 >= 4.0.2)

mcrypt_generic_init -- This function initializes all buffers needed for encryption

Description

int **mcrypt_generic_init** (resource td, string key, string iv)

The maximum length of the key should be the one obtained by calling **mcrypt_enc_get_key_size()** and every value smaller than this is legal. The IV should normally have the size of the algorithms block size, but you must obtain the size by calling **mcrypt_enc_get_iv_size()**. IV is ignored in ECB. IV MUST exist in CFB, CBC, STREAM, nOFB and OFB modes. It needs to be random and unique (but not secret). The same IV must be used for encryption/decryption. If you do not want to use it you should set it to zeros, but this is not recommended. The function returns a negative value on error.

You need to call this function before every call to **mcrypt_generic()** or **mdecrypt_generic()**.

See for an example **mcrypt_module_open()** and the entry on **mcrypt_generic_deinit()**.

mcrypt_generic

(PHP 4 >= 4.0.2)

mcrypt_generic -- This function encrypts data

Description

string **mcrypt_generic** (resource td, string data)

This function encrypts data. The data is padded with "\0" to make sure the length of the data is n * blocksize. This function returns the encrypted data. Note that the length of the returned string can in fact be longer then the input, due to the padding of the data.

The encryption handle should alwayws be initialized with **mcrypt_generic_init()** with a key and an IV before calling this function. Where the encryption is done, you should free the encryption buffers by calling **mcrypt_generic_deinit()**. See **mcrypt_module_open()** for an example.

See also **mdecrypt_generic()**, **mcrypt_generic_init()** and **mcrypt_generic_deinit()**.

mcrypt_get_block_size

(PHP 3>= 3.0.8, PHP 4)

mcrypt_get_block_size -- Get the block size of the specified cipher

Description

int **mcrypt_get_block_size** (int cipher)

int **mcrypt_get_block_size** (string cipher, string module)

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x or 2.5.x.

mcrypt_get_block_size() is used to get the size of a block of the specified *cipher* (in combination with an encryption mode).

This example shows how to use this function when linked against libmcrypt 2.4.x and 2.5.x.

Example 1. mcrypt_get_block_size() example

```
<?php
    echo mcrypt_get_block_size ('tripledes', 'ecb');
?>

Prints:
8
```

See also: **mcrypt_get_key_size()** and **mcrypt_encrypt()**.

mcrypt_get_cipher_name

(PHP 3>= 3.0.8, PHP 4)

mcrypt_get_cipher_name -- Get the name of the specified cipher

Description

string **mcrypt_get_cipher_name** (int cipher)

string **mcrypt_get_cipher_name** (string cipher)

mcrypt_get_cipher_name() is used to get the name of the specified cipher.

mcrypt_get_cipher_name() takes the cipher number as an argument (libmcrypt 2.2.x) or takes the cipher name as an argument (libmcrypt 2.4.x or higher) and returns the name of the cipher or **FALSE**, if the cipher does not exist.

Example 1. mcrypt_get_cipher_name() Example

```
<?php
   $cipher = MCRYPT_TripleDES;

   echo mcrypt_get_cipher_name ($cipher);
?>
```

The above example will produce:

```
3DES
```

mcrypt_get_iv_size

(PHP 4 >= 4.0.2)

mcrypt_get_iv_size -- Returns the size of the IV belonging to a specific cipher/mode combination

Description

int **mcrypt_get_iv_size** (resource td)

int **mcrypt_get_iv_size** (string cipher, string mode)

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x or higher.

mcrypt_get_iv_size() returns the size of the Initialisation Vector (IV) in bytes. On error the function returns **FALSE**. If the IV is ignored in the specified cipher/mode combination zero is returned.

cipher is one of the MCRYPT_ciphername constants of the name of the algorithm as string.

mode is one of the MCRYPT_MODE_modename constants of one of "ecb", "cbc", "cfb", "ofb", "nofb" or "stream".

td is the resource that is returned by **mcrypt_module_open()**.

Example 1. mcrypt_create_iv() example

```
<?php
    $size = mcrypt_get_iv_size (MCRYPT_CAST_256, MCRYPT_MODE_CFB);

    $size = mcrypt_get_iv_size ('des', 'ecb');
?>
```

See also: **mcrypt_create_iv()**

mcrypt_get_key_size

(PHP 3>= 3.0.8, PHP 4)

mcrypt_get_key_size -- Get the key size of the specified cipher

Description

int **mcrypt_get_key_size** (int cipher)

int **mcrypt_get_key_size** (string cipher, string module)

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x or 2.5.x.

mcrypt_get_key_size() is used to get the size of a key of the specified *cipher* (in combination with an encryption mode).

This example shows how to use this function when linked against libmcrypt 2.4.x and 2.5.x.

Example 1. mcrypt_get_block_size() example

```
<?php
    echo mcrypt_get_key_size ('tripledes', 'ecb');
?>

Prints:
24
```

See also: **mcrypt_get_block_size()** and **mcrypt_encrypt()**.

mcrypt_list_algorithms

(PHP 4 >= 4.0.2)

mcrypt_list_algorithms -- Get an array of all supported ciphers

Description

array **mcrypt_list_algorithms** ([string lib_dir])

mcrypt_list_algorithms() is used to get an array of all supported algorithms in the *lib_dir* parameter.

mcrypt_list_algorithms() takes an optional *lib_dir* parameter which specifies the directory where all algorithms are located. If not specifies, the value of the mcrypt.algorithms_dir `php.ini` directive is used.

Example 1. mcrypt_list_algorithms() Example

```
<?php
    $algorithms = mcrypt_list_algorithms ("/usr/local/lib/libmcrypt");
```

```
    foreach ($algorithms as $cipher) {
        echo "$cipher<br />\n";
    }
?>
```

The above example will produce a list with all supported algorithms in the "/usr/local/lib/libmcrypt" directory.

mcrypt_list_modes

(PHP 4 >= 4.0.2)

mcrypt_list_modes -- Get an array of all supported modes

Description

array **mcrypt_list_modes** ([string lib_dir])

mcrypt_list_modes() is used to get an array of all supported modes in the *lib_dir*.

mcrypt_list_modes() takes as optional parameter a directory which specifies the directory where all modes are located. If not specifies, the value of the mcrypt.modes_dir `php.ini` directive is used.

Example 1. mcrypt_list_modes() Example

```
<?php
    $modes = mcrypt_list_modes ();

    foreach ($modes as $mode) {
        echo "$mode <br />\n";
    }
?>
```

The above example will produce a list with all supported algorithms in the default mode directory. If it is not set with the ini directive mcrypt.modes_dir, the default directory of mcrypt is used (which is /usr/local/lib/libmcrypt).

mcrypt_module_close

(PHP 4 >= 4.0.2)

mcrypt_module_close -- Close the mcrypt module

Description

bool **mcrypt_module_close** (resource td)

This function closes the specified encryption handle.

See **mcrypt_module_open()** for an example.

mcrypt_module_get_algo_block_size

(PHP 4 >= 4.0.2)

mcrypt_module_get_algo_block_size -- Returns the blocksize of the specified algorithm

Description

int **mcrypt_module_get_algo_block_size** (string algorithm [, string lib_dir])

This function returns the block size of the algorithm specified in bytes. The optional *lib_dir* parameter can contain the location where the mode module is on the system.

mcrypt_module_get_algo_key_size

(PHP 4 >= 4.0.2)

mcrypt_module_get_algo_key_size -- Returns the maximum supported keysize of the opened mode

Description

int **mcrypt_module_get_algo_key_size** (string algorithm [, string lib_dir])

This function returns the maximum supported key size of the algorithm specified in bytes. The optional *lib_dir* parameter can contain the location where the mode module is on the system.

mcrypt_module_get_supported_key_sizes

(PHP 4 >= 4.0.2)

mcrypt_module_get_supported_key_sizes -- Returns an array with the supported keysizes of the opened algorithm

Description

array **mcrypt_module_get_supported_key_sizes** (string algorithm [, string lib_dir])

Returns an array with the key sizes supported by the specified algorithm. If it returns an empty array then all key sizes between 1 and **mcrypt_module_get_algo_key_size()** are supported by the algorithm. The optional *lib_dir* parameter can contain the location where the mode module is on the system.

See also **mcrypt_enc_get_supported_key_sizes()** which is used on open encryption modules.

mcrypt_module_is_block_algorithm_mode

(PHP 4 >= 4.0.2)

mcrypt_module_is_block_algorithm_mode -- This function returns if the the specified module is a block algorithm or not

Description

bool **mcrypt_module_is_block_algorithm_mode** (string mode [, string lib_dir])

This function returns **TRUE** if the mode is for use with block algorithms, otherwise it returns **FALSE**. (eg. **FALSE** for stream, and **TRUE** for cbc, cfb, ofb). The optional *lib_dir* parameter can contain the location where the mode module is on the system.

mcrypt_module_is_block_algorithm

(PHP 4 >= 4.0.2)

mcrypt_module_is_block_algorithm -- This function checks whether the specified algorithm is a block algorithm

Description

bool **mcrypt_module_is_block_algorithm** (string algorithm [, string lib_dir])

This function returns **TRUE** if the specified algorithm is a block algorithm, or **FALSE** is it is a stream algorithm. The optional *lib_dir* parameter can contain the location where the algorithm module is on the system.

mcrypt_module_is_block_mode

(PHP 4 >= 4.0.2)

mcrypt_module_is_block_mode -- This function returns if the the specified mode outputs blocks or not

Description

bool **mcrypt_module_is_block_mode** (string mode [, string lib_dir])

This function returns **TRUE** if the mode outputs blocks of bytes or **FALSE** if it outputs just bytes. (eg. **TRUE** for cbc and ecb, and **FALSE** for cfb and stream). The optional *lib_dir* parameter can contain the location where the mode module is on the system.

mcrypt_module_open

(PHP 4 >= 4.0.2)

mcrypt_module_open -- This function opens the module of the algorithm and the mode to be used

Description

resource **mcrypt_module_open** (string algorithm, string algorithm_directory, string mode, string mode_directory)

This function opens the module of the algorithm and the mode to be used. The name of the algorithm is specified in algorithm, eg. "twofish" or is one of the MCRYPT_ciphername constants. The module is closed by calling **mcrypt_module_close()**. Normally it returns an encryption descriptor, or **FALSE** on error.

The *algorithm_directory* and *mode_directory* are used to locate the encryption modules. When you supply a directory name, it is used. When you set one of these to the empty string (""), the value set by the *mcrypt.algorithms_dir* or *mcrypt.modes_dir* ini-directive is used. When these are not set, the default directories that are used are the ones that were compiled in into libmcrypt (usally /usr/local/lib/libmcrypt).

Example 1. mcrypt_module_open() Example

```
<?php
    $td = mcrypt_module_open (MCRYPT_DES, '', MCRYPT_MODE_ECB, '/usr/lib/mcrypt-modes');
    $td = mcrypt_module_open ('rijndael-256', '', 'ofb', '');
?>
```

The first line in the example above will try to open the DES cipher from the default directory and the EBC mode from the directory `/usr/lib/mcrypt-modes`. The second example uses strings as name for the cipher an dmode, this only works when the extension is linked against libmcrypt 2.4.x or 2.5.x.

Example 2. Using mcrypt_module_open() in encryption

```
<?php
    /* Open the cipher */
    $td = mcrypt_module_open ('rijndael-256', '', 'ofb', '');

    /* Create the IV and determine the keysize length */
    $iv = mcrypt_create_iv (mcrypt_enc_get_iv_size($td), MCRYPT_DEV_RANDOM);
    $ks = mcrypt_enc_get_key_size ($td);

    /* Create key */
    $key = substr (md5 ('very secret key'), 0, $ks);

    /* Intialize encryption */
    mcrypt_generic_init ($td, $key, $iv);

    /* Encrypt data */
    $encrypted = mcrypt_generic ($td, 'This is very important data');

    /* Terminate encryption handler */
    mcrypt_generic_deinit ($td);

    /* Initialize encryption module for decryption */
    mcrypt_generic_init ($td, $key, $iv);

    /* Decrypt encrypted string */
    $decrypted = mdecrypt_generic ($td, $encrypted);

    /* Terminate decryption handle and close module */
    mcrypt_generic_deinit ($td);
    mcrypt_module_close ($td);

    /* Show string */
    echo trim ($decrypted)."\n";
?>
```

The first line in the example above will try to open the DES cipher from the default directory and the EBC mode from the directory `/usr/lib/mcrypt-modes`. The second example uses strings as name for the cipher an dmode, this only works when the extension is linked against libmcrypt 2.4.x or 2.5.x.

See also **mcrypt_module_close()**, **mcrypt_generic()**, **mdecrypt_generic()**, **mcrypt_generic_init()** and **mcrypt_generic_deinit()**.

mcrypt_module_self_test

(PHP 4 >= 4.0.2)

mcrypt_module_self_test -- This function runs a self test on the specified module

Description

bool **mcrypt_module_self_test** (string algorithm [, string lib_dir])

This function runs the self test on the algorithm specified. The optional *lib_dir* parameter can contain the location of where the algorithm module is on the system.

The function returns **TRUE** if the self test succeeds, or **FALSE** when if fails.

mcrypt_ofb

(PHP 3>= 3.0.8, PHP 4)

mcrypt_ofb -- Encrypt/decrypt data in OFB mode

Description

string **mcrypt_ofb** (int cipher, string key, string data, int mode, string iv)

string **mcrypt_ofb** (string cipher, string key, string data, int mode [, string iv])

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x or higher.

This function should not be used anymore, see **mcrypt_generic()** and **mdecrypt_generic()** for replacements.

mdecrypt_generic

(PHP 4 >= 4.0.2)

mdecrypt_generic -- This function decrypts data

Description

string **mdecrypt_generic** (resource td, string data)

This function decrypts data. Note that the length of the returned string can in fact be longer then the unencrypted string, due to the padding of the data.

Example 1. mdecrypt_generic() Example

```
<?php
    /* Data */
    $key = 'this is a very long key, even too long for the cipher';
    $plain_text = 'very important data';

    /* Open module, and create IV */
    $td = mcrypt_module_open ('des', '', 'ecb', '');
    $key = substr ($key, 0, mcrypt_enc_get_key_size ($td));
    $iv_size = mcrypt_enc_get_iv_size ($td);
    $iv = mcrypt_create_iv ($iv_size, MCRYPT_RAND);

    /* Initialize encryption handle */
    if (mcrypt_generic_init ($td, $key, $iv) != -1) {
```

```
        /* Encrypt data */
        $c_t = mcrypt_generic ($td, $plain_text);
        mcrypt_generic_deinit ($td);

        /* Reinitialize buffers for decryption */
        mcrypt_generic_init ($td, $key, $iv);
        $p_t = mdecrypt_generic ($td, $c_t);

        /* Clean up */
        mcrypt_generic_deinit ($td);
        mcrypt_module_close ($td);
    }

    if (strncmp ($p_t, $plain_text, strlen($plain_text)) == 0) {
        echo "ok\n";
    } else {
        echo "error\n";
    }
?>
```

The above example shows how to check if the data before the encryption is the same as the data after the decryption. It is very important to reinitialize the encryption buffer with **mcrypt_generic_init()** before you try to decrypt the data.

The decryption handle should alwayws be initialized with **mcrypt_generic_init()** with a key and an IV before calling this function. Where the encryption is done, you should free the encryption buffers by calling **mcrypt_generic_deinit()**. See **mcrypt_module_open()** for an example.

See also **mcrypt_generic()**, **mcrypt_generic_init()** and **mcrypt_generic_deinit()**.

LV. MCVE Payment Functions

Introduction

These functions interface the MCVE API (libmcve), allowing you to work directly with MCVE from your PHP scripts. MCVE is Main Street Softworks' solution to direct credit card processing. It lets you directly address the credit card clearing houses via your *nix box, modem and/or internet connection (bypassing the need for an additional service such as Authorize.Net or Pay Flow Pro). Using the MCVE module for PHP, you can process credit cards directly through MCVE via your PHP scripts. The following references will outline the process.

> **Note:** MCVE is the replacement for RedHat's CCVS. They contracted with RedHat in late 2001 to migrate all existing clientelle to the MCVE platform.

> **Note:** This extension is not available on Windows platforms.

Installation

To enable MCVE Support in PHP, first verify your LibMCVE installation directory. You will then need to configure PHP with the `--with-mcve` option. If you use this option without specifying the path to your MCVE installation, PHP will attempt to look in the default LibMCVE Install location (`/usr/local`). If MCVE is in a non-standard location, run configure with: `--with-mcve=$mcve_path`, where $mcve_path is the path to your MCVE installation. Please note that MCVE support requires that $mcve_path/lib and $mcve_path/include exist, and include `mcve.h` under the include directory and `libmcve.so` and/or `libmcve.a` under the lib directory.

Since MCVE has true server/client separation, there are no additional requirements for running PHP with MCVE support. To test your MCVE extension in PHP, you may connect to testbox.mcve.com on port 8333 for IP, or port 8444 for SSL using the MCVE PHP API. Use 'vitale' for your username, and 'test' for your password. Additional information about test facilities are available at www.mcve.com.

See Also

Additional documentation about MCVE's PHP API can be found at http://www.mcve.com/docs/phpapi.pdf. Main Street's documentation is complete and should be the primary reference for functions.

Table of Contents

mcve_checkstatus -- Check to see if a transaction has completed
mcve_chkpwd -- Verify Password
mcve_chngpwd -- Change the system administrator's password
mcve_completeauthorizations -- Number of complete authorizations in queue, returning an array of their identifiers
mcve_connect -- Establish the connection to MCVE
mcve_connectionerror -- Get a textual representation of why a connection failed
mcve_deleteresponse -- Delete specified transaction from MCVE_CONN structure
mcve_deletetrans -- Delete specified transaction from MCVE_CONN structure
mcve_deleteusersetup -- Deallocate data associated with usersetup structure
mcve_deluser -- Delete an MCVE user account
mcve_destroyconn -- Destroy the connection and MCVE_CONN structure
mcve_destroyengine -- Free memory associated with IP/SSL connectivity
mcve_disableuser -- Disable an active MCVE user account
mcve_edituser -- Edit MCVE user using usersetup structure
mcve_enableuser -- Enable an inactive MCVE user account
mcve_force -- Send a FORCE to MCVE. (typically, a phone-authorization)
mcve_getcell -- Get a specific cell from a comma delimited response by column name
mcve_getcellbynum -- Get a specific cell from a comma delimited response by column number
mcve_getcommadelimited -- Get the RAW comma delimited data returned from MCVE
mcve_getheader -- Get the name of the column in a comma-delimited response
mcve_getuserarg -- Grab a value from usersetup structure
mcve_getuserparam -- Get a user response parameter
mcve_gft -- Audit MCVE for Failed transactions
mcve_gl -- Audit MCVE for settled transactions
mcve_gut -- Audit MCVE for Unsettled Transactions
mcve_initconn -- Create and initialize an MCVE_CONN structure
mcve_initengine -- Ready the client for IP/SSL Communication
mcve_initusersetup -- Initialize structure to store user data
mcve_iscommadelimited -- Checks to see if response is comma delimited
mcve_liststats -- List statistics for all users on MCVE system
mcve_listusers -- List all users on MCVE system
mcve_maxconntimeout -- The maximum amount of time the API will attempt a connection to MCVE
mcve_monitor -- Perform communication with MCVE (send/receive data) Non-blocking
mcve_numcolumns -- Number of columns returned in a comma delimited response
mcve_numrows -- Number of rows returned in a comma delimited response
mcve_override -- Send an OVERRIDE to MCVE
mcve_parsecommadelimited -- Parse the comma delimited response so mcve_getcell, etc will work
mcve_ping -- Send a ping request to MCVE
mcve_preauth -- Send a PREAUTHORIZATION to MCVE
mcve_preauthcompletion -- Complete a PREAUTHORIZATION... Ready it for settlement
mcve_qc -- Audit MCVE for a list of transactions in the outgoing queue
mcve_responseparam -- Get a custom response parameter
mcve_return -- Issue a RETURN or CREDIT to MCVE
mcve_returncode -- Grab the exact return code from the transaction
mcve_returnstatus -- Check to see if the transaction was successful
mcve_sale -- Send a SALE to MCVE
mcve_setblocking -- Set blocking/non-blocking mode for connection
mcve_setdropfile -- Set the connection method to Drop-File
mcve_setip -- Set the connection method to IP
mcve_setssl -- Set the connection method to SSL
mcve_settimeout -- Set maximum transaction time (per trans)
mcve_settle -- Issue a settlement command to do a batch deposit
mcve_text_avs -- Get a textual representation of the return_avs
mcve_text_code -- Get a textual representation of the return_code
mcve_text_cv -- Get a textual representation of the return_cv
mcve_transactionauth -- Get the authorization number returned for the transaction (alpha-numeric)
mcve_transactionavs -- Get the Address Verification return status
mcve_transactionbatch -- Get the batch number associated with the transaction
mcve_transactioncv -- Get the CVC2/CVV2/CID return status
mcve_transactionid -- Get the unique system id for the transaction
mcve_transactionitem -- Get the ITEM number in the associated batch for this transaction
mcve_transactionssent -- Check to see if outgoing buffer is clear
mcve_transactiontext -- Get verbiage (text) return from MCVE or processing institution
mcve_transinqueue -- Number of transactions in client-queue
mcve_transnew -- Start a new transaction
mcve_transparam -- Add a parameter to a transaction
mcve_transsend -- Finalize and send the transaction
mcve_ub -- Get a list of all Unsettled batches
mcve_uwait -- Wait x microsecs
mcve_verifyconnection -- Set whether or not to PING upon connect to verify connection

mcve_verifysslcert -- Set whether or not to verify the server ssl certificate
mcve_void -- VOID a transaction in the settlement queue

mcve_adduser

(PHP 4 >= 4.2.0)

mcve_adduser -- Add an MCVE user using usersetup structure

Description

int **mcve_adduser** (resource conn, string admin_password, int usersetup)

Warning
This function is currently not documented; only the argument list is available.

mcve_adduserarg

(PHP 4 >= 4.2.0)

mcve_adduserarg -- Add a value to user configuration structure

Description

int **mcve_adduserarg** (resource usersetup, int argtype, string argval)

Warning
This function is currently not documented; only the argument list is available.

mcve_bt

(PHP 4 >= 4.2.0)

mcve_bt -- Get unsettled batch totals

Description

int **mcve_bt** (resource conn, string username, string password)

Warning
This function is currently not documented; only the argument list is available.

mcve_checkstatus

(PHP 4 >= 4.2.0)

mcve_checkstatus -- Check to see if a transaction has completed

Description

int **mcve_checkstatus** (resource conn, int identifier)

Warning
This function is currently not documented; only the argument list is available.

mcve_chkpwd

(PHP 4 >= 4.2.0)

mcve_chkpwd -- Verify Password

Description

int **mcve_chkpwd** (resource conn, string username, string password)

Warning
This function is currently not documented; only the argument list is available.

mcve_chngpwd

(PHP 4 >= 4.2.0)

mcve_chngpwd -- Change the system administrator's password

Description

int **mcve_chngpwd** (resource conn, string admin_password, string new_password)

Warning
This function is currently not documented; only the argument list is available.

mcve_completeauthorizations

(PHP 4 >= 4.2.0)

mcve_completeauthorizations -- Number of complete authorizations in queue, returning an array of their identifiers

Description

int **mcve_completeauthorizations** (resource conn, int &array)

Warning
This function is currently not documented; only the argument list is available.

mcve_connect

(PHP 4 >= 4.2.0)

mcve_connect -- Establish the connection to MCVE

Description

int **mcve_connect** (resource conn)

Warning
This function is currently not documented; only the argument list is available.

mcve_connectionerror

(PHP 4 >= 4.3.0)

mcve_connectionerror -- Get a textual representation of why a connection failed

Description

string **mcve_connectionerror** (resource conn)

Warning
This function is currently not documented; only the argument list is available.

mcve_deleteresponse

(PHP 4 >= 4.2.0)

mcve_deleteresponse -- Delete specified transaction from MCVE_CONN structure

Description

bool **mcve_deleteresponse** (resource conn, int identifier)

Warning
This function is currently not documented; only the argument list is available.

mcve_deletetrans

(PHP 4 >= 4.3.0)

mcve_deletetrans -- Delete specified transaction from MCVE_CONN structure

Description

bool **mcve_deletetrans** (resource conn, int identifier)

Warning
This function is currently not documented; only the argument list is available.

mcve_deleteusersetup

(PHP 4 >= 4.2.0)

mcve_deleteusersetup -- Deallocate data associated with usersetup structure

Description

void **mcve_deleteusersetup** (resource usersetup)

Warning
This function is currently not documented; only the argument list is available.

mcve_deluser

(PHP 4 >= 4.2.0)

mcve_deluser -- Delete an MCVE user account

Description

int **mcve_deluser** (resource conn, string admin_password, string username)

Warning
This function is currently not documented; only the argument list is available.

mcve_destroyconn

(PHP 4 >= 4.2.0)

mcve_destroyconn -- Destroy the connection and MCVE_CONN structure

Description

void **mcve_destroyconn** (resource conn)

Warning
This function is currently not documented; only the argument list is available.

mcve_destroyengine

(PHP 4 >= 4.2.0)

mcve_destroyengine -- Free memory associated with IP/SSL connectivity

Description

void **mcve_destroyengine** (void)

Warning
This function is currently not documented; only the argument list is available.

mcve_disableuser

(PHP 4 >= 4.2.0)

mcve_disableuser -- Disable an active MCVE user account

Description

int **mcve_disableuser** (resource conn, string admin_password, string username)

Warning
This function is currently not documented; only the argument list is available.

mcve_edituser

(PHP 4 >= 4.2.0)

mcve_edituser -- Edit MCVE user using usersetup structure

Description

int **mcve_edituser** (resource conn, string admin_password, int usersetup)

Warning
This function is currently not documented; only the argument list is available.

mcve_enableuser

(PHP 4 >= 4.2.0)

mcve_enableuser -- Enable an inactive MCVE user account

Description

int **mcve_enableuser** (resource conn, string admin_password, string username)

Warning
This function is currently not documented; only the argument list is available.

mcve_force

(PHP 4 >= 4.2.0)

mcve_force -- Send a FORCE to MCVE. (typically, a phone-authorization)

Description

int **mcve_force** (resiurce conn, string username, string password, string trackdata, string account, string expdate, float amount, string authcode, string comments, string clerkid, string stationid, int ptrannum)

Warning
This function is currently not documented; only the argument list is available.

mcve_getcell

(PHP 4 >= 4.2.0)

mcve_getcell -- Get a specific cell from a comma delimited response by column name

Description

string **mcve_getcell** (resource conn, int identifier, string column, int row)

Warning
This function is currently not documented; only the argument list is available.

mcve_getcellbynum

(PHP 4 >= 4.2.0)

mcve_getcellbynum -- Get a specific cell from a comma delimited response by column number

Description

string **mcve_getcellbynum** (resource conn, int identifier, int column, int row)

Warning
This function is currently not documented; only the argument list is available.

mcve_getcommadelimited

(PHP 4 >= 4.2.0)

mcve_getcommadelimited -- Get the RAW comma delimited data returned from MCVE

Description

string **mcve_getcommadelimited** (resource conn, int identifier)

Warning
This function is currently not documented; only the argument list is available.

mcve_getheader

(PHP 4 >= 4.2.0)

mcve_getheader -- Get the name of the column in a comma-delimited response

Description

string **mcve_getheader** (resource conn, int identifier, int column_num)

Warning
This function is currently not documented; only the argument list is available.

mcve_getuserarg

(PHP 4 >= 4.2.0)

mcve_getuserarg -- Grab a value from usersetup structure

Description

string **mcve_getuserarg** (resource usersetup, int argtype)

Warning
This function is currently not documented; only the argument list is available.

mcve_getuserparam

(PHP 4 >= 4.3.0)

mcve_getuserparam -- Get a user response parameter

Description

string **mcve_getuserparam** (resource conn, long identifier, int key)

Warning
This function is currently not documented; only the argument list is available.

mcve_gft

(PHP 4 >= 4.2.0)

mcve_gft -- Audit MCVE for Failed transactions

Description

int **mcve_gft** (resource conn, string username, string password, int type, string account, string clerkid, string stationid, string comments, int ptrannum, string startdate, string enddate)

Warning
This function is currently not documented; only the argument list is available.

mcve_gl

(PHP 4 >= 4.2.0)

mcve_gl -- Audit MCVE for settled transactions

Description

int **mcve_gl** (int conn, string username, string password, int type, string account, string batch, string clerkid, string stationid, string comments, int ptrannum, string startdate, string enddate)

Warning
This function is currently not documented; only the argument list is available.

mcve_gut

(PHP 4 >= 4.2.0)

mcve_gut -- Audit MCVE for Unsettled Transactions

Description

int **mcve_gut** (resource conn, string username, string password, int type, string account, string clerkid, string stationid, string comments, int ptrannum, string startdate, string enddate)

Warning
This function is currently not documented; only the argument list is available.

mcve_initconn

(PHP 4 >= 4.2.0)

mcve_initconn -- Create and initialize an MCVE_CONN structure

Description

resource **mcve_initconn** (void)

Warning
This function is currently not documented; only the argument list is available.

mcve_initengine

(PHP 4 >= 4.2.0)

mcve_initengine -- Ready the client for IP/SSL Communication

Description

int **mcve_initengine** (string location)

Warning
This function is currently not documented; only the argument list is available.

mcve_initusersetup

(PHP 4 >= 4.2.0)

mcve_initusersetup -- Initialize structure to store user data

Description

resource **mcve_initusersetup** (void)

Warning
This function is currently not documented; only the argument list is available.

mcve_iscommadelimited

(PHP 4 >= 4.2.0)

mcve_iscommadelimited -- Checks to see if response is comma delimited

Description

int **mcve_iscommadelimited** (resource conn, int identifier)

Warning
This function is currently not documented; only the argument list is available.

mcve_liststats

(PHP 4 >= 4.2.0)

mcve_liststats -- List statistics for all users on MCVE system

Description

int **mcve_liststats** (resource conn, string admin_password)

Warning
This function is currently not documented; only the argument list is available.

mcve_listusers

(PHP 4 >= 4.2.0)

mcve_listusers -- List all users on MCVE system

Description

int **mcve_listusers** (resource conn, string admin_password)

Warning
This function is currently not documented; only the argument list is available.

mcve_maxconntimeout

(PHP 4 >= 4.3.0)

mcve_maxconntimeout -- The maximum amount of time the API will attempt a connection to MCVE

Description

bool **mcve_maxconntimeout** (resource conn, int secs)

Warning
This function is currently not documented; only the argument list is available.

mcve_monitor

(PHP 4 >= 4.2.0)

mcve_monitor -- Perform communication with MCVE (send/receive data) Non-blocking

Description

int **mcve_monitor** (resource conn)

Warning
This function is currently not documented; only the argument list is available.

mcve_numcolumns

(PHP 4 >= 4.2.0)

mcve_numcolumns -- Number of columns returned in a comma delimited response

Description

int **mcve_numcolumns** (resource conn, int identifier)

Warning
This function is currently not documented; only the argument list is available.

mcve_numrows

(PHP 4 >= 4.2.0)

mcve_numrows -- Number of rows returned in a comma delimited response

Description

int **mcve_numrows** (resource conn, int identifier)

Warning
This function is currently not documented; only the argument list is available.

mcve_override

(PHP 4 >= 4.2.0)

mcve_override -- Send an OVERRIDE to MCVE

Description

int **mcve_override** (resource conn, string username, string password, string trackdata, string account, string expdate, float amount, string street, string zip, string cv, string comments, string clerkid, string stationid, int ptrannum)

Warning
This function is currently not documented; only the argument list is available.

mcve_parsecommadelimited

(PHP 4 >= 4.2.0)

mcve_parsecommadelimited -- Parse the comma delimited response so mcve_getcell, etc will work

Description

int **mcve_parsecommadelimited** (resource conn, int identifier)

Warning
This function is currently not documented; only the argument list is available.

mcve_ping

(PHP 4 >= 4.3.0)

mcve_ping -- Send a ping request to MCVE

Description

int **mcve_ping** (resource conn)

Warning
This function is currently not documented; only the argument list is available.

mcve_preauth

(PHP 4 >= 4.2.0)

mcve_preauth -- Send a PREAUTHORIZATION to MCVE

Description

int **mcve_preauth** (resource conn, string username, string password, string trackdata, string account, string expdate, float amount, string street, string zip, string cv, string comments, string clerkid, string stationid, int ptrannum)

Warning

This function is currently not documented; only the argument list is available.

mcve_preauthcompletion

(PHP 4 >= 4.2.0)

mcve_preauthcompletion -- Complete a PREAUTHORIZATION... Ready it for settlement

Description

int **mcve_preauthcompletion** (resource conn, string username, string password, float finalamount, int sid, int ptrannum)

Warning
This function is currently not documented; only the argument list is available.

mcve_qc

(PHP 4 >= 4.2.0)

mcve_qc -- Audit MCVE for a list of transactions in the outgoing queue

Description

int **mcve_qc** (resource conn, string username, string password, string clerkid, string stationid, string comments, int ptrannum)

Warning
This function is currently not documented; only the argument list is available.

mcve_responseparam

(PHP 4 >= 4.3.0)

mcve_responseparam -- Get a custom response parameter

Description

string **mcve_responseparam** (resource conn, long identifier, string key)

Warning
This function is currently not documented; only the argument list is available.

mcve_return

(PHP 4 >= 4.2.0)

mcve_return -- Issue a RETURN or CREDIT to MCVE

Description

int **mcve_return** (int conn, string username, string password, string trackdata, string account, string expdate, float amount, string comments, string clerkid, string stationid, int ptrannum)

Warning
This function is currently not documented; only the argument list is available.

mcve_returncode

(PHP 4 >= 4.2.0)

mcve_returncode -- Grab the exact return code from the transaction

Description

int **mcve_returncode** (resource conn, int identifier)

Warning
This function is currently not documented; only the argument list is available.

mcve_returnstatus

(PHP 4 >= 4.2.0)

mcve_returnstatus -- Check to see if the transaction was successful

Description

int **mcve_returnstatus** (resource conn, int identifier)

Warning
This function is currently not documented; only the argument list is available.

mcve_sale

(PHP 4 >= 4.2.0)

mcve_sale -- Send a SALE to MCVE

Description

int **mcve_sale** (resource conn, string username, string password, string trackdata, string account, string expdate, float amount, string street, string zip, string cv, string comments, string clerkid, string stationid, int ptrannum)

Warning
This function is currently not documented; only the argument list is available.

mcve_setblocking

(PHP 4 >= 4.3.0)

mcve_setblocking -- Set blocking/non-blocking mode for connection

Description

int **mcve_setblocking** (resource conn, int tf)

Warning
This function is currently not documented; only the argument list is available.

mcve_setdropfile

(PHP 4 >= 4.2.0)

mcve_setdropfile -- Set the connection method to Drop-File

Description

int **mcve_setdropfile** (resource conn, string directory)

Warning
This function is currently not documented; only the argument list is available.

mcve_setip

(PHP 4 >= 4.2.0)

mcve_setip -- Set the connection method to IP

Description

int **mcve_setip** (resource conn, string host, int port)

Warning
This function is currently not documented; only the argument list is available.

mcve_setssl

(PHP 4 >= 4.2.0)

mcve_setssl -- Set the connection method to SSL

Description

int **mcve_setssl** (resource conn, string host, int port)

Warning
This function is currently not documented; only the argument list is available.

mcve_settimeout

(PHP 4 >= 4.2.0)

mcve_settimeout -- Set maximum transaction time (per trans)

Description

int **mcve_settimeout** (resource conn, int seconds)

Warning
This function is currently not documented; only the argument list is available.

mcve_settle

(PHP 4 >= 4.2.0)

mcve_settle -- Issue a settlement command to do a batch deposit

Description

int **mcve_settle** (resource conn, string username, string password, string batch)

Warning
This function is currently not documented; only the argument list is available.

mcve_text_avs

(PHP 4 >= 4.3.0)

mcve_text_avs -- Get a textual representation of the return_avs

Description

string **mcve_text_avs** (string code)

Warning
This function is currently not documented; only the argument list is available.

mcve_text_code

(PHP 4 >= 4.3.0)

mcve_text_code -- Get a textual representation of the return_code

Description

string **mcve_text_code** (string code)

Warning
This function is currently not documented; only the argument list is available.

mcve_text_cv

(PHP 4 >= 4.3.0)

mcve_text_cv -- Get a textual representation of the return_cv

Description

string **mcve_text_cv** (int code)

Warning
This function is currently not documented; only the argument list is available.

mcve_transactionauth

(PHP 4 >= 4.2.0)

mcve_transactionauth -- Get the authorization number returned for the transaction (alpha-numeric)

Description

string **mcve_transactionauth** (resource conn, int identifier)

Warning
This function is currently not documented; only the argument list is available.

mcve_transactionavs

(PHP 4 >= 4.2.0)

mcve_transactionavs -- Get the Address Verification return status

Description

int **mcve_transactionavs** (resource conn, int identifier)

Warning
This function is currently not documented; only the argument list is available.

mcve_transactionbatch

(PHP 4 >= 4.2.0)

mcve_transactionbatch -- Get the batch number associated with the transaction

Description

int **mcve_transactionbatch** (resource conn, int identifier)

Warning
This function is currently not documented; only the argument list is available.

mcve_transactioncv

(PHP 4 >= 4.2.0)

mcve_transactioncv -- Get the CVC2/CVV2/CID return status

Description

int **mcve_transactioncv** (resource conn, int identifier)

Warning
This function is currently not documented; only the argument list is available.

mcve_transactionid

(PHP 4 >= 4.2.0)

mcve_transactionid -- Get the unique system id for the transaction

Description

int **mcve_transactionid** (resource conn, int identifier)

Warning
This function is currently not documented; only the argument list is available.

mcve_transactionitem

(PHP 4 >= 4.2.0)

mcve_transactionitem -- Get the ITEM number in the associated batch for this transaction

Description

int **mcve_transactionitem** (resource conn, int identifier)

Warning
This function is currently not documented; only the argument list is available.

mcve_transactionssent

(PHP 4 >= 4.2.0)

mcve_transactionssent -- Check to see if outgoing buffer is clear

Description

int **mcve_transactionssent** (resource conn)

Warning
This function is currently not documented; only the argument list is available.

mcve_transactiontext

(PHP 4 >= 4.2.0)

mcve_transactiontext -- Get verbiage (text) return from MCVE or processing institution

Description

string **mcve_transactiontext** (resource conn, int identifier)

Warning
This function is currently not documented; only the argument list is available.

mcve_transinqueue

(PHP 4 >= 4.2.0)

mcve_transinqueue -- Number of transactions in client-queue

Description

int **mcve_transinqueue** (resource conn)

Warning
This function is currently not documented; only the argument list is available.

mcve_transnew

(PHP 4 >= 4.3.0)

mcve_transnew -- Start a new transaction

Description

int **mcve_transnew** (resource conn)

Warning
This function is currently not documented; only the argument list is available.

mcve_transparam

(PHP 4 >= 4.3.0)

mcve_transparam -- Add a parameter to a transaction

Description

int **mcve_transparam** (resource conn, long identifier, int key)

Warning
This function is currently not documented; only the argument list is available.

mcve_transsend

(PHP 4 >= 4.3.0)

mcve_transsend -- Finalize and send the transaction

Description

int **mcve_transsend** (resource conn, long identifier)

Warning
This function is currently not documented; only the argument list is available.

mcve_ub

(PHP 4 >= 4.2.0)

mcve_ub -- Get a list of all Unsettled batches

Description

int **mcve_ub** (resource conn, string username, string password)

Warning
This function is currently not documented; only the argument list is available.

mcve_uwait

(PHP 4 >= 4.3.0)

mcve_uwait -- Wait x microsecs

Description

int **mcve_uwait** (long microsecs)

Warning
This function is currently not documented; only the argument list is available.

mcve_verifyconnection

(PHP 4 >= 4.3.0)

mcve_verifyconnection -- Set whether or not to PING upon connect to verify connection

Description

bool **mcve_verifyconnection** (resource conn, int tf)

Warning
This function is currently not documented; only the argument list is available.

mcve_verifysslcert

(PHP 4 >= 4.3.0)

mcve_verifysslcert -- Set whether or not to verify the server ssl certificate

Description

bool **mcve_verifysslcert** (resource conn, int tf)

Warning
This function is currently not documented; only the argument list is available.

mcve_void

(PHP 4 >= 4.2.0)

mcve_void -- VOID a transaction in the settlement queue

Description

int **mcve_void** (resource conn, string username, string password, int sid, int ptrannum)

Warning
This function is currently not documented; only the argument list is available.

LVI. Mhash Functions

Introduction

These functions are intended to work with mhash. Mhash can be used to create checksums, message digests, message authentication codes, and more.

This is an interface to the mhash library. mhash supports a wide variety of hash algorithms such as MD5, SHA1, GOST, and many others. For a complete list of supported hashes, refer to the documentation of mhash. The general rule is that you can access

the hash algorithm from PHP with MHASH_HASHNAME. For example, to access TIGER you use the PHP constant MHASH_TIGER.

Requirements

To use it, download the mhash distribution from its web site and follow the included installation instructions.

Installation

You need to compile PHP with the `--with-mhash[=DIR]` parameter to enable this extension. DIR is the mhash install directory.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

Here is a list of hashes which are currently supported by mhash. If a hash is not listed here, but is listed by mhash as supported, you can safely assume that this documentation is outdated.

- **MHASH_MD5**
- **MHASH_SHA1**
- **MHASH_HAVAL256**
- **MHASH_HAVAL192**
- **MHASH_HAVAL160**
- **MHASH_HAVAL128**
- **MHASH_RIPEMD160**
- **MHASH_GOST**
- **MHASH_TIGER**
- **MHASH_CRC32**
- **MHASH_CRC32B**

Examples

Example 1. Compute the MD5 digest and hmac and print it out as hex

```
<?php
$input = "what do ya want for nothing?";
$hash = mhash (MHASH_MD5, $input);
print "The hash is ".bin2hex ($hash)."<br />\n";
$hash = mhash (MHASH_MD5, $input, "Jefe");
print "The hmac is ".bin2hex ($hash)."<br />\n";
?>
```

This will produce:

```
The hash is d03cb659cbf9192dcd066272249f8412
The hmac is 750c783e6ab0b503eaa86e310a5db738
```

Table of Contents

mhash_count

(PHP 3>= 3.0.9, PHP 4)

mhash_count -- Get the highest available hash id

Description

int **mhash_count** (void)

mhash_count() returns the highest available hash id. Hashes are numbered from o to this hash id.

Example 1. Traversing all hashes

```
<?php

$nr = mhash_count();

for ($i = 0; $i <= $nr; $i++) {
    echo sprintf ("The blocksize of %s is %d\n",
        mhash_get_hash_name ($i),
        mhash_get_block_size ($i));
}
?>
```

mhash_get_block_size

(PHP 3>= 3.0.9, PHP 4)

mhash_get_block_size -- Get the block size of the specified hash

Description

int **mhash_get_block_size** (int hash)

mhash_get_block_size() is used to get the size of a block of the specified *hash*.

mhash_get_block_size() takes one argument, the *hash* and returns the size in bytes or **FALSE**, if the *hash* does not exist.

mhash_get_hash_name

(PHP 3>= 3.0.9, PHP 4)

mhash_get_hash_name -- Get the name of the specified hash

Description

string **mhash_get_hash_name** (int hash)

mhash_get_hash_name() is used to get the name of the specified hash.

mhash_get_hash_name() takes the hash id as an argument and returns the name of the hash or **FALSE**, if the hash does not exist.

Example 1. mhash_get_hash_name() example

```
<?php
$hash = MHASH_MD5;

print mhash_get_hash_name ($hash);
?>
```

The above example will print out:

```
MD5
```

mhash_keygen_s2k

(PHP 4 >= 4.0.4)

mhash_keygen_s2k -- Generates a key

Description

string **mhash_keygen_s2k** (int hash, string password, string salt, int bytes)

mhash_keygen_s2k() generates a key that is *bytes* long, from a user given password. This is the Salted S2K algorithm as specified in the OpenPGP document (RFC 2440). That algorithm will use the specified *hash* algorithm to create the key. The *salt* must be different and random enough for every key you generate in order to create different keys. That salt must be known when you check the keys, thus it is a good idea to append the key to it. Salt has a fixed length of 8 bytes and will be padded with zeros if you supply less bytes.

Keep in mind that user supplied passwords are not really suitable to be used as keys in cryptographic algorithms, since users normally choose keys they can write on keyboard. These passwords use only 6 to 7 bits per character (or less). It is highly recommended to use some kind of tranformation (like this function) to the user supplied key.

mhash

(PHP 3>= 3.0.9, PHP 4)

mhash -- Compute hash

Description

string **mhash** (int hash, string data [, string key])

mhash() applies a hash function specified by *hash* to the *data* and returns the resulting hash (also called digest). If the *key* is specified it will return the resulting HMAC. HMAC is keyed hashing for message authentication, or simply a message digest that depends on the specified key. Not all algorithms supported in mhash can be used in HMAC mode. In case of an error returns **FALSE**.

LVII. Mimetype Functions

Introduction

The functions in this module try to guess the content type and encoding of a file by looking for certain *magic* byte sequences at specific positions within the file. While this is not a bullet proof approach the heuristics used do a very good job.

This extension is derivated from Apache mod_mime_magic, which is itself based on the `file` command maintaind by Ian F. Darwin. See the source code for further historic and copyright information.

Requirements

No external libraries are needed to build this extension.

Installation

You must compile PHP with the configure switch `--enable-mime-magic` to get support for mime-type functions. The extension needs a copy of the `magic.mime` as distributed with the `file` command. This file also part of most recent Linux distributions and usually stored in the `/usr/share/misc` directory.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. Mimetype configuration options

Name	Default	Changeable
mime_magic.magicfile	"/usr/share/misc/magic.mime"	PHP_INI_SYSTEM

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Table of Contents

mime_content_type

(PHP 4 >= 4.3.0)

mime_content_type -- Detect MIME Content-type for a file

Description

string **mime_content_type** (string filename)

Returns the MIME content type for a file as determined by using information from the `magic.mime` file. Content types are returned in MIME format, like `text/plain` or `application/octet-stream`.

LVIII. Microsoft SQL Server functions

Introduction

These functions allow you to access MS SQL Server database.

Requirements

Requirements for WIn32 platforms.

The extension requires the MS SQL Client Tools to be installed on the system where PHP is installed. The Client Tools can be installed from the MS SQL Server CD or by copying `ntwdblib.dll` from `\winnt\system32` on the server to `\winnt\system32` on the PHP box. Copying `ntwdblib.dll` will only provide access. Configuration of the client will require installation of all the tools.

Requirements for Unix/Linux platforms.

To use the MSSQL extension on Unix/Linux, you first need to build and install the FreeTDS library. Source code and installation instructions are available at the FreeTDS home page: http://www.freetds.org/

Installation

The MSSQL extension is enabled by adding extension=php_mssql.dll to `php.ini`.

To get these functions to work, you have to compile PHP with `--with-mssql[=DIR]`, where DIR is the FreeTDS install prefix.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. MS SQL Server configuration options

Name	Default	Changeable
mssql.allow_persistent	"1"	PHP_INI_SYSTEM
mssql.max_persistent	"-1"	PHP_INI_SYSTEM
mssql.max_links	"-1"	PHP_INI_SYSTEM
mssql.min_error_severity	"10"	PHP_INI_ALL
mssql.min_message_severity	"10"	PHP_INI_ALL
mssql.compatability_mode	"0"	PHP_INI_ALL
mssql.connect_timeout	"5"	PHP_INI_ALL
mssql.timeout	"60"	PHP_INI_ALL
mssql.textsize	"-1"	PHP_INI_ALL
mssql.textlimit	"-1"	PHP_INI_ALL
mssql.batchsize	"0"	PHP_INI_ALL
mssql.datetimeconvert	"1"	PHP_INI_ALL
mssql.secure_connection	"0"	PHP_INI_SYSTEM
mssql.max_procs	"25"	PHP_INI_ALL

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Resource Types

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

MSSQL_ASSOC (integer)

MSSQL_NUM (integer)

MSSQL_BOTH (integer)

SQLTEXT (integer)

SQLVARCHAR (integer)

SQLCHAR (integer)

SQLINT1 (integer)

SQLINT2 (integer)

SQLINT4 (integer)

SQLBIT (integer)

SQLFLT8 (integer)

Table of Contents

mssql_bind

(PHP 4 >= 4.1.0)

mssql_bind -- Adds a parameter to a stored procedure or a remote stored procedure

Description

int **mssql_bind** (int stmt, string param_name, mixed var, int type [, int is_output [, int is_null [, int maxlen]]])

Warning
This function is currently not documented; only the argument list is available.

See also **mssql_execute()**, **mssql_free_statement()** and **mssql_init()**

mssql_close

(PHP 3, PHP 4)

mssql_close -- Close MS SQL Server connection

Description

int **mssql_close** ([int link_identifier])

Returns: **TRUE** on success, **FALSE** on error.

mssql_close() closes the link to a MS SQL Server database that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

Note that this isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

mssql_close() will not close persistent links generated by **mssql_pconnect()**.

See also: **mssql_connect()**, **mssql_pconnect()**.

mssql_connect

(PHP 3, PHP 4)

mssql_connect -- Open MS SQL server connection

Description

int **mssql_connect** ([string servername [, string username [, string password]]])

Returns: A positive MS SQL link identifier on success, or **FALSE** on error.

mssql_connect() establishes a connection to a MS SQL server. The servername argument has to be a valid servername that is defined in the 'interfaces' file.

In case a second call is made to **mssql_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling **mssql_close()**.

See also **mssql_pconnect()**, **mssql_close()**.

mssql_data_seek

(PHP 3, PHP 4)

mssql_data_seek -- Move internal row pointer

Description

int **mssql_data_seek** (int result_identifier, int row_number)

Returns: **TRUE** on success, **FALSE** on failure.

mssql_data_seek() moves the internal row pointer of the MS SQL result associated with the specified result identifier to point to the specified row number. The next call to **mssql_fetch_row()** would return that row.

See also: **mssql_data_seek()**.

mssql_execute

(PHP 4 >= 4.1.0)

mssql_execute -- Executes a stored procedure on a MS SQL server database

Description

int **mssql_execute** (int stmt)

Warning
This function is currently not documented; only the argument list is available.

> **Note:** if the stored procedure returns parameters or a return value these will be available after the call to **mssql_execute()** unless the stored procedure returns more than one result set. In that case use **mssql_next_result()** to shift through the results. When the last result has been processed the output parameters and return values will be available.

See also **mssql_bind()**, **mssql_free_statement()** and **mssql_init()**

mssql_fetch_array

(PHP 3, PHP 4)

mssql_fetch_array -- Fetch row as array

Description

array **mssql_fetch_array** (int result)

Returns: An array that corresponds to the fetched row, or **FALSE** if there are no more rows.

mssql_fetch_array() is an extended version of **mssql_fetch_row()**. In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

An important thing to note is that using **mssql_fetch_array()** is NOT significantly slower than using **mssql_fetch_row()**, while it provides a significant added value.

For further details, also see **mssql_fetch_row()**.

mssql_fetch_assoc

(PHP 4 >= 4.2.0)

mssql_fetch_assoc -- Returns an associative array of the current row in the result set specified by result_id

Description

array **mssql_fetch_assoc** (int result_id [, int result_type])

Warning
This function is currently not documented; only the argument list is available.

mssql_fetch_batch

(PHP 4 >= 4.0.4)

mssql_fetch_batch -- Returns the next batch of records

Description

int **mssql_fetch_batch** (string result_index)

Warning
This function is currently not documented; only the argument list is available.

mssql_fetch_field

(PHP 3, PHP 4)

mssql_fetch_field -- Get field information

Description

object **mssql_fetch_field** (int result [, int field_offset])

Returns an object containing field information.

mssql_fetch_field() can be used in order to obtain information about fields in a certain query result. If the field offset isn't specified, the next field that wasn't yet retrieved by **mssql_fetch_field()** is retrieved.

The properties of the object are:

- name - column name. if the column is a result of a function, this property is set to computed#N, where #N is a serial number.
- column_source - the table from which the column was taken
- max_length - maximum length of the column
- numeric - 1 if the column is numeric

See also **mssql_field_seek()**.

mssql_fetch_object

(PHP 3)

mssql_fetch_object -- Fetch row as object

Description

int **mssql_fetch_object** (int result)

Returns: An object with properties that correspond to the fetched row, or **FALSE** if there are no more rows.

mssql_fetch_object() is similar to **mssql_fetch_array()**, with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

Speed-wise, the function is identical to **mssql_fetch_array()**, and almost as quick as **mssql_fetch_row()** (the difference is insignificant).

See also: **mssql_fetch-array()** and **mssql_fetch-row()**.

mssql_fetch_row

(PHP 3, PHP 4)

mssql_fetch_row -- Get row as enumerated array

Description

array **mssql_fetch_row** (int result)

Returns: An array that corresponds to the fetched row, or **FALSE** if there are no more rows.

mssql_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to **mssql_fetch_rows()** would return the next row in the result set, or **FALSE** if there are no more rows.

See also: **mssql_fetch_array()**, **mssql_fetch_object()**, **mssql_data_seek()**, **mssql_fetch_lengths()**, and **mssql_result()**.

mssql_field_length

(PHP 3>= 3.0.3, PHP 4)

mssql_field_length -- Get the length of a field

Description

int **mssql_field_length** (int result [, int offset])

Warning
This function is currently not documented; only the argument list is available.

mssql_field_name

(PHP 3>= 3.0.3, PHP 4)

mssql_field_name -- Get the name of a field

Description

int **mssql_field_name** (int result [, int offset])

mssql_field_seek

(PHP 3, PHP 4)

mssql_field_seek -- Set field offset

Description

int **mssql_field_seek** (int result, int field_offset)

Seeks to the specified field offset. If the next call to **mssql_fetch_field()** won't include a field offset, this field would be returned.

See also: **mssql_fetch_field()**.

mssql_field_type

(PHP 3>= 3.0.3, PHP 4)

mssql_field_type -- Get the type of a field

Description

string **mssql_field_type** (int result [, int offset])

mssql_free_result

(PHP 3, PHP 4)

mssql_free_result -- Free result memory

Description

int **mssql_free_result** (int result)

mssql_free_result() only needs to be called if you are worried about using too much memory while your script is running. All result memory will automatically be freed when the script ends. You may call **mssql_free_result()** with the result identifier as an argument and the associated result memory will be freed.

mssql_free_statement

(PHP 5 CVS only)

mssql_free_statement -- Free statement memory

Description

int **mssql_free_statement** (int statement)

mssql_free_statement() only needs to be called if you are worried about using too much memory while your script is running. All statement memory will automatically be freed when the script ends. You may call **mssql_free_statement()** with the statement identifier as an argument and the associated statement memory will be freed.

See also **mssql_bind()**, **mssql_execute()** and **mssql_init()**

mssql_get_last_message

(PHP 3, PHP 4)

mssql_get_last_message -- Returns the last message from server (over min_message_severity?)

Description

string **mssql_get_last_message** (void)

mssql_guid_string

(PHP 4 >= 4.1.0)

mssql_guid_string -- Converts a 16 byte binary GUID to a string

Description

string **mssql_guid_string** (string binary [, int short_format])

Warning
This function is currently not documented; only the argument list is available.

mssql_init

(PHP 4 >= 4.1.0)

mssql_init -- Initializes a stored procedure or a remote stored procedure

Description

int **mssql_init** (string sp_name [, int conn_id])

Warning
This function is currently not documented; only the argument list is available.

See also **mssql_bind()**, **mssql_execute()** and **mssql_free_statement()**

mssql_min_error_severity

(PHP 3, PHP 4)

mssql_min_error_severity -- Sets the lower error severity

Description

void **mssql_min_error_severity** (int severity)

mssql_min_message_severity

(PHP 3, PHP 4)

mssql_min_message_severity -- Sets the lower message severity

Description

void **mssql_min_message_severity** (int severity)

mssql_next_result

(PHP 4 >= 4.0.5)

mssql_next_result -- Move the internal result pointer to the next result

Description

bool **mssql_next_result** (int result_id)

When sending more than one SQL statement to the server or executing a stored procedure with multiple results, it will cause the server to return multiple result sets. This function will test for additional results available form the server. If an additional result set exists it will free the existing result set and prepare to fetch the rows from the new result set. The function will return **TRUE** if an additional result set was available or **FALSE** otherwise.

Example 1. mssql_next_result() example

```
<?php
    $link = mssql_connect ("localhost", "userid", "secret");
    mssql_select_db("MyDB", $link);
    $SQL = "Select * from table1 select * from table2";
    $rs = mssql_query($SQL, $link);
    do {
        while ($row = mssql_fetch_row($rs)) {
        }
    } while (mssql_next_result($rs));
    mssql_free_result($rs);
    mssql_close ($link);
?>
```

mssql_num_fields

(PHP 3, PHP 4)

mssql_num_fields -- Get number of fields in result

Description

int **mssql_num_fields** (int result)

mssql_num_fields() returns the number of fields in a result set.

See also: **mssql_db_query()**, mssql_query(), mssql_fetch_field(), and mssql_num_rows().

mssql_num_rows

(PHP 3, PHP 4)

mssql_num_rows -- Get number of rows in result

Description

int **mssql_num_rows** (string result)

mssql_num_rows() returns the number of rows in a result set.

See also: **mssql_db_query()**, mssql_query(), and mssql_fetch_row().

mssql_pconnect

(PHP 3, PHP 4)

mssql_pconnect -- Open persistent MS SQL connection

Description

int **mssql_pconnect** ([string servername [, string username [, string password]]])

Returns: A positive MS SQL persistent link identifier on success, or **FALSE** on error.

mssql_pconnect() acts very much like mssql_connect() with two major differences.

First, when connecting, the function would first try to find a (persistent) link that's already open with the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (mssql_close() will not close links established by **mssql_pconnect()**).

This type of links is therefore called 'persistent'.

mssql_query

(PHP 3, PHP 4)

mssql_query -- Send MS SQL query

Description

int **mssql_query** (string query [, int link_identifier])

Returns: A positive MS SQL result identifier on success, or **FALSE** on error.

mssql_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if mssql_connect() was called, and use it.

See also: **mssql_db_query()**, mssql_select_db(), and mssql_connect().

mssql_result

(PHP 3, PHP 4)

mssql_result -- Get result data

Description

int **mssql_result** (int result, int i, mixed field)

mssql_result() returns the contents of one cell from a MS SQL result set. The field argument can be the field's offset, the field's name or the field's table dot field's name (tablename.fieldname). If the column name has been aliased ('select foo as bar from...'), it uses the alias instead of the column name.

When working on large result sets, you should consider using one of the functions that fetch an entire row (specified below). As these functions return the contents of multiple cells in one function call, they're MUCH quicker than **mssql_result()**. Also, note that specifying a numeric offset for the field argument is much quicker than specifying a fieldname or tablename.fieldname argument.

Recommended high-performance alternatives: **mssql_fetch_row()**, **mssql_fetch_array()**, and **mssql_fetch_object()**.

mssql_rows_affected

(PHP 4 >= 4.0.4)

mssql_rows_affected -- Returns the number of records affected by the query

Description

int **mssql_rows_affected** (int conn_id)

Warning
This function is currently not documented; only the argument list is available.

mssql_select_db

(PHP 3, PHP 4)

mssql_select_db -- Select MS SQL database

Description

int **mssql_select_db** (string database_name [, int link_identifier])

Returns: **TRUE** on success, **FALSE** on error

mssql_select_db() sets the current active database on the server that's associated with the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if **mssql_connect()** was called, and use it.

Every subsequent call to **mssql_query()** will be made on the active database.

See also: **mssql_connect()**, **mssql_pconnect()**, and **mssql_query()**

LIX. Ming functions for Flash

Warning
This extension is *EXPERIMENTAL*. The behaviour of this extension -- including the names of its functions and anything else documented about this extension -- may change without notice in a future release of PHP. Use this extension at your own risk.

Introduction

First of all: Ming is not an acronym. Ming is an open-source (LGPL) library which allows you to create SWF ("Flash") format movies. Ming supports almost all of Flash 4's features, including: shapes, gradients, bitmaps (pngs and jpegs), morphs ("shape tweens"), text, buttons, actions, sprites ("movie clips"), streaming mp3, and color transforms --the only thing that's missing is sound events.

Note that all values specifying length, distance, size, etc. are in "twips", twenty units per pixel. That's pretty much arbitrary, though, since the player scales the movie to whatever pixel size is specified in the embed/object tag, or the entire frame if not embedded.

Ming offers a number of advantages over the existing PHP/libswf module. You can use Ming anywhere you can compile the code, whereas libswf is closed-source and only available for a few platforms, Windows not one of them. Ming provides some insulation from the mundane details of the SWF file format, wrapping the movie elements in PHP objects. Also, Ming is still being maintained; if there's a feature that you want to see, just let us know ming@opaque.net.

Ming was added in PHP 4.0.5.

Requirements

To use Ming with PHP, you first need to build and install the Ming library. Source code and installation instructions are available at the Ming home page: http://ming.sourceforge.net/ along with examples, a small tutorial, and the latest news.

Download the ming archive. Unpack the archive. Go in the Ming directory. make. make install.

This will build `libming.so` and install it into `/usr/lib/`, and copy `ming.h` into `/usr/include/`. Edit the `PREFIX=` line in the `Makefile` to change the installation directory.

Installation

Example 1. built into php (unix)

```
mkdir <phpdir>/ext/ming
cp php_ext/* <phpdir>/ext/ming
cd <phpdir>
./buildconf
./configure --with-ming <other config options>
```

Build and install php as usual, restart web server if necessary.

Now either just add `extension=php_ming.so` to your `php.ini` file, or put `dl('php_ming.so');` at the head of all of your Ming scripts.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

SWFBUTTON_HIT (integer)

SWFBUTTON_DOWN (integer)

SWFBUTTON_OVER (integer)

SWFBUTTON_UP (integer)

SWFBUTTON_MOUSEUPOUTSIDE (integer)

SWFBUTTON_DRAGOVER (integer)

SWFBUTTON_DRAGOUT (integer)

SWFBUTTON_MOUSEUP (integer)

SWFBUTTON_MOUSEDOWN (integer)

SWFBUTTON_MOUSEOUT (integer)

SWFBUTTON_MOUSEOVER (integer)

SWFFILL_RADIAL_GRADIENT (integer)

SWFFILL_LINEAR_GRADIENT (integer)

SWFFILL_TILED_BITMAP (integer)

SWFFILL_CLIPPED_BITMAP (integer)

SWFTEXTFIELD_HASLENGTH (integer)

SWFTEXTFIELD_NOEDIT (integer)

SWFTEXTFIELD_PASSWORD (integer)

SWFTEXTFIELD_MULTILINE (integer)

SWFTEXTFIELD_WORDWRAP (integer)

SWFTEXTFIELD_DRAWBOX (integer)

SWFTEXTFIELD_NOSELECT (integer)

SWFTEXTFIELD_HTML (integer)

SWFTEXTFIELD_ALIGN_LEFT (integer)

SWFTEXTFIELD_ALIGN_RIGHT (integer)

SWFTEXTFIELD_ALIGN_CENTER (integer)

SWFTEXTFIELD_ALIGN_JUSTIFY (integer)

SWFACTION_ONLOAD (integer)

SWFACTION_ENTERFRAME (integer)

SWFACTION_UNLOAD (integer)

SWFACTION_MOUSEMOVE (integer)

SWFACTION_MOUSEDOWN (integer)

SWFACTION_MOUSEUP (integer)

SWFACTION_KEYDOWN (integer)

SWFACTION_KEYUP (integer)

SWFACTION_DATA (integer)

Predefined Classes

The classes below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

Ming introduces 13 new objects in PHP, all with matching methods and attributes. To use them, you need to know about objects.

swfshape

swffill

swfgradient

swfbitmap

swftext

swftextfield

swffont

swfdisplayitem

swfmovie

swfbutton

swfaction

swfmorph

swfsprite

Table of Contents

SWFFill->skewXTo -- Sets fill x-skew
SWFFill->skewYTo -- Sets fill y-skew
SWFFill -- Loads SWFFill object
swffont->getwidth -- Returns the string's width
SWFFont -- Loads a font definition
SWFGradient->addEntry -- Adds an entry to the gradient list.
SWFGradient -- Creates a gradient object
SWFMorph->getshape1 -- Gets a handle to the starting shape
SWFMorph->getshape2 -- Gets a handle to the ending shape
SWFMorph -- Creates a new SWFMorph object.
SWFMovie->add -- Adds any type of data to a movie.
SWFMovie->nextframe -- Moves to the next frame of the animation.
SWFMovie->output -- Dumps your lovingly prepared movie out.
SWFMovie->remove -- Removes the object instance from the display list.
SWFMovie->save -- Saves your movie in a file.
SWFMovie->setbackground -- Sets the background color.
SWFMovie->setdimension -- Sets the movie's width and height.
SWFMovie->setframes -- Sets the total number of frames in the animation.
SWFMovie->setrate -- Sets the animation's frame rate.
SWFMovie->streammp3 -- Streams a MP3 file.
SWFMovie -- Creates a new movie object, representing an SWF version 4 movie.
SWFShape->addFill -- Adds a solid fill to the shape.
SWFShape->drawCurve -- Draws a curve (relative).
SWFShape->drawCurveTo -- Draws a curve.
SWFShape->drawLine -- Draws a line (relative).
SWFShape->drawLineTo -- Draws a line.
SWFShape->movePen -- Moves the shape's pen (relative).
SWFShape->movePenTo -- Moves the shape's pen.
SWFShape->setLeftFill -- Sets left rasterizing color.
SWFShape->setLine -- Sets the shape's line style.
SWFShape->setRightFill -- Sets right rasterizing color.
SWFShape -- Creates a new shape object.
SWFSprite->add -- Adds an object to a sprite
SWFSprite->nextframe -- Moves to the next frame of the animation.
SWFSprite->remove -- Removes an object to a sprite
SWFSprite->setframes -- Sets the total number of frames in the animation.
SWFSprite -- Creates a movie clip (a sprite)
SWFText->addString -- Draws a string
SWFText->getWidth -- Computes string's width
SWFText->moveTo -- Moves the pen
SWFText->setColor -- Sets the current font color
SWFText->setFont -- Sets the current font
SWFText->setHeight -- Sets the current font height
SWFText->setSpacing -- Sets the current font spacing
SWFText -- Creates a new SWFText object.
SWFTextField->addstring -- Concatenates the given string to the text field
SWFTextField->align -- Sets the text field alignment
SWFTextField->setbounds -- Sets the text field width and height
SWFTextField->setcolor -- Sets the color of the text field.
SWFTextField->setFont -- Sets the text field font
SWFTextField->setHeight -- Sets the font height of this text field font.
SWFTextField->setindentation -- Sets the indentation of the first line.
SWFTextField->setLeftMargin -- Sets the left margin width of the text field.
SWFTextField->setLineSpacing -- Sets the line spacing of the text field.
SWFTextField->setMargins -- Sets the margins width of the text field.
SWFTextField->setname -- Sets the variable name
SWFTextField->setrightMargin -- Sets the right margin width of the text field.
SWFTextField -- Creates a text field object

ming_setcubicthreshold

(PHP 4 >= 4.0.5)

ming_setcubicthreshold -- Set cubic threshold (?)

Description

void **ming_setcubicthreshold** (int threshold)

Warning
This function is currently not documented; only the argument list is available.

ming_setscale

(PHP 4 >= 4.0.5)

ming_setscale -- Set scale (?)

Description

void **ming_setscale** (int scale)

Warning
This function is currently not documented; only the argument list is available.

ming_useswfversion

(PHP 4 >= 4.2.0)

ming_useswfversion -- Use SWF version (?)

Description

void **ming_useswfversion** (int version)

Warning
This function is currently not documented; only the argument list is available.

SWFAction

(PHP 4 >= 4.0.5)

SWFAction -- Creates a new Action.

Description

new **swfaction** (string script)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfaction() creates a new Action, and compiles the given script into an SWFAction object.

The script syntax is based on the C language, but with a lot taken out- the SWF bytecode machine is just too simpleminded to do a lot of things we might like. For instance, we can't implement function calls without a tremendous amount of hackery because the jump bytecode has a hardcoded offset value. No pushing your calling address to the stack and returning- every function would have to know exactly where to return to.

So what's left? The compiler recognises the following tokens:

- break
- for
- continue

- if
- else
- do
- while

There is no typed data; all values in the SWF action machine are stored as strings. The following functions can be used in expressions:

time()

Returns the number of milliseconds (?) elapsed since the movie started.

random(seed)

Returns a pseudo-random number in the range o-seed.

length(expr)

Returns the length of the given expression.

int(number)

Returns the given number rounded down to the nearest integer.

concat(expr, expr)

Returns the concatenation of the given expressions.

ord(expr)

Returns the ASCII code for the given character

chr(num)

Returns the character for the given ASCII code

substr(string, location, length)

Returns the substring of length length at location location of the given string string.

Additionally, the following commands may be used:

duplicateClip(clip, name, depth)

Duplicate the named movie clip (aka sprite). The new movie clip has name name and is at depth depth.

removeClip(expr)

Removes the named movie clip.

trace(expr)

Write the given expression to the trace log. Doubtful that the browser plugin does anything with this.

startDrag(target, lock, [left, top, right, bottom])

Start dragging the movie clip target. The lock argument indicates whether to lock the mouse (?)- use 0 (**FALSE**) or 1 (**TRUE**). Optional parameters define a bounding area for the dragging.

stopDrag()

Stop dragging my heart around. And this movie clip, too.

callFrame(expr)

Call the named frame as a function.

getURL(url, target, [method])

Load the given URL into the named target. The target argument corresponds to HTML document targets (such as "_top" or "_blank"). The optional method argument can be POST or GET if you want to submit variables back to the server.

loadMovie(url, target)

Load the given URL into the named target. The target argument can be a frame name (I think), or one of the magical values "_level0" (replaces current movie) or "_level1" (loads new movie on top of current movie).

nextFrame()

Go to the next frame.

prevFrame()

Go to the last (or, rather, previous) frame.

play()

Start playing the movie.

stop()

Stop playing the movie.

toggleQuality()

Toggle between high and low quality.

stopSounds()

Stop playing all sounds.

gotoFrame(num)

Go to frame number num. Frame numbers start at 0.

gotoFrame(name)

Go to the frame named name. Which does a lot of good, since I haven't added frame labels yet.

setTarget(expr)

Sets the context for action. Or so they say- I really have no idea what this does.

And there's one weird extra thing. The expression frameLoaded(num) can be used in if statements and while loops to check if the given frame number has been loaded yet. Well, it's supposed to, anyway, but I've never tested it and I seriously doubt it actually works. You can just use /:framesLoaded instead.

Movie clips (all together now- aka sprites) have properties. You can read all of them (or can you?), you can set some of them, and here they are:

- x
- y
- xScale
- yScale
- currentFrame - (read-only)
- totalFrames - (read-only)
- alpha - transparency level
- visible - 1=on, 0=off (?)
- width - (read-only)
- height - (read-only)
- rotation
- target - (read-only) (???)
- framesLoaded - (read-only)
- name
- dropTarget - (read-only) (???)

- url - (read-only) (???)
- highQuality - 1=high, 0=low (?)
- focusRect - (???)
- soundBufTime - (???)

So, setting a sprite's x position is as simple as `/box.x = 100;`. Why the slash in front of the box, though? That's how flash keeps track of the sprites in the movie, just like a unix filesystem- here it shows that box is at the top level. If the sprite named box had another sprite named biff inside of it, you'd set its x position with /box/biff.x = 100;. At least, I think so; correct me if I'm wrong here.

This simple example will move the red square across the window.

Example 1. swfaction() example

```
<?php
 $s = new SWFShape();
  $f = $s->addFill(0xff, 0, 0);
  $s->setRightFill($f);

  $s->movePenTo(-500,-500);
  $s->drawLineTo(500,-500);
  $s->drawLineTo(500,500);
  $s->drawLineTo(-500,500);
  $s->drawLineTo(-500,-500);

  $p = new SWFSprite();
  $i = $p->add($s);
  $i->setDepth(1);
  $p->nextFrame();

  for($n=0; $n<5; ++$n)
  {
    $i->rotate(-15);
    $p->nextFrame();
  }

  $m = new SWFMovie();
  $m->setBackground(0xff, 0xff, 0xff);
  $m->setDimension(6000,4000);

  $i = $m->add($p);
  $i->setDepth(1);
  $i->moveTo(-500,2000);
  $i->setName("box");

  $m->add(new SWFAction("/box.x += 3;"));
  $m->nextFrame();
  $m->add(new SWFAction("gotoFrame(0); play();"));
  $m->nextFrame();

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

This simple example tracks down your mouse on the screen.

Example 2. swfaction() example

```
<?php

  $m = new SWFMovie();
  $m->setRate(36.0);
  $m->setDimension(1200, 800);
  $m->setBackground(0, 0, 0);

  /* mouse tracking sprite - empty, but follows mouse so we can
     get its x and y coordinates */

  $i = $m->add(new SWFSprite());
  $i->setName('mouse');

  $m->add(new SWFAction("
    startDrag('/mouse', 1); /* '1' means lock sprite to the mouse */
  "));

  /* might as well turn off antialiasing, since these are just squares. */

  $m->add(new SWFAction("
    this.quality = 0;
  "));

  /* morphing box */
  $r = new SWFMorph();
  $s = $r->getShape1();

  /* Note this is backwards from normal shapes.  No idea why. */
```

```
  $s->setLeftFill($s->addFill(0xff, 0xff, 0xff));
  $s->movePenTo(-40, -40);
  $s->drawLine(80, 0);
  $s->drawLine(0, 80);
  $s->drawLine(-80, 0);
  $s->drawLine(0, -80);

  $s = $r->getShape2();

  $s->setLeftFill($s->addFill(0x00, 0x00, 0x00));
  $s->movePenTo(-1, -1);
  $s->drawLine(2, 0);
  $s->drawLine(0, 2);
  $s->drawLine(-2, 0);
  $s->drawLine(0, -2);

  /* sprite container for morphing box -
     this is just a timeline w/ the box morphing */

  $box = new SWFSprite();
  $box->add(new SWFAction("
    stop();
  "));
  $i = $box->add($r);

  for($n=0; $n<=20; ++$n)
  {
    $i->setRatio($n/20);
    $box->nextFrame();
  }

  /* this container sprite allows us to use the same action code many times */

  $cell = new SWFSprite();
  $i = $cell->add($box);
  $i->setName('box');

  $cell->add(new SWFAction("

    setTarget('box');

    /* ...x means the x coordinate of the parent, i.e. (..).x */
    dx = (/mouse.x + random(6)-3 - ...x)/5;
    dy = (/mouse.y + random(6)-3 - ...y)/5;
    gotoFrame(int(dx*dx + dy*dy));

  "));

  $cell->nextFrame();
  $cell->add(new SWFAction("

    gotoFrame(0);
    play();

  "));

  $cell->nextFrame();

  /* finally, add a bunch of the cells to the movie */

  for($x=0; $x<12; ++$x)
  {
    for($y=0; $y<8; ++$y)
    {
      $i = $m->add($cell);
      $i->moveTo(100*$x+50, 100*$y+50);
    }
  }

  $m->nextFrame();

  $m->add(new SWFAction("

    gotoFrame(1);
    play();

  "));

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

Same as above, but with nice colored balls...

Example 3. swfaction() example

```
<?php

  $m = new SWFMovie();
  $m->setDimension(11000, 8000);
  $m->setBackground(0x00, 0x00, 0x00);

  $m->add(new SWFAction("
```

```
this.quality = 0;
/frames.visible = 0;
startDrag('/mouse', 1);

  "));

  // mouse tracking sprite
  $t = new SWFSprite();
  $i = $m->add($t);
  $i->setName('mouse');

  $g = new SWFGradient();
  $g->addEntry(0, 0xff, 0xff, 0xff, 0xff);
  $g->addEntry(0.1, 0xff, 0xff, 0xff, 0xff);
  $g->addEntry(0.5, 0xff, 0xff, 0xff, 0x5f);
  $g->addEntry(1.0, 0xff, 0xff, 0xff, 0);

  // gradient shape thing
  $s = new SWFShape();
  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.03);
  $s->setRightFill($f);
  $s->movePenTo(-600, -600);
  $s->drawLine(1200, 0);
  $s->drawLine(0, 1200);
  $s->drawLine(-1200, 0);
  $s->drawLine(0, -1200);

  // need to make this a sprite so we can multColor it
  $p = new SWFSprite();
  $p->add($s);
  $p->nextFrame();

  // put the shape in here, each frame a different color
  $q = new SWFSprite();
  $q->add(new SWFAction("gotoFrame(random(7)+1); stop();"));
  $i = $q->add($p);

  $i->multColor(1.0, 1.0, 1.0);
  $q->nextFrame();
  $i->multColor(1.0, 0.5, 0.5);
  $q->nextFrame();
  $i->multColor(1.0, 0.75, 0.5);
  $q->nextFrame();
  $i->multColor(1.0, 1.0, 0.5);
  $q->nextFrame();
  $i->multColor(0.5, 1.0, 0.5);
  $q->nextFrame();
  $i->multColor(0.5, 0.5, 1.0);
  $q->nextFrame();
  $i->multColor(1.0, 0.5, 1.0);
  $q->nextFrame();

  // finally, this one contains the action code
  $p = new SWFSprite();
  $i = $p->add($q);
  $i->setName('frames');
  $p->add(new SWFAction("

dx = (/:mousex-/:lastx)/3 + random(10)-5;
dy = (/:mousey-/:lasty)/3;
x = /:mousex;
y = /:mousey;
alpha = 100;

  "));
  $p->nextFrame();

  $p->add(new SWFAction("

this.x = x;
this.y = y;
this.alpha = alpha;
x += dx;
y += dy;
dy += 3;
alpha -= 8;

  "));
  $p->nextFrame();

  $p->add(new SWFAction("prevFrame(); play();"));
  $p->nextFrame();

  $i = $m->add($p);
  $i->setName('frames');
  $m->nextFrame();

  $m->add(new SWFAction("

lastx = mousex;
lasty = mousey;
mousex = /mouse.x;
mousey = /mouse.y;

++num;
```

```
if(num == 11)
  num = 1;

removeClip('char' & num);
duplicateClip(/frames, 'char' & num, num);

  "));

  $m->nextFrame();
  $m->add(new SWFAction("prevFrame(); play();"));

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

This simple example will handles keyboard actions. (You'll probably have to click in the window to give it focus. And you'll probably have to leave your mouse in the frame, too. If you know how to give buttons focus programatically, feel free to share, won't you?)

Example 4. swfaction() example

```
<?php

  /* sprite has one letter per frame */

  $p = new SWFSprite();
  $p->add(new SWFAction("stop();"));

  $chars = "abcdefghijklmnopqrstuvwxyz".
       "ABCDEFGHIJKLMNOPQRSTUVWXYZ".
           "1234567890!@#$%^&/*()_+-=/[]{}|;:,.<>?`~";

  $f = new SWFFont("_sans");

  for($n=0; $nremove($i);
    $t = new SWFTextField();
    $t->setFont($f);
    $t->setHeight(240);
    $t->setBounds(600,240);
    $t->align(SWFTEXTFIELD_ALIGN_CENTER);
    $t->addString($c);
    $i = $p->add($t);
    $p->labelFrame($c);
    $p->nextFrame();
  }

  /* hit region for button - the entire frame */

  $s = new SWFShape();
  $s->setFillStyle0($s->addSolidFill(0, 0, 0, 0));
  $s->drawLine(600, 0);
  $s->drawLine(0, 400);
  $s->drawLine(-600, 0);
  $s->drawLine(0, -400);

  /* button checks for pressed key, sends sprite to the right frame */

  $b = new SWFButton();
  $b->addShape($s, SWFBUTTON_HIT);

  for($n=0; $naddAction(new SWFAction("

setTarget('/char');
gotoFrame('$c');

    "), SWFBUTTON_KEYPRESS($c));
  }

  $m = new SWFMovie();
  $m->setDimension(600,400);
  $i = $m->add($p);
  $i->setName('char');
  $i->moveTo(0,80);

  $m->add($b);

  header('Content-type: application/x-shockwave-flash');
  $m->output();

?>
```

SWFBitmap->getHeight

(no version information, might be only in CVS)

SWFBitmap->getHeight -- Returns the bitmap's height.

Description

int **swfbitmap->getheight** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfbitmap->getheight() returns the bitmap's height in pixels.

See also **swfbitmap->getwidth()**.

SWFBitmap->getWidth

(no version information, might be only in CVS)

SWFBitmap->getWidth -- Returns the bitmap's width.

Description

int **swfbitmap->getwidth** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfbitmap->getwidth() returns the bitmap's width in pixels.

See also **swfbitmap->getheight()**.

SWFBitmap

(PHP 4 >= 4.0.5)

SWFBitmap -- Loads Bitmap object

Description

new **swfbitmap** (string filename [, int alphafilename])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfbitmap() creates a new SWFBitmap object from the Jpeg or DBL file named *filename*. *alphafilename* indicates a MSK file to be used as an alpha mask for a Jpeg image.

> **Note:** We can only deal with baseline (frame 0) jpegs, no baseline optimized or progressive scan jpegs!

SWFBitmap has the following methods : **swfbitmap->getwidth()** and **swfbitmap->getheight()**.

You can't import png images directly, though- have to use the png2dbl utility to make a dbl ("define bits lossless") file from the png. The reason for this is that I don't want a dependency on the png library in ming- autoconf should solve this, but that's not set up yet.

Example 1. Import PNG files

```
<?php
  $s = new SWFShape();
  $f = $s->addFill(new SWFBitmap("png.dbl"));
  $s->setRightFill($f);

  $s->drawLine(32, 0);
  $s->drawLine(0, 32);
```

```
  $s->drawLine(-32, 0);
  $s->drawLine(0, -32);

  $m = new SWFMovie();
  $m->setDimension(32, 32);
  $m->add($s);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

And you can put an alpha mask on a jpeg fill.

Example 2. swfbitmap() example

```
<?php

  $s = new SWFShape();

  // .msk file generated with "gif2mask" utility
  $f = $s->addFill(new SWFBitmap("alphafill.jpg", "alphafill.msk"));
  $s->setRightFill($f);

  $s->drawLine(640, 0);
  $s->drawLine(0, 480);
  $s->drawLine(-640, 0);
  $s->drawLine(0, -480);

  $c = new SWFShape();
  $c->setRightFill($c->addFill(0x99, 0x99, 0x99));
  $c->drawLine(40, 0);
  $c->drawLine(0, 40);
  $c->drawLine(-40, 0);
  $c->drawLine(0, -40);

  $m = new SWFMovie();
  $m->setDimension(640, 480);
  $m->setBackground(0xcc, 0xcc, 0xcc);

  // draw checkerboard background
  for($y=0; $y<480; $y+=40)
  {
    for($x=0; $x<640; $x+=80)
    {
      $i = $m->add($c);
      $i->moveTo($x, $y);
    }

    $y+=40;

    for($x=40; $x<640; $x+=80)
    {
      $i = $m->add($c);
      $i->moveTo($x, $y);
    }
  }

  $m->add($s);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

swfbutton_keypress

(PHP 4 >= 4.0.5)

swfbutton_keypress -- Returns the action flag for keyPress(char)

Description

int **swfbutton_keypress** (string str)

Warning
This function is currently not documented; only the argument list is available.

SWFbutton->addAction

(no version information, might be only in CVS)

SWFbutton->addAction -- Adds an action

Description

void **swfbutton->addaction** (ressource action, int flags)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfbutton->addaction() adds the action *action* to this button for the given conditions. The following *flags* are valid: SWFBUTTON_MOUSEOVER, SWFBUTTON_MOUSEOUT, SWFBUTTON_MOUSEUP, SWFBUTTON_MOUSEUPOUTSIDE, SWFBUTTON_MOUSEDOWN, SWFBUTTON_DRAGOUT and SWFBUTTON_DRAGOVER.

See also **swfbutton->addshape()** and SWFAction().

SWFbutton->addShape

(no version information, might be only in CVS)

SWFbutton->addShape -- Adds a shape to a button

Description

void **swfbutton->addshape** (ressource shape, int flags)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfbutton->addshape() adds the shape *shape* to this button. The following *flags*' values are valid: SWFBUTTON_UP, SWFBUTTON_OVER, SWFBUTTON_DOWN or SWFBUTTON_HIT. SWFBUTTON_HIT isn't ever displayed, it defines the hit region for the button. That is, everywhere the hit shape would be drawn is considered a "touchable" part of the button.

SWFbutton->setAction

(no version information, might be only in CVS)

SWFbutton->setAction -- Sets the action

Description

void **swfbutton->setaction** (ressource action)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfbutton->setaction() sets the action to be performed when the button is clicked. Alias for addAction(shape, SWFBUTTON_MOUSEUP). *action* is a **swfaction()**.

See also **swfbutton->addshape()** and SWFAction().

SWFbutton->setdown

(no version information, might be only in CVS)

SWFbutton->setdown -- Alias for addShape(shape, SWFBUTTON_DOWN))

Description

void **swfbutton->setdown** (ressource shape)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfbutton->setdown() alias for addShape(shape, SWFBUTTON_DOWN).

See also **swfbutton->addshape()** and **SWFAction()**.

SWFbutton->setHit

(no version information, might be only in CVS)

SWFbutton->setHit -- Alias for addShape(shape, SWFBUTTON_HIT)

Description

void **swfbutton->sethit** (ressource shape)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfbutton->sethit() alias for addShape(shape, SWFBUTTON_HIT).

See also **swfbutton->addshape()** and **SWFAction()**.

SWFbutton->setOver

(no version information, might be only in CVS)

SWFbutton->setOver -- Alias for addShape(shape, SWFBUTTON_OVER)

Description

void **swfbutton->setover** (ressource shape)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfbutton->setover() alias for addShape(shape, SWFBUTTON_OVER).

See also **swfbutton->addshape()** and **SWFAction()**.

SWFbutton->setUp

(no version information, might be only in CVS)

SWFbutton->setUp -- Alias for addShape(shape, SWFBUTTON_UP)

Description

void **swfbutton->setup** (ressource shape)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfbutton->setup() alias for addShape(shape, SWFBUTTON_UP).

See also **swfbutton->addshape()** and **SWFAction()**.

SWFbutton

(PHP 4 >= 4.0.5)

SWFbutton -- Creates a new Button.

Description

new **swfbutton** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfbutton() creates a new Button. Roll over it, click it, see it call action code. Swank.

SWFButton has the following methods : **swfbutton->addshape()**, **swfbutton->setup()**, **swfbutton->setover()** **swfbutton->setdown()**, **swfbutton->sethit()** **swfbutton->setaction()** and **swfbutton->addaction()**.

This simple example will show your usual interactions with buttons : rollover, rollon, mouseup, mousedown, noaction.

Example 1. swfbutton() example

```
<?php

  $f = new SWFFont("_serif");

  $p = new SWFSprite();

  function label($string)
  {
    global $f;

    $t = new SWFTextField();
    $t->setFont($f);
    $t->addString($string);
    $t->setHeight(200);
    $t->setBounds(3200,200);
    return $t;
  }
  function addLabel($string)
  {
    global $p;

    $i = $p->add(label($string));
    $p->nextFrame();
    $p->remove($i);
  }

  $p->add(new SWFAction("stop();"));
  addLabel("NO ACTION");
  addLabel("SWFBUTTON_MOUSEUP");
  addLabel("SWFBUTTON_MOUSEDOWN");
  addLabel("SWFBUTTON_MOUSEOVER");
  addLabel("SWFBUTTON_MOUSEOUT");
  addLabel("SWFBUTTON_MOUSEUPOUTSIDE");
  addLabel("SWFBUTTON_DRAGOVER");
  addLabel("SWFBUTTON_DRAGOUT");

  function rect($r, $g, $b)
  {
    $s = new SWFShape();
    $s->setRightFill($s->addFill($r, $g, $b));
    $s->drawLine(600,0);
    $s->drawLine(0,600);
    $s->drawLine(-600,0);
    $s->drawLine(0,-600);

    return $s;
  }

  $b = new SWFButton();
  $b->addShape(rect(0xff, 0, 0), SWFBUTTON_UP | SWFBUTTON_HIT);
  $b->addShape(rect(0, 0xff, 0), SWFBUTTON_OVER);
  $b->addShape(rect(0, 0, 0xff), SWFBUTTON_DOWN);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(1);"),
            SWFBUTTON_MOUSEUP);
```

```
  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(2);"),
        SWFBUTTON_MOUSEDOWN);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(3);"),
        SWFBUTTON_MOUSEOVER);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(4);"),
        SWFBUTTON_MOUSEOUT);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(5);"),
        SWFBUTTON_MOUSEUPOUTSIDE);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(6);"),
        SWFBUTTON_DRAGOVER);

  $b->addAction(new SWFAction("setTarget('/label'); gotoFrame(7);"),
        SWFBUTTON_DRAGOUT);

  $m = new SWFMovie();
  $m->setDimension(4000,3000);

  $i = $m->add($p);
  $i->setName("label");
  $i->moveTo(400,1900);

  $i = $m->add($b);
  $i->moveTo(400,900);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

This simple example will enables you to drag draw a big red button on the windows. No drag-and-drop, just moving around.

Example 2. swfbutton->addaction() example

```
<?php

  $s = new SWFShape();
  $s->setRightFill($s->addFill(0xff, 0, 0));
  $s->drawLine(1000,0);
  $s->drawLine(0,1000);
  $s->drawLine(-1000,0);
  $s->drawLine(0,-1000);

  $b = new SWFButton();
  $b->addShape($s, SWFBUTTON_HIT | SWFBUTTON_UP | SWFBUTTON_DOWN | SWFBUTTON_OVER);

  $b->addAction(new SWFAction("startDrag('/test', 0);"), // '0' means don't lock to mouse
        SWFBUTTON_MOUSEDOWN);

  $b->addAction(new SWFAction("stopDrag();"),
        SWFBUTTON_MOUSEUP | SWFBUTTON_MOUSEUPOUTSIDE);

  $p = new SWFSprite();
  $p->add($b);
  $p->nextFrame();

  $m = new SWFMovie();
  $i = $m->add($p);
  $i->setName('test');
  $i->moveTo(1000,1000);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFDisplayItem->addColor

(no version information, might be only in CVS)

SWFDisplayItem->addColor -- Adds the given color to this item's color transform.

Description

void **swfdisplayitem->addcolor** ([int red [, int green [, int blue [, int a]]]])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfdisplayitem->addcolor() adds the color to this item's color transform. The color is given in its RGB form.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

SWFDisplayItem->move

(no version information, might be only in CVS)

SWFDisplayItem->move -- Moves object in relative coordinates.

Description

void **swfdisplayitem->move** (int dx, int dy)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfdisplayitem->move() moves the current object by (*dx*,*dy*) from its current position.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->moveto()**.

SWFDisplayItem->moveTo

(no version information, might be only in CVS)

SWFDisplayItem->moveTo -- Moves object in global coordinates.

Description

void **swfdisplayitem->moveto** (int x, int y)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfdisplayitem->moveto() moves the current object to (*x*,*y*) in global coordinates.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->move()**.

SWFDisplayItem->multColor

(no version information, might be only in CVS)

SWFDisplayItem->multColor -- Multiplies the item's color transform.

Description

void **swfdisplayitem->multcolor** ([int red [, int green [, int blue [, int a]]]])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfdisplayitem->multcolor() multiplies the item's color transform by the given values.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

This simple example will modify your picture's atmospher to Halloween (use a landscape or bright picture).

Example 1. swfdisplayitem->multcolor() example

```
<?php

  $b = new SWFBitmap("backyard.jpg");
  // note use your own picture :-)
  $s = new SWFShape();
  $s->setRightFill($s->addFill($b));
  $s->drawLine($b->getWidth(), 0);
  $s->drawLine(0, $b->getHeight());
  $s->drawLine(-$b->getWidth(), 0);
  $s->drawLine(0, -$b->getHeight());

  $m = new SWFMovie();
  $m->setDimension($b->getWidth(), $b->getHeight());

  $i = $m->add($s);

  for($n=0; $n<=20; ++$n)
  {
    $i->multColor(1.0-$n/10, 1.0, 1.0);
    $i->addColor(0xff*$n/20, 0, 0);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFDisplayItem->remove

(no version information, might be only in CVS)

SWFDisplayItem->remove -- Removes the object from the movie

Description

void **swfdisplayitem->remove** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfdisplayitem->remove() removes this object from the movie's display list.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfmovie->add()**.

SWFDisplayItem->Rotate

(no version information, might be only in CVS)

SWFDisplayItem->Rotate -- Rotates in relative coordinates.

Description

void **swfdisplayitem->rotate** (float ddegrees)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfdisplayitem->rotate() rotates the current object by *ddegrees* degrees from its current rotation.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->rotateto()**.

SWFDisplayItem->rotateTo

(no version information, might be only in CVS)

SWFDisplayItem->rotateTo -- Rotates the object in global coordinates.

Description

void **swfdisplayitem->rotateto** (float degrees)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfdisplayitem->rotateto() set the current object rotation to *degrees* degrees in global coordinates.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

This example bring three rotating string from the background to the foreground. Pretty nice.

Example 1. swfdisplayitem->rotateto() example

```
<?php
  $thetext =  "ming!";

  $f = new SWFFont("Bauhaus 93.fdb");

  $m = new SWFMovie();
  $m->setRate(24.0);
  $m->setDimension(2400, 1600);
  $m->setBackground(0xff, 0xff, 0xff);

  // functions with huge numbers of arbitrary
  // arguments are always a good idea!  Really!

  function text($r, $g, $b, $a, $rot, $x, $y, $scale, $string)
  {
    global $f, $m;

    $t = new SWFText();
    $t->setFont($f);
    $t->setColor($r, $g, $b, $a);
    $t->setHeight(960);
    $t->moveTo(-($f->getWidth($string))/2, $f->getAscent()/2);
    $t->addString($string);

    // we can add properties just like a normal php var,
    // as long as the names aren't already used.
    // e.g., we can't set $i->scale, because that's a function

    $i = $m->add($t);
    $i->x = $x;
    $i->y = $y;
    $i->rot = $rot;
    $i->s = $scale;
    $i->rotateTo($rot);
    $i->scale($scale, $scale);

    // but the changes are local to the function, so we have to
    // return the changed object.  kinda weird..

    return $i;
 }

  function step($i)
  {
    $oldrot = $i->rot;
    $i->rot = 19*$i->rot/20;
    $i->x = (19*$i->x + 1200)/20;
    $i->y = (19*$i->y + 800)/20;
    $i->s = (19*$i->s + 1.0)/20;

    $i->rotateTo($i->rot);
    $i->scaleTo($i->s, $i->s);
    $i->moveTo($i->x, $i->y);
```

```
    return $i;
  }

  // see?  it sure paid off in legibility:

  $i1 = text(0xff, 0x33, 0x33, 0xff, 900, 1200, 800, 0.03, $thetext);
  $i2 = text(0x00, 0x33, 0xff, 0x7f, -560, 1200, 800, 0.04, $thetext);
  $i3 = text(0xff, 0xff, 0xff, 0x9f, 180, 1200, 800, 0.001, $thetext);

  for($i=1; $i<=100; ++$i)
  {
    $i1 = step($i1);
    $i2 = step($i2);
    $i3 = step($i3);

    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

See also **swfdisplayitem->rotate()**.

SWFDisplayItem->scale

(no version information, might be only in CVS)

SWFDisplayItem->scale -- Scales the object in relative coordinates.

Description

void **swfdisplayitem->scale** (int dx, int dy)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfdisplayitem->scale() scales the current object by (*dx*,*dy*) from its current size.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->scaleto()**.

SWFDisplayItem->scaleTo

(no version information, might be only in CVS)

SWFDisplayItem->scaleTo -- Scales the object in global coordinates.

Description

void **swfdisplayitem->scaleto** (int x, int y)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfdisplayitem->scaleto() scales the current object to (*x*,*y*) in global coordinates.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->scale()**.

SWFDisplayItem->setDepth

(no version information, might be only in CVS)

SWFDisplayItem->setDepth -- Sets z-order

Description

void **swfdisplayitem->setdepth** (float depth)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfdisplayitem->rotate() sets the object's z-order to *depth*. Depth defaults to the order in which instances are created (by add'ing a shape/text to a movie)- newer ones are on top of older ones. If two objects are given the same depth, only the later-defined one can be moved.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

SWFDisplayItem->setName

(no version information, might be only in CVS)

SWFDisplayItem->setName -- Sets the object's name

Description

void **swfdisplayitem->setname** (string name)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfdisplayitem->setname() sets the object's name to *name*, for targetting with action script. Only useful on sprites.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

SWFDisplayItem->setRatio

(no version information, might be only in CVS)

SWFDisplayItem->setRatio -- Sets the object's ratio.

Description

void **swfdisplayitem->setratio** (float ratio)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfdisplayitem->setratio() sets the object's ratio to *ratio*. Obviously only useful for morphs.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

This simple example will morph nicely three concentric circles.

Example 1. swfdisplayitem->setname() example

```
<?php
```

```
  $p = new SWFMorph();

  $g = new SWFGradient();
  $g->addEntry(0.0, 0, 0, 0);
  $g->addEntry(0.16, 0xff, 0xff, 0xff);
  $g->addEntry(0.32, 0, 0, 0);
  $g->addEntry(0.48, 0xff, 0xff, 0xff);
  $g->addEntry(0.64, 0, 0, 0);
  $g->addEntry(0.80, 0xff, 0xff, 0xff);
  $g->addEntry(1.00, 0, 0, 0);

  $s = $p->getShape1();
  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.05);
  $s->setLeftFill($f);
  $s->movePenTo(-160, -120);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $g = new SWFGradient();
  $g->addEntry(0.0, 0, 0, 0);
  $g->addEntry(0.16, 0xff, 0, 0);
  $g->addEntry(0.32, 0, 0, 0);
  $g->addEntry(0.48, 0, 0xff, 0);
  $g->addEntry(0.64, 0, 0, 0);
  $g->addEntry(0.80, 0, 0, 0xff);
  $g->addEntry(1.00, 0, 0, 0);

  $s = $p->getShape2();
  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.05);
  $f->skewXTo(1.0);
  $s->setLeftFill($f);
  $s->movePenTo(-160, -120);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $m = new SWFMovie();
  $m->setDimension(320, 240);
  $i = $m->add($p);
  $i->moveTo(160, 120);

  for($n=0; $n<=1.001; $n+=0.01)
  {
    $i->setRatio($n);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFDisplayItem->skewX

(no version information, might be only in CVS)

SWFDisplayItem->skewX -- Sets the X-skew.

Description

void **swfdisplayitem->skewx** (float ddegrees)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfdisplayitem->skewx() adds *ddegrees* to current x-skew.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->skewx()**, **swfdisplayitem->skewy()** and **swfdisplayitem->skewyto()**.

SWFDisplayItem->skewXTo

(no version information, might be only in CVS)

SWFDisplayItem->skewXTo -- Sets the X-skew.

Description

void **swfdisplayitem->skewxto** (float degrees)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfdisplayitem->skewxto() sets the x-skew to `degrees`. For `degrees` is 1.0, it means a 45-degree forward slant. More is more forward, less is more backward.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->skewx()**, **swfdisplayitem->skewy()** and **swfdisplayitem->skewyto()**.

SWFDisplayItem->skewY

(no version information, might be only in CVS)

SWFDisplayItem->skewY -- Sets the Y-skew.

Description

void **swfdisplayitem->skewy** (float ddegrees)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfdisplayitem->skewy() adds `ddegrees` to current y-skew.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->skewyto()**, **swfdisplayitem->skewx()** and **swfdisplayitem->skewxto()**.

SWFDisplayItem->skewYTo

(no version information, might be only in CVS)

SWFDisplayItem->skewYTo -- Sets the Y-skew.

Description

void **swfdisplayitem->skewyto** (float degrees)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfdisplayitem->skewyto() sets the y-skew to `degrees`. For `degrees` is 1.0, it means a 45-degree forward slant. More is more upward, less is more downward.

The object may be a **swfshape()**, a **swfbutton()**, a **swftext()** or a **swfsprite()** object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->skewy()**, **swfdisplayitem->skewx()** and **swfdisplayitem->skewxto()**.

SWFDisplayItem

(no version information, might be only in CVS)

SWFDisplayItem -- Creates a new displayitem object.

Description

new **swfdisplayitem** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfdisplayitem() creates a new swfdisplayitem object.

Here's where all the animation takes place. After you define a shape, a text object, a sprite, or a button, you add it to the movie, then use the returned handle to move, rotate, scale, or skew the thing.

SWFDisplayItem has the following methods : **swfdisplayitem->move()**, **swfdisplayitem->moveto()**, **swfdisplayitem->scaleto()**, **swfdisplayitem->scale()**, **swfdisplayitem->rotate()**, **swfdisplayitem->rotateto()**, **swfdisplayitem->skewxto()**, **swfdisplayitem->skewx()**, **swfdisplayitem->skewyto() swfdisplayitem->skewyto()**, **swfdisplayitem->setdepth() swfdisplayitem->remove()**, **swfdisplayitem->setname() swfdisplayitem->setratio()**, **swfdisplayitem->addcolor()** and **swfdisplayitem->multcolor()**.

SWFFill->moveTo

(no version information, might be only in CVS)

SWFFill->moveTo -- Moves fill origin

Description

void **swffill->moveto** (int x, int y)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swffill->moveto() moves fill's origin to (*x*,*y*) in global coordinates.

SWFFill->rotateTo

(no version information, might be only in CVS)

SWFFill->rotateTo -- Sets fill's rotation

Description

void **swffill->rotateto** (float degrees)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swffill->rotateto() sets fill's rotation to *degrees* degrees.

SWFFill->scaleTo

(no version information, might be only in CVS)

SWFFill->scaleTo -- Sets fill's scale

Description

void **swffill->scaleto** (int x, int y)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swffill->scaleto() sets fill's scale to *x* in the x-direction, *y* in the y-direction.

SWFFill->skewXTo

(no version information, might be only in CVS)

SWFFill->skewXTo -- Sets fill x-skew

Description

void **swffill->skewxto** (float x)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swffill->skewxto() sets fill x-skew to *x*. For *x* is 1.0, it is a is a 45-degree forward slant. More is more forward, less is more backward.

SWFFill->skewYTo

(no version information, might be only in CVS)

SWFFill->skewYTo -- Sets fill y-skew

Description

void **swffill->skewyto** (float y)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swffill->skewyto() sets fill y-skew to *y*. For *y* is 1.0, it is a is a 45-degree upward slant. More is more upward, less is more downward.

SWFFill

SWFFill -- Loads SWFFill object

Description

The **swffill()** object allows you to transform (scale, skew, rotate) bitmap and gradient fills. **swffill()** objects are created by the **swfshape->addfill()** methods.

SWFFill has the following methods : **swffill->moveto()** and **swffill->scaleto()**, **swffill->rotateto()**, **swffill->skewxto()** and **swffill->skewyto()**.

swffont->getwidth

(no version information, might be only in CVS)

swffont->getwidth -- Returns the string's width

Description

int **swffont->getwidth** (string string)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swffont->getwidth() returns the string *string*'s width, using font's default scaling. You'll probably want to use the **SWFText()** version of this method which uses the text object's scale.

SWFFont

(PHP 4 >= 4.0.5)

SWFFont -- Loads a font definition

Description

new **swffont** (string filename)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

If *filename* is the name of an FDB file (i.e., it ends in ".fdb"), load the font definition found in said file. Otherwise, create a browser-defined font reference.

FDB ("font definition block") is a very simple wrapper for the SWF DefineFont2 block which contains a full description of a font. One may create FDB files from SWT Generator template files with the included makefdb utility- look in the util directory off the main ming distribution directory.

Browser-defined fonts don't contain any information about the font other than its name. It is assumed that the font definition will be provided by the movie player. The fonts _serif, _sans, and _typewriter should always be available. For example:

```
<?php
$f = newSWFFont("_sans");
?>
```

will give you the standard sans-serif font, probably the same as what you'd get with `<font name="sans-serif">` in HTML.

swffont() returns a reference to the font definition, for use in the **SWFText->setFont()** and the **SWFTextField->setFont()** methods.

SWFFont has the following methods : **swffont->getwidth()**.

SWFGradient->addEntry

(no version information, might be only in CVS)

SWFGradient->addEntry -- Adds an entry to the gradient list.

Description

void **swfgradient->addentry** (float ratio, int red, int green, int blue [, int a])

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfgradient->addentry() adds an entry to the gradient list. *ratio* is a number between 0 and 1 indicating where in the gradient this color appears. Thou shalt add entries in order of increasing ratio.

red, *green*, *blue* is a color (RGB mode). Last parameter *a* is optional.

SWFGradient

(PHP 4 >= 4.0.5)

SWFGradient -- Creates a gradient object

Description

new **swfgradient** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfgradient() creates a new SWFGradient object.

After you've added the entries to your gradient, you can use the gradient in a shape fill with the **swfshape->addfill()** method.

SWFGradient has the following methods : **swfgradient->addentry()**.

This simple example will draw a big black-to-white gradient as background, and a redish disc in its center.

Example 1. swfgradient() example

```
<?php

  $m = new SWFMovie();
  $m->setDimension(320, 240);

  $s = new SWFShape();

  // first gradient- black to white
  $g = new SWFGradient();
  $g->addEntry(0.0, 0, 0, 0);
  $g->addEntry(1.0, 0xff, 0xff, 0xff);

  $f = $s->addFill($g, SWFFILL_LINEAR_GRADIENT);
  $f->scaleTo(0.01);
  $f->moveTo(160, 120);
  $s->setRightFill($f);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $m->add($s);

  $s = new SWFShape();

  // second gradient- radial gradient from red to transparent
  $g = new SWFGradient();
  $g->addEntry(0.0, 0xff, 0, 0, 0xff);
  $g->addEntry(1.0, 0xff, 0, 0, 0);

  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.005);
  $f->moveTo(160, 120);
  $s->setRightFill($f);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $m->add($s);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFMorph->getshape1

(no version information, might be only in CVS)

SWFMorph->getshape1 -- Gets a handle to the starting shape

Description

mixed **swfmorph->getshape1** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfmorph->getshape1() gets a handle to the morph's starting shape. **swfmorph->getshape1()** returns an **swfshape()** object.

SWFMorph->getshape2

(no version information, might be only in CVS)

SWFMorph->getshape2 -- Gets a handle to the ending shape

Description

mixed **swfmorph->getshape2** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfmorph->getshape2() gets a handle to the morph's ending shape. **swfmorph->getshape2()** returns an **swfshape()** object.

SWFMorph

(PHP 4 >= 4.0.5)

SWFMorph -- Creates a new SWFMorph object.

Description

new **swfmorph** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfmorph() creates a new SWFMorph object.

Also called a "shape tween". This thing lets you make those tacky twisting things that make your computer choke. Oh, joy!

The methods here are sort of weird. It would make more sense to just have newSWFMorph(shape1, shape2);, but as things are now, shape2 needs to know that it's the second part of a morph. (This, because it starts writing its output as soon as it gets drawing commands- if it kept its own description of its shapes and wrote on completion this and some other things would be much easier.)

SWFMorph has the following methods : **swfmorph->getshape1()** and **swfmorph->getshape1()**.

This simple example will morph a big red square into a smaller blue black-bordered square.

Example 1. swfmorph() example

```
<?php
  $p = new SWFMorph();

  $s = $p->getShape1();
```

```
  $s->setLine(0,0,0,0);

  /* Note that this is backwards from normal shapes (left instead of right).
     I have no idea why, but this seems to work.. */

  $s->setLeftFill($s->addFill(0xff, 0, 0));
  $s->movePenTo(-1000,-1000);
  $s->drawLine(2000,0);
  $s->drawLine(0,2000);
  $s->drawLine(-2000,0);
  $s->drawLine(0,-2000);

  $s = $p->getShape2();
  $s->setLine(60,0,0,0);
  $s->setLeftFill($s->addFill(0, 0, 0xff));
  $s->movePenTo(0,-1000);
  $s->drawLine(1000,1000);
  $s->drawLine(-1000,1000);
  $s->drawLine(-1000,-1000);
  $s->drawLine(1000,-1000);

  $m = new SWFMovie();
  $m->setDimension(3000,2000);
  $m->setBackground(0xff, 0xff, 0xff);

  $i = $m->add($p);
  $i->moveTo(1500,1000);

  for($r=0.0; $r<=1.0; $r+=0.1)
  {
    $i->setRatio($r);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFMovie->add

(no version information, might be only in CVS)

SWFMovie->add -- Adds any type of data to a movie.

Description

void **swfmovie->add** (ressource instance)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfmovie->add() adds *instance* to the current movie. *instance* is any type of data : Shapes, text, fonts, etc. must all be add'ed to the movie to make this work.

For displayable types (shape, text, button, sprite), this returns an **SWFDisplayItem()**, a handle to the object in a display list. Thus, you can add the same shape to a movie multiple times and get separate handles back for each separate instance.

See also all other objects (adding this later), and **swfmovie->remove()**

See examples in : **swfdisplayitem->rotateto()** and **swfshape->addfill()**.

SWFMovie->nextframe

(no version information, might be only in CVS)

SWFMovie->nextframe -- Moves to the next frame of the animation.

Description

void **swfmovie->nextframe** (void)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfmovie->nextframe() moves to the next frame of the animation.

SWFMovie->output

(no version information, might be only in CVS)

SWFMovie->output -- Dumps your lovingly prepared movie out.

Description

void **swfmovie->output** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfmovie->output() dumps your lovingly prepared movie out. In PHP, preceding this with the command

```
<?php
header('Content-type: application/x-shockwave-flash');
?>
```

convinces the browser to display this as a flash movie.

See also **swfmovie->save()**.

See examples in : **swfmovie->streammp3()**, **swfdisplayitem->rotateto()**, **swfaction()**... Any example will use this method.

SWFMovie->remove

(no version information, might be only in CVS)

SWFMovie->remove -- Removes the object instance from the display list.

Description

void **swfmovie->remove** (resource instance)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfmovie->remove() removes the object instance *instance* from the display list.

See also **swfmovie->add()**.

SWFMovie->save

(no version information, might be only in CVS)

SWFMovie->save -- Saves your movie in a file.

Description

void **swfmovie->save** (string filename)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfmovie->save() saves your movie to the file named *filename*.

See also **output()**.

SWFMovie->setbackground

(no version information, might be only in CVS)

SWFMovie->setbackground -- Sets the background color.

Description

void **swfmovie->setbackground** (int red, int green, int blue)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfmovie->setbackground() sets the background color. Why is there no rgba version? Think about it. (Actually, that's not such a dumb question after all- you might want to let the html background show through. There's a way to do that, but it only works on IE4. Search the http://www.macromedia.com/ site for details.)

SWFMovie->setdimension

(no version information, might be only in CVS)

SWFMovie->setdimension -- Sets the movie's width and height.

Description

void **swfmovie->setdimension** (int width, int height)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfmovie->setdimension() sets the movie's width to *width* and height to *height*.

SWFMovie->setframes

(no version information, might be only in CVS)

SWFMovie->setframes -- Sets the total number of frames in the animation.

Description

void **swfmovie->setframes** (string numberofframes)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfmovie->setframes() sets the total number of frames in the animation to *numberofframes*.

SWFMovie->setrate

(no version information, might be only in CVS)

SWFMovie->setrate -- Sets the animation's frame rate.

Description

void **swfmovie->setrate** (int rate)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfmovie->setrate() sets the frame rate to *rate*, in frame per seconds. Animation will slow down if the player can't render frames fast enough- unless there's a streaming sound, in which case display frames are sacrificed to keep sound from skipping.

SWFMovie->streammp3

(no version information, might be only in CVS)

SWFMovie->streammp3 -- Streams a MP3 file.

Description

void **swfmovie->streammp3** (string mp3FileName)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfmovie->streammp3() streams the mp3 file *mp3FileName*. Not very robust in dealing with oddities (can skip over an initial ID3 tag, but that's about it). Like **SWFShape->addJpegFill()**, this isn't a stable function- we'll probably need to make a separate SWFSound object to contain sound types.

Note that the movie isn't smart enough to put enough frames in to contain the entire mp3 stream- you'll have to add (length of song * frames per second) frames to get the entire stream in.

Yes, now you can use ming to put that rock and roll devil worship music into your SWF files. Just don't tell the RIAA.

Example 1. swfmovie->streammp3() example

```
<?php
  $m = new SWFMovie();
  $m->setRate(12.0);
  $m->streamMp3("distortobass.mp3");
  // use your own MP3

  // 11.85 seconds at 12.0 fps = 142 frames
  $m->setFrames(142);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFMovie

(PHP 4 >= 4.0.5)

SWFMovie -- Creates a new movie object, representing an SWF version 4 movie.

Description

new **swfmovie** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfmovie() creates a new movie object, representing an SWF version 4 movie.

SWFMovie has the following methods : **swfmovie->output(),swfmovie->save()**, **swfmovie->add()**, **swfmovie->remove()**, **swfmovie->nextframe()**, **swfmovie->setbackground()**, **swfmovie->setrate()**, **swfmovie->setdimension()**, **swfmovie->setframes()** and **swfmovie->streammp3()**.

See examples in : **swfdisplayitem->rotateto()**, **swfshape->setline()**, **swfshape->addfill()**... Any example will use this object.

SWFShape->addFill

(no version information, might be only in CVS)

SWFShape->addFill -- Adds a solid fill to the shape.

Description

void **swfshape->addfill** (int red, int green, int blue [, int a])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

void **swfshape->addfill** (SWFbitmap bitmap [, int flags])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

void **swfshape->addfill** (SWFGradient gradient [, int flags])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfshape->addfill() adds a solid fill to the shape's list of fill styles. **swfshape->addfill()** accepts three different types of arguments.

red, *green*, *blue* is a color (RGB mode). Last parameter *a* is optional.

The *bitmap* argument is an **swfbitmap()** object. The *flags* argument can be one of the following values : SWFFILL_CLIPPED_BITMAP or SWFFILL_TILED_BITMAP. Default is SWFFILL_TILED_BITMAP. I think.

The *gradient* argument is an **swfgradient()** object. The flags argument can be one of the following values : SWFFILL_RADIAL_GRADIENT or SWFFILL_LINEAR_GRADIENT. Default is SWFFILL_LINEAR_GRADIENT. I'm sure about this one. Really.

swfshape->addfill() returns an **swffill()** object for use with the **swfshape->setleftfill()** and **swfshape->setrightfill()** functions described below.

See also **swfshape->setleftfill()** and **swfshape->setrightfill()**.

This simple example will draw a frame on a bitmap. Ah, here's another buglet in the flash player- it doesn't seem to care about the second shape's bitmap's transformation in a morph. According to spec, the bitmap should stretch along with the shape in this example..

Example 1. swfshape->addfill() example

```
<?php

   $p = new SWFMorph();

   $b = new SWFBitmap("alphafill.jpg");
   // use your own bitmap
   $width = $b->getWidth();
   $height = $b->getHeight();

   $s = $p->getShape1();
   $f = $s->addFill($b, SWFFILL_TILED_BITMAP);
   $f->moveTo(-$width/2, -$height/4);
   $f->scaleTo(1.0, 0.5);
   $s->setLeftFill($f);
   $s->movePenTo(-$width/2, -$height/4);
```

```
  $s->drawLine($width, 0);
  $s->drawLine(0, $height/2);
  $s->drawLine(-$width, 0);
  $s->drawLine(0, -$height/2);

  $s = $p->getShape2();
  $f = $s->addFill($b, SWFFILL_TILED_BITMAP);

  // these two have no effect!
  $f->moveTo(-$width/4, -$height/2);
  $f->scaleTo(0.5, 1.0);

  $s->setLeftFill($f);
  $s->movePenTo(-$width/4, -$height/2);
  $s->drawLine($width/2, 0);
  $s->drawLine(0, $height);
  $s->drawLine(-$width/2, 0);
  $s->drawLine(0, -$height);

  $m = new SWFMovie();
  $m->setDimension($width, $height);
  $i = $m->add($p);
  $i->moveTo($width/2, $height/2);

  for($n=0; $n<1.001; $n+=0.03)
  {
    $i->setRatio($n);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFShape->drawCurve

(no version information, might be only in CVS)

SWFShape->drawCurve -- Draws a curve (relative).

Description

void **swfshape->drawcurve** (int controldx, int controldy, int anchordx, int anchordy)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfshape->drawcurve() draws a quadratic curve (using the current line style,set by **swfshape->setline()**) from the current pen position to the relative position (*anchorx*,*anchory*) using relative control point (*controlx*,*controly*). That is, head towards the control point, then smoothly turn to the anchor point.

See also **swfshape->drawlineto()**, **swfshape->drawline()**, **swfshape->movepento()** and **swfshape->movepen()**.

SWFShape->drawCurveTo

(no version information, might be only in CVS)

SWFShape->drawCurveTo -- Draws a curve.

Description

void **swfshape->drawcurveto** (int controlx, int controly, int anchorx, int anchory)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfshape->drawcurveto() draws a quadratic curve (using the current line style, set by **swfshape->setline()**) from the current pen position to (*anchorx*,*anchory*) using (*controlx*,*controly*) as a control point. That is, head towards the control point, then smoothly turn to the anchor point.

See also **swfshape->drawlineto()**, **swfshape->drawline()**, **swfshape->movepento()** and **swfshape->movepen()**.

SWFShape->drawLine

(no version information, might be only in CVS)

SWFShape->drawLine -- Draws a line (relative).

Description

void **swfshape->drawline** (int dx, int dy)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfshape->drawline() draws a line (using the current line style set by **swfshape->setline()**) from the current pen position to displacement (*dx*,*dy*).

See also **swfshape->movepento()**, **swfshape->drawcurveto()**, **swfshape->movepen()** and **swfshape->drawlineto()**.

SWFShape->drawLineTo

(no version information, might be only in CVS)

SWFShape->drawLineTo -- Draws a line.

Description

void **swfshape->drawlineto** (int x, int y)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfshape->setrightfill() draws a line (using the current line style, set by **swfshape->setline()**) from the current pen position to point (*x*,*y*) in the shape's coordinate space.

See also **swfshape->movepento()**, **swfshape->drawcurveto()**, **swfshape->movepen()** and **swfshape->drawline()**.

SWFShape->movePen

(no version information, might be only in CVS)

SWFShape->movePen -- Moves the shape's pen (relative).

Description

void **swfshape->movepen** (int dx, int dy)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfshape->setrightfill() move the shape's pen from coordinates (current x,current y) to (current x + *dx*, current y + *dy*) in the shape's coordinate space.

See also **swfshape->movepento()**, **swfshape->drawcurveto()**, **swfshape->drawlineto()** and **swfshape->drawline()**.

SWFShape->movePenTo

(no version information, might be only in CVS)

SWFShape->movePenTo -- Moves the shape's pen.

Description

void **swfshape->movepento** (int x, int y)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfshape->setrightfill() move the shape's pen to (*x*,*y*) in the shape's coordinate space.

See also **swfshape->movepen()**, **swfshape->drawcurveto()**, **swfshape->drawlineto()** and **swfshape->drawline()**.

SWFShape->setLeftFill

(no version information, might be only in CVS)

SWFShape->setLeftFill -- Sets left rasterizing color.

Description

void **swfshape->setleftfill** (swfgradient fill)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

void **swfshape->setleftfill** (int red, int green, int blue [, int a])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

What this nonsense is about is, every edge segment borders at most two fills. When rasterizing the object, it's pretty handy to know what those fills are ahead of time, so the swf format requires these to be specified.

swfshape->setleftfill() sets the fill on the left side of the edge- that is, on the interior if you're defining the outline of the shape in a counter-clockwise fashion. The fill object is an SWFFill object returned from one of the addFill functions above.

This seems to be reversed when you're defining a shape in a morph, though. If your browser crashes, just try setting the fill on the other side.

Shortcut for `swfshape->setleftfill($s->addfill($r, $g, $b [, $a]));`.

See also **swfshape->setrightfill()**.

SWFShape->setLine

(no version information, might be only in CVS)

SWFShape->setLine -- Sets the shape's line style.

Description

void **swfshape->setline** (int width [, int red [, int green [, int blue [, int a]]]])

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfshape->setline() sets the shape's line style. *width* is the line's width. If *width* is 0, the line's style is removed (then, all other arguments are ignored). If *width* > 0, then line's color is set to *red*, *green*, *blue*. Last parameter *a* is optional.

swfshape->setline() accepts 1, 4 or 5 arguments (not 3 or 2).

You must declare all line styles before you use them (see example).

This simple example will draw a big "!#%*@", in funny colors and gracious style.

Example 1. swfshape->setline() example

```
<?php
  $s = new SWFShape();
  $f1 = $s->addFill(0xff, 0, 0);
  $f2 = $s->addFill(0xff, 0x7f, 0);
  $f3 = $s->addFill(0xff, 0xff, 0);
  $f4 = $s->addFill(0, 0xff, 0);
  $f5 = $s->addFill(0, 0, 0xff);

  // bug: have to declare all line styles before you use them
  $s->setLine(40, 0x7f, 0, 0);
  $s->setLine(40, 0x7f, 0x3f, 0);
  $s->setLine(40, 0x7f, 0x7f, 0);
  $s->setLine(40, 0, 0x7f, 0);
  $s->setLine(40, 0, 0, 0x7f);

  $f = new SWFFont('Techno.fdb');

  $s->setRightFill($f1);
  $s->setLine(40, 0x7f, 0, 0);
  $s->drawGlyph($f, '!');
  $s->movePen($f->getWidth('!'), 0);

  $s->setRightFill($f2);
  $s->setLine(40, 0x7f, 0x3f, 0);
  $s->drawGlyph($f, '#');
  $s->movePen($f->getWidth('#'), 0);

  $s->setRightFill($f3);
  $s->setLine(40, 0x7f, 0x7f, 0);
  $s->drawGlyph($f, '%');
  $s->movePen($f->getWidth('%'), 0);

  $s->setRightFill($f4);
  $s->setLine(40, 0, 0x7f, 0);
  $s->drawGlyph($f, '*');
  $s->movePen($f->getWidth('*'), 0);

  $s->setRightFill($f5);
  $s->setLine(40, 0, 0, 0x7f);
  $s->drawGlyph($f, '@');

  $m = new SWFMovie();
  $m->setDimension(3000,2000);
  $m->setRate(12.0);
  $i = $m->add($s);
  $i->moveTo(1500-$f->getWidth("!#%*@")/2, 1000+$f->getAscent()/2);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFShape->setRightFill

(no version information, might be only in CVS)

SWFShape->setRightFill -- Sets right rasterizing color.

Description

void **swfshape->setrightfill** (swfgradient fill)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

void **swfshape->setrightfill** (int red, int green, int blue [, int a])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

See also **swfshape->setleftfill()**.

Shortcut for `swfshape->setrightfill($s->addfill($r, $g, $b [, $a]));`.

SWFShape

(PHP 4 >= 4.0.5)

SWFShape -- Creates a new shape object.

Description

new **swfshape** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfshape() creates a new shape object.

SWFShape has the following methods : **swfshape->setline()**, **swfshape->addfill()**, **swfshape->setleftfill()**, **swfshape->setrightfill()**, **swfshape->movepento()**, **swfshape->movepen()**, **swfshape->drawlineto()**, **swfshape->drawline()**, **swfshape->drawcurveto()** and **swfshape->drawcurve()**.

This simple example will draw a big red elliptic quadrant.

Example 1. swfshape() example

```
<?php
  $s = new SWFShape();
  $s->setLine(40, 0x7f, 0, 0);
  $s->setRightFill($s->addFill(0xff, 0, 0));
  $s->movePenTo(200, 200);
  $s->drawLineTo(6200, 200);
  $s->drawLineTo(6200, 4600);
  $s->drawCurveTo(200, 4600, 200, 200);

  $m = new SWFMovie();
  $m->setDimension(6400, 4800);
  $m->setRate(12.0);
  $m->add($s);
  $m->nextFrame();

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFSprite->add

(no version information, might be only in CVS)

SWFSprite->add -- Adds an object to a sprite

Description

void **swfsprite->add** (resource object)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfsprite->add() adds a **swfshape()**, a **swfbutton()**, a **swftext()**, a **swfaction()** or a **swfsprite()** object.

For displayable types (**swfshape()**, **swfbutton()**, **swftext()**, **swfaction()** or **swfsprite()**), this returns a handle to the object in a

display list.

SWFSprite->nextframe

(no version information, might be only in CVS)

SWFSprite->nextframe -- Moves to the next frame of the animation.

Description

void **swfsprite->nextframe** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfsprite->setframes() moves to the next frame of the animation.

SWFSprite->remove

(no version information, might be only in CVS)

SWFSprite->remove -- Removes an object to a sprite

Description

void **swfsprite->remove** (ressource object)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfsprite->remove() remove a **swfshape()**, a **swfbutton()**, a **swftext()**, a **swfaction()** or a **swfsprite()** object from the sprite.

SWFSprite->setframes

(no version information, might be only in CVS)

SWFSprite->setframes -- Sets the total number of frames in the animation.

Description

void **swfsprite->setframes** (int numberofframes)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfsprite->setframes() sets the total number of frames in the animation to `numberofframes`.

SWFSprite

(PHP 4 >= 4.0.5)

SWFSprite -- Creates a movie clip (a sprite)

Description

new **swfsprite** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swfsprite() are also known as a "movie clip", this allows one to create objects which are animated in their own timelines. Hence, the sprite has most of the same methods as the movie.

swfsprite() has the following methods : **swfsprite->add()**, **swfsprite->remove()**, **swfsprite->nextframe()** and **swfsprite->setframes()**.

This simple example will spin gracefully a big red square.

Example 1. swfsprite() example

```
<?php
  $s = new SWFShape();
  $s->setRightFill($s->addFill(0xff, 0, 0));
  $s->movePenTo(-500,-500);
  $s->drawLineTo(500,-500);
  $s->drawLineTo(500,500);
  $s->drawLineTo(-500,500);
  $s->drawLineTo(-500,-500);

  $p = new SWFSprite();
  $i = $p->add($s);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();

  $m = new SWFMovie();
  $i = $m->add($p);
  $i->moveTo(1500,1000);
  $i->setName("blah");

  $m->setBackground(0xff, 0xff, 0xff);
  $m->setDimension(3000,2000);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFText->addString

(no version information, might be only in CVS)

SWFText->addString -- Draws a string

Description

void **swftext->addstring** (string string)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swftext->addstring() draws the string *string* at the current pen (cursor) location. Pen is at the baseline of the text; i.e., ascending text is in the -y direction.

SWFText->getWidth

(no version information, might be only in CVS)

SWFText->getWidth -- Computes string's width

Description

void **swftext->addstring** (string string)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swftext->addstring() returns the rendered width of the *string* string at the text object's current font, scale, and spacing settings.

SWFText->moveTo

(no version information, might be only in CVS)

SWFText->moveTo -- Moves the pen

Description

void **swftext->moveto** (int x, int y)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swftext->moveto() moves the pen (or cursor, if that makes more sense) to (*x*,*y*) in text object's coordinate space. If either is zero, though, value in that dimension stays the same. Annoying, should be fixed.

SWFText->setColor

(no version information, might be only in CVS)

SWFText->setColor -- Sets the current font color

Description

void **swftext->setcolor** (int red, int green, int blue [, int a])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swftext->setspacing() changes the current text color. Default is black. I think. Color is represented using the RGB system.

SWFText->setFont

(no version information, might be only in CVS)

SWFText->setFont -- Sets the current font

Description

void **swftext->setfont** (string font)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swftext->setfont() sets the current font to *font*.

SWFText->setHeight

(no version information, might be only in CVS)

SWFText->setHeight -- Sets the current font height

Description

void **swftext->setheight** (int height)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swftext->setheight() sets the current font height to *height*. Default is 240.

SWFText->setSpacing

(no version information, might be only in CVS)

SWFText->setSpacing -- Sets the current font spacing

Description

void **swftext->setspacing** (float spacing)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swftext->setspacing() sets the current font spacing to *spacingspacing*. Default is 1.0. 0 is all of the letters written at the same point. This doesn't really work that well because it inflates the advance across the letter, doesn't add the same amount of spacing between the letters. I should try and explain that better, prolly. Or just fix the damn thing to do constant spacing. This was really just a way to figure out how letter advances work, anyway.. So nyah.

SWFText

(PHP 4 >= 4.0.5)

SWFText -- Creates a new SWFText object.

Description

new **swftext** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swftext() creates a new SWFText object, fresh for manipulating.

SWFText has the following methods : **swftext->setfont()**, **swftext->setheight()**, **swftext->setspacing()**, **swftext->setcolor()**, **swftext->moveto()**, **swftext->addstring()** and **swftext->getwidth()**.

This simple example will draw a big yellow "PHP generates Flash with Ming" text, on white background.

Example 1. swftext() example

```
<?php
  $f = new SWFFont("Techno.fdb");
  $t = new SWFText();
  $t->setFont($f);
  $t->moveTo(200, 2400);
  $t->setColor(0xff, 0xff, 0);
  $t->setHeight(1200);
  $t->addString("PHP generates Flash with Ming!!");

  $m = new SWFMovie();
  $m->setDimension(5400, 3600);

  $m->add($t);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFTextField->addstring

(no version information, might be only in CVS)

SWFTextField->addstring -- Concatenates the given string to the text field

Description

void **swftextfield->addstring** (string string)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swftextfield->setname() concatenates the string *`string`* to the text field.

SWFTextField->align

(no version information, might be only in CVS)

SWFTextField->align -- Sets the text field alignment

Description

void **swftextfield->align** (int alignement)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swftextfield->align() sets the text field alignment to *`alignement`*. Valid values for *`alignement`* are : SWFTEXTFIELD_ALIGN_LEFT, SWFTEXTFIELD_ALIGN_RIGHT, SWFTEXTFIELD_ALIGN_CENTER and SWFTEXTFIELD_ALIGN_JUSTIFY.

SWFTextField->setbounds

(no version information, might be only in CVS)

SWFTextField->setbounds -- Sets the text field width and height

Description

void **swftextfield->setbounds** (int width, int height)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swftextfield->setbounds() sets the text field width to *width* and height to *height*. If you don't set the bounds yourself, Ming makes a poor guess at what the bounds are.

SWFTextField->setcolor

(no version information, might be only in CVS)

SWFTextField->setcolor -- Sets the color of the text field.

Description

void **swftextfield->setcolor** (int red, int green, int blue [, int a])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swftextfield->setcolor() sets the color of the text field. Default is fully opaque black. Color is represented using RGB system.

SWFTextField->setFont

(no version information, might be only in CVS)

SWFTextField->setFont -- Sets the text field font

Description

void **swftextfield->setfont** (string font)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swftextfield->setfont() sets the text field font to the [browser-defined?] *font* font.

SWFTextField->setHeight

(no version information, might be only in CVS)

SWFTextField->setHeight -- Sets the font height of this text field font.

Description

void **swftextfield->setheight** (int height)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swftextfield->setheight() sets the font height of this text field font to the given height *height*. Default is 240.

SWFTextField->setindentation

(no version information, might be only in CVS)

SWFTextField->setindentation -- Sets the indentation of the first line.

Description

void **swftextfield->setindentation** (int width)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swftextfield->setindentation() sets the indentation of the first line in the text field, to *width*.

SWFTextField->setLeftMargin

(no version information, might be only in CVS)

SWFTextField->setLeftMargin -- Sets the left margin width of the text field.

Description

void **swftextfield->setleftmargin** (int width)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swftextfield->setleftmargin() sets the left margin width of the text field to *width*. Default is 0.

SWFTextField->setLineSpacing

(no version information, might be only in CVS)

SWFTextField->setLineSpacing -- Sets the line spacing of the text field.

Description

void **swftextfield->setlinespacing** (int height)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swftextfield->setlinespacing() sets the line spacing of the text field to the height of *height*. Default is 40.

SWFTextField->setMargins

(no version information, might be only in CVS)

SWFTextField->setMargins -- Sets the margins width of the text field.

Description

void **swftextfield->setmargins** (int left, int right)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swftextfield->setmargins() set both margins at once, for the man on the go.

SWFTextField->setname

(no version information, might be only in CVS)

SWFTextField->setname -- Sets the variable name

Description

void **swftextfield->setname** (string name)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swftextfield->setname() sets the variable name of this text field to *name*, for form posting and action scripting purposes.

SWFTextField->setrightMargin

(no version information, might be only in CVS)

SWFTextField->setrightMargin -- Sets the right margin width of the text field.

Description

void **swftextfield->setrightmargin** (int width)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swftextfield->setrightmargin() sets the right margin width of the text field to *width*. Default is 0.

SWFTextField

(PHP 4 >= 4.0.5)

SWFTextField -- Creates a text field object

Description

new **swftextfield** ([int flags])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

swftextfield() creates a new text field object. Text Fields are less flexible than **swftext()** objects- they can't be rotated, scaled non-proportionally, or skewed, but they can be used as form entries, and they can use browser-defined fonts.

The optional flags change the text field's behavior. It has the following possibles values :

- SWFTEXTFIELD_DRAWBOX draws the outline of the textfield
- SWFTEXTFIELD_HASLENGTH
- SWFTEXTFIELD_HTML allows text markup using HTML-tags
- SWFTEXTFIELD_MULTILINE allows multiple lines
- SWFTEXTFIELD_NOEDIT indicates that the field shouldn't be user-editable
- SWFTEXTFIELD_NOSELECT makes the field non-selectable
- SWFTEXTFIELD_PASSWORD obscures the data entry

- SWFTEXTFIELD_WORDWRAP allows text to wrap

Flags are combined with the bitwise OR operation. For example,

```
<?php
$t = newSWFTextField(SWFTEXTFIELD_PASSWORD | SWFTEXTFIELD_NOEDIT);
?>
```

creates a totally useless non-editable password field.

SWFTextField has the following methods : **swftextfield->setfont()**, **swftextfield->setbounds()**, **swftextfield->align()**, **swftextfield->setheight()**, **swftextfield->setleftmargin()**, **swftextfield->setrightmargin()**, **swftextfield->setmargins()**, **swftextfield->setindentation()**, **swftextfield->setlinespacing()**, **swftextfield->setcolor()**, **swftextfield->setname()** and **swftextfield->addstring()**.

LX. Miscellaneous functions

Introduction

These functions were placed here because none of the other categories seemed to fit.

Requirements

No external libraries are needed to build this extension.

Installation

There is no installation needed to use these functions; they are part of the PHP core.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. Misc. Configuration Options

Name	Default	Changeable
ignore_user_abort	"0"	PHP_INI_ALL
highlight.string	#CC0000	PHP_INI_ALL
highlight.comment	#FF9900	PHP_INI_ALL
highlight.keyword	#006600	PHP_INI_ALL
highlight.bg	#FFFFFF	PHP_INI_ALL
highlight.default	#0000CC	PHP_INI_ALL
highlight.html	#000000	PHP_INI_ALL
browscap	NULL	PHP_INI_SYSTEM

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Here is a short explanation of the configuration directives.

`ignore_user_abort` **boolean**

TRUE by default. If changed to **FALSE** scripts will be terminated as soon as they try to output something after a client has aborted their connection.

See also **ignore_user_abort()**.

`highlight.xxx` **string**

Colors for Syntax Highlighting mode. Anything that's acceptable in <font color="??????"> would work.

browscap **string**

Name (e.g.: `browscap.ini`)and location of browser capabilities file. See also **get_browser()**.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Table of Contents

connection_aborted

(PHP 3>= 3.0.7, PHP 4)

connection_aborted -- Returns **TRUE** if client disconnected

Description

int **connection_aborted** (void)

Returns **TRUE** if client disconnected. See the Connection Handling description in the Features chapter for a complete explanation.

connection_status

(PHP 3>= 3.0.7, PHP 4)

connection_status -- Returns connection status bitfield

Description

int **connection_status** (void)

Returns the connection status bitfield. See the Connection Handling description in the Features chapter for a complete explanation.

connection_timeout

(PHP 3>= 3.0.7, PHP 4 <= 4.0.4)

connection_timeout -- Return **TRUE** if script timed out

Description

bool **connection_timeout** (void)

Returns **TRUE** if script timed out.

Deprecated
This function is deprecated, and doesn't even exist anymore as of 4.0.5.

See the Connection Handling description in the Features chapter for a complete explanation.

constant

(PHP 4 >= 4.0.4)

constant -- Returns the value of a constant

Description

mixed **constant** (string name)

constant() will return the value of the constant indicated by *name*.

constant() is useful if you need to retrieve the value of a constant, but do not know it's name. i.e. It is stored in a variable or returned by a function.

Example 1. constant() example

```
<?php

define ("MAXSIZE", 100);

echo MAXSIZE;
echo constant("MAXSIZE"); // same thing as the previous line

?>
```

See also **define()**, **defined()** and the section on Constants.

define

(PHP 3, PHP 4)

define -- Defines a named constant.

Description

bool **define** (string name, mixed value [, bool case_insensitive])

Defines a named constant. See the section on constants for more details.

The name of the constant is given by *name*; the value is given by *value*.

The optional third parameter *case_insensitive* is also available. If the value **TRUE** is given, then the constant will be defined case-insensitive. The default behaviour is case-sensitive; i.e. CONSTANT and Constant represent different values.

Example 1. Defining Constants

```
<?php
define ("CONSTANT", "Hello world.");
echo CONSTANT; // outputs "Hello world."
echo Constant; // outputs "Constant" and issues a notice.

define ("GREETING", "Hello you.",TRUE);
echo GREETING; // outputs "Hello you."
echo Greeting; // outputs "Hello you."

?>
```

Returns **TRUE** on success or **FALSE** on failure.

See also **defined()**, **constant()** and the section on Constants.

defined

(PHP 3, PHP 4)

defined -- Checks whether a given named constant exists

Description

bool **defined** (string name)

Returns **TRUE** if the named constant given by *name* has been defined, **FALSE** otherwise.

Example 1. Checking Constants

```
<?php
if (defined("CONSTANT")){ // Note that it should be quoted
    echo CONSTANT; //
    }
?>
```

See also **define()**, **constant()**, **get_defined_constants()** and the section on Constants.

die

die -- Alias of **exit()**

Description

This function is an alias of **exit()**.

eval

(PHP 3, PHP 4)

eval -- Evaluate a string as PHP code

Description

mixed **eval** (string code_str)

eval() evaluates the string given in *code_str* as PHP code. Among other things, this can be useful for storing code in a database text field for later execution.

There are some factors to keep in mind when using **eval()**. Remember that the string passed must be valid PHP code, including things like terminating statements with a semicolon so the parser doesn't die on the line after the **eval()**, and properly escaping things in *code_str*.

Also remember that variables given values under **eval()** will retain these values in the main script afterwards.

A `return` statement will terminate the evaluation of the string immediately. In PHP 4, **eval()** returns **NULL** unless **return()** is called in the evaluated code, in which case the value passed to **return()** is returned. In PHP 3, **eval()** does not return a value.

Example 1. eval() example - simple text merge

```
<?php
$string = 'cup';
$name = 'coffee';
$str = 'This is a $string with my $name in it.<br>';
echo $str;
eval ("\$str = \"$str\";");
echo $str;
?>
```

The above example will show:

```
This is a $string with my $name in it.
This is a cup with my coffee in it.
```

> **Tip:** As with anything that outputs its result directly to the browser, you can use the output-control functions to capture the output of this function, and save it in a **string** (for example).

exit

(PHP 3, PHP 4)

exit -- Output a message and terminate the current script

Description

void **exit** ([string status])

void **exit** (int status)

> **Note:** This is not a real function, but a language construct.

The **exit()** function terminates execution of the script. It prints *status* just before exiting.

If *status* is an **integer**, that value will also be used as the exit status. Exit statuses should be in the range 1 to 254, the exit status 255 is reserved by PHP and shall not be used.

> **Note:** The current CVS version does NOT print the *status* if it is an **integer**.

> **Note:** The **die()** function is an alias for **exit()**.

Example 1. exit() example

```
<?php

$filename = '/path/to/data-file';
$file = fopen ($filename, 'r')
    or exit("unable to open file ($filename)");

?>
```

get_browser

(PHP 3, PHP 4)

get_browser -- Tells what the user's browser is capable of

Description

object **get_browser** ([string user_agent])

get_browser() attempts to determine the capabilities of the user's browser. This is done by looking up the browser's information in the `browscap.ini` file. By default, the value of `HTTP_USER_AGENT` is used; however, you can alter this (i.e., look up another browser's info) by passing the optional *user_agent* parameter to **get_browser()**.

The information is returned in an **object**, which will contain various data elements representing, for instance, the browser's major and minor version numbers and ID string; **TRUE/FALSE** values for features such as frames, JavaScript, and cookies; and so

forth.

While `browscap.ini` contains information on many browsers, it relies on user updates to keep the database current. The format of the file is fairly self-explanatory.

The following example shows how one might list all available information retrieved about the user's browser.

Example 1. get_browser() example

```
<?php
echo $_SERVER['HTTP_USER_AGENT'] . "<hr />\n";

$browser = get_browser();

foreach ($browser as $name => $value) {
    print "<b>$name</b> $value <br />\n";
}

?>
```

The output of the above script would look something like this:

```
Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586)<hr />
<b>browser_name_pattern:</b> Mozilla/4\.5.*<br />
<b>parent:</b> Netscape 4.0<br />
<b>platform:</b> Linux<br />
<b>majorver:</b> 4<br />
<b>minorver:</b> 5<br />
<b>browser:</b> Netscape<br />
<b>version:</b> 4<br />
<b>frames:</b> 1<br />
<b>tables:</b> 1<br />
<b>cookies:</b> 1<br />
<b>backgroundsounds:</b> <br />
<b>vbscript:</b> <br />
<b>javascript:</b> 1<br />
<b>javaapplets:</b> 1<br />
<b>activexcontrols:</b> <br />
<b>beta:</b> <br />
<b>crawler:</b> <br />
<b>authenticodeupdate:</b> <br />
<b>msn:</b> <br />
```

In order for this to work, your browscap configuration setting in `php.ini` must point to the correct location of the `browscap.ini` file on your system. `browscap.ini` is not bundled with PHP but you may find an up-to-date browscap.ini file here. By default, the browscap directive is commented out.

> **Note:** The `cookies` value simply means that the browser itself is capable of accepting cookies and does not mean the user has enabled the browser to accept cookies or not. The only way to test if cookies are accepted is to set one with **setcookie()**, reload, and check for the value.

> **Note:** On versions older than PHP 4.0.6, you will have to pass the user agent in via the optional *`user_agent`* parameter if the PHP directive register_globals is `off`. In this case, you will pass in `$HTTP_SERVER_VARS['HTTP_USER_AGENT']`.

highlight_file

(PHP 4)

highlight_file -- Syntax highlighting of a file

Description

mixed **highlight_file** (string filename [, bool return])

The **highlight_file()** function prints out a syntax higlighted version of the code contained in *`filename`* using the colors defined in the built-in syntax highlighter for PHP.

If the second parameter *`return`* is set to **TRUE** then **highlight_file()** will return the highlighted code as a string instead of printing it out. If the second parameter is not set to **TRUE** then **highlight_file()** will return **TRUE** on success, **FALSE** on failure.

> **Note:** The *`return`* parameter became available in PHP 4.2.0. Before this time it behaved like the default, which is **FALSE**

> **Note:** Care should be taken when using the **show_source()** and **highlight_file()** functions to make sure that you do not inadvertently reveal sensitive information such as passwords or any other type of information that might create a potential security risk.

> **Note:** Since PHP 4.2.1 this function is also affected by safe_mode and open_basedir.

Example 1. Creating a source highlighting URL

To setup a URL that can code hightlight any script that you pass to it, we will make use of the "ForceType" directive in Apache to generate a nice URL pattern, and use the function **highlight_file()** to show a nice looking code list.

In your httpd.conf you can add the following:

```
<Location /source>
    ForceType application/x-httpd-php
</Location>
```

And then make a file named "source" and put it in your web root directory.

```
<HTML>
<HEAD>
<TITLE>Source Display</TITLE>
</HEAD>
<BODY BGCOLOR="white">
<?php
    $script = getenv ("PATH_TRANSLATED");
    if(!$script) {
    echo "<BR><B>ERROR: Script Name needed</B><BR>";
    } else {
    if (ereg("(\.php|\.inc)$",$script)) {
    echo "<H1>Source of: $PATH_INFO</H1>\n<HR>\n";
    highlight_file($script);
    } else {
    echo "<H1>ERROR: Only PHP or include script names are allowed</H1>";
    }
    }
    echo "<HR>Processed: ".date("Y/M/d H:i:s",time());
?>
</BODY>
</HTML>
```

Then you can use an URL like the one below to display a colorized version of a script located in "/path/to/script.php" in your web site.

```
http://your.server.com/source/path/to/script.php
```

See also highlight_string(), show_source().

highlight_string

(PHP 4)

highlight_string -- Syntax highlighting of a string

Description

mixed **highlight_string** (string str [, bool return])

The **highlight_string()** function outputs a syntax highlighted version of *str* using the colors defined in the built-in syntax highlighter for PHP.

If the second parameter *return* is set to **TRUE** then **highlight_string()** will return the highlighted code as a string instead of printing it out. If the second parameter is not set to **TRUE** then **highlight_string()** will return **TRUE** on success, **FALSE** on failure.

> **Note:** The *return* parameter became available in PHP 4.2.0. Before this time it behaved like the default, which is **FALSE**

See also highlight_file(), and show_source().

ignore_user_abort

(PHP 3>= 3.0.7, PHP 4)

ignore_user_abort -- Set whether a client disconnect should abort script execution

Description

int **ignore_user_abort** ([int setting])

This function sets whether a client disconnect should cause a script to be aborted. It will return the previous setting and can be called without an argument to not change the current setting and only return the current setting. See the Connection Handling section in the Features chapter for a complete description of connection handling in PHP.

pack

(PHP 3, PHP 4)

pack -- Pack data into binary string.

Description

string **pack** (string format [, mixed args])

Pack given arguments into binary string according to *format*. Returns binary string containing data.

The idea to this function was taken from Perl and all formatting codes work the same as there, however, there are some formatting codes that are missing such as Perl's "u" format code. The format string consists of format codes followed by an optional repeater argument. The repeater argument can be either an integer value or * for repeating to the end of the input data. For a, A, h, H the repeat count specifies how many characters of one data argument are taken, for @ it is the absolute position where to put the next data, for everything else the repeat count specifies how many data arguments are consumed and packed into the resulting binary string. Currently implemented are

- a NUL-padded string
- A SPACE-padded string
- h Hex string, low nibble first
- H Hex string, high nibble first
- c signed char
- C unsigned char
- s signed short (always 16 bit, machine byte order)
- S unsigned short (always 16 bit, machine byte order)
- n unsigned short (always 16 bit, big endian byte order)
- v unsigned short (always 16 bit, little endian byte order)
- i signed integer (machine dependent size and byte order)
- I unsigned integer (machine dependent size and byte order)
- l signed long (always 32 bit, machine byte order)
- L unsigned long (always 32 bit, machine byte order)
- N unsigned long (always 32 bit, big endian byte order)
- V unsigned long (always 32 bit, little endian byte order)
- f float (machine dependent size and representation)
- d double (machine dependent size and representation)
- x NUL byte
- X Back up one byte
- @ NUL-fill to absolute position

Example 1. pack() format string

```
$binarydata = pack ("nvc*", 0x1234, 0x5678, 65, 66);
```

The resulting binary string will be 6 bytes long and contain the byte sequence 0x12, 0x34, 0x78, 0x56, 0x41, 0x42.

Note that the distinction between signed and unsigned values only affects the function **unpack()**, where as function **pack()** gives the same result for signed and unsigned format codes.

Also note that PHP internally stores **integer** values as signed values of a machine dependent size. If you give it an unsigned integer value too large to be stored that way it is converted to a **float** which often yields an undesired result.

show_source

(PHP 4)

show_source -- Syntax highlighting of a file

Description

bool **show_source** (string filename [, bool return])

This function is an alias to **highlight_file()**. For more information see the documentation there.

See also **highlight_string()** and **highlight_file()**.

sleep

(PHP 3, PHP 4)

sleep -- Delay execution

Description

void **sleep** (int seconds)

The **sleep()** function delays program execution for the given number of *seconds*.

See also **usleep()** and **set_time_limit()**

uniqid

(PHP 3, PHP 4)

uniqid -- Generate a unique ID

Description

string **uniqid** (string prefix [, bool lcg])

uniqid() returns a prefixed unique identifier based on the current time in microseconds. The prefix can be useful for instance if you generate identifiers simultaneously on several hosts that might happen to generate the identifier at the same microsecond. *Prefix* can be up to 114 characters long.

If the optional *lcg* parameter is **TRUE**, **uniqid()** will add additional "combined LCG" entropy at the end of the return value, which should make the results more unique.

With an empty *prefix*, the returned string will be 13 characters long. If *lcg* is **TRUE**, it will be 23 characters.

> **Note:** The *lcg* parameter is only available in PHP 4 and PHP 3.0.13 and later.

If you need a unique identifier or token and you intend to give out that token to the user via the network (i.e. session cookies), it is recommended that you use something along the lines of

```
$token = md5(uniqid("")); // no prefix
$better_token = md5(uniqid(rand(),1)); // better, difficult to guess
```

This will create a 32 character identifier (a 128 bit hex number) that is extremely difficult to predict.

unpack

(PHP 3, PHP 4)

unpack -- Unpack data from binary string

Description

array **unpack** (string format, string data)

unpack() from binary string into array according to *format*. Returns array containing unpacked elements of binary string.

unpack() works slightly different from Perl as the unpacked data is stored in an associative array. To accomplish this you have to name the different format codes and separate them by a slash /.

Example 1. unpack() format string

```
$array = unpack ("c2chars/nint", $binarydata);
```

The resulting array will contain the entries "chars1", "chars2" and "int".

For an explanation of the format codes see also: **pack()**

Note that PHP internally stores integral values as signed. If you unpack a large unsigned long and it is of the same size as PHP internally stored values the result will be a negative number even though unsigned unpacking was specified.

usleep

(PHP 3, PHP 4)

usleep -- Delay execution in microseconds

Description

void **usleep** (int micro_seconds)

The **usleep()** function delays program execution for the given number of *micro_seconds*. A microsecond is one millionth of a second.

See also **sleep()** and **set_time_limit()**.

> **Note:** This function does not work on Windows systems.

LXI. mnoGoSearch Functions

Introduction

These functions allow you to access the mnoGoSearch (former UdmSearch) free search engine. mnoGoSearch is a full-featured search engine software for intranet and internet servers, distributed under the GNU license. mnoGoSearch has a number of unique features, which makes it appropriate for a wide range of applications from search within your site to a specialized search system such as cooking recipes or newspaper search, FTP archive search, news articles search, etc. It offers full-text indexing and searching for HTML, PDF, and text documents. mnoGoSearch consists of two parts. The first is an indexing mechanism (indexer). The purpose of the indexer is to walk through HTTP, FTP, NEWS servers or local files, recursively grabbing all the documents and storing meta-data about that documents in a SQL database in a smart and effective manner. After every document is referenced by its corresponding URL, meta-data is collected by the indexer for later use in a search process. The search is performed via Web interface. C, CGI, PHP and Perl search front ends are included.

More information about mnoGoSearch can be found at http://www.mnogosearch.ru/.

> **Note:** This extension is not available on Windows platforms.

Requirements

Download mnoGosearch from http://www.mnogosearch.ru/ and install it on your system. You need at least version 3.1.10 of mnoGoSearch installed to use these functions.

Installation

In order to have these functions available, you must compile PHP with mnoGosearch support by using the `--with-mnogosearch`option. If you use this option without specifying the path to mnoGosearch, PHP will look for mnoGosearch under `/usr/local/mnogosearch` path by default. If you installed mnoGosearch at a different location you should specify it: `--with-mnogosearch=DIR`.

> **Note:** PHP contains built-in MySQL access library, which can be used to access MySQL. It is known that mnoGoSearch is not compatible with this built-in library and can work only with generic MySQL libraries. Thus, if you use mnoGoSearch with MySQL, during PHP configuration you have to indicate the directory of your MySQL installation, that was used during mnoGoSearch configuration, i.e. for example: `--with-mnogosearch --with-mysql=/usr`.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

`UDM_FIELD_URLID` (integer)

`UDM_FIELD_URL` (integer)

`UDM_FIELD_CONTENT` (integer)

`UDM_FIELD_TITLE` (integer)

`UDM_FIELD_KEYWORDS` (integer)

`UDM_FIELD_DESC` (integer)

`UDM_FIELD_DESCRIPTION` (integer)

`UDM_FIELD_TEXT` (integer)

`UDM_FIELD_SIZE` (integer)

`UDM_FIELD_RATING` (integer)

`UDM_FIELD_SCORE` (integer)

`UDM_FIELD_MODIFIED` (integer)

`UDM_FIELD_ORDER` (integer)

`UDM_FIELD_CRC` (integer)

UDM_FIELD_CATEGORY (integer)

UDM_FIELD_LANG (integer)

UDM_FIELD_CHARSET (integer)

UDM_PARAM_PAGE_SIZE (integer)

UDM_PARAM_PAGE_NUM (integer)

UDM_PARAM_SEARCH_MODE (integer)

UDM_PARAM_CACHE_MODE (integer)

UDM_PARAM_TRACK_MODE (integer)

UDM_PARAM_PHRASE_MODE (integer)

UDM_PARAM_CHARSET (integer)

UDM_PARAM_LOCAL_CHARSET (integer)

UDM_PARAM_BROWSER_CHARSET (integer)

UDM_PARAM_STOPTABLE (integer)

UDM_PARAM_STOP_TABLE (integer)

UDM_PARAM_STOPFILE (integer)

UDM_PARAM_STOP_FILE (integer)

UDM_PARAM_WEIGHT_FACTOR (integer)

UDM_PARAM_WORD_MATCH (integer)

UDM_PARAM_MAX_WORD_LEN (integer)

UDM_PARAM_MAX_WORDLEN (integer)

UDM_PARAM_MIN_WORD_LEN (integer)

UDM_PARAM_MIN_WORDLEN (integer)

UDM_PARAM_ISPELL_PREFIXES (integer)

UDM_PARAM_ISPELL_PREFIX (integer)

UDM_PARAM_PREFIXES (integer)

UDM_PARAM_PREFIX (integer)

UDM_PARAM_CROSS_WORDS (integer)

UDM_PARAM_CROSSWORDS (integer)

UDM_PARAM_VARDIR (integer)

UDM_PARAM_DATADIR (integer)

UDM_PARAM_HLBEG (integer)

UDM_PARAM_HLEND (integer)

UDM_PARAM_SYNONYM (integer)

UDM_PARAM_SEARCHD (integer)

UDM_PARAM_QSTRING (integer)

UDM_PARAM_REMOTE_ADDR (integer)

UDM_LIMIT_CAT (integer)

UDM_LIMIT_URL (integer)

UDM_LIMIT_TAG (integer)

UDM_LIMIT_LANG (integer)

UDM_LIMIT_DATE (integer)

UDM_PARAM_FOUND (integer)

UDM_PARAM_NUM_ROWS (integer)

UDM_PARAM_WORDINFO (integer)

UDM_PARAM_WORD_INFO (integer)

UDM_PARAM_SEARCHTIME (integer)

UDM_PARAM_SEARCH_TIME (integer)

UDM_PARAM_FIRST_DOC (integer)

UDM_PARAM_LAST_DOC (integer)

UDM_MODE_ALL (integer)

UDM_MODE_ANY (integer)

UDM_MODE_BOOL (integer)

UDM_MODE_PHRASE (integer)

UDM_CACHE_ENABLED (integer)

UDM_CACHE_DISABLED (integer)

UDM_TRACK_ENABLED (integer)

UDM_TRACK_DISABLED (integer)

UDM_PHRASE_ENABLED (integer)

UDM_PHRASE_DISABLED (integer)

UDM_CROSS_WORDS_ENABLED (integer)

UDM_CROSSWORDS_ENABLED (integer)

UDM_CROSS_WORDS_DISABLED (integer)

UDM_CROSSWORDS_DISABLED (integer)

UDM_PREFIXES_ENABLED (integer)

UDM_PREFIX_ENABLED (integer)

UDM_ISPELL_PREFIXES_ENABLED (integer)

UDM_ISPELL_PREFIX_ENABLED (integer)

UDM_PREFIXES_DISABLED (integer)

UDM_PREFIX_DISABLED (integer)

UDM_ISPELL_PREFIXES_DISABLED (integer)

UDM_ISPELL_PREFIX_DISABLED (integer)

UDM_ISPELL_TYPE_AFFIX (integer)

UDM_ISPELL_TYPE_SPELL (integer)

UDM_ISPELL_TYPE_DB (integer)

UDM_ISPELL_TYPE_SERVER (integer)

UDM_MATCH_WORD (integer)

UDM_MATCH_BEGIN (integer)

UDM_MATCH_SUBSTR (integer)

UDM_MATCH_END (integer)

Table of Contents

udm_add_search_limit

(PHP 4 >= 4.0.5)

udm_add_search_limit -- Add various search limits

Description

int **udm_add_search_limit** (int agent, int var, string val)

udm_add_search_limit() returns **TRUE** on success, **FALSE** on error. Adds search restrictions.

agent - a link to Agent, received after call to **udm_alloc_agent()**.

var - defines parameter, indicating limit.

val - defines value of the current parameter.

Possible *var* values:

- UDM_LIMIT_URL - defines document URL limitations to limit search through subsection of database. It supports SQL % and _ LIKE wildcards, where % matches any number of characters, even zero characters, and _ matches exactly one character. E.g. http://my.domain.__/catalog may stand for http://my.domain.ru/catalog and http://my.domain.ua/catalog.
- UDM_LIMIT_TAG - defines site TAG limitations. In indexer-conf you can assign specific TAGs to various sites and parts of a site. Tags in mnoGoSearch 3.1.x are lines, that may contain metasymbols % and _. Metasymbols allow searching among groups of tags. E.g. there are links with tags ABCD and ABCE, and search restriction is by ABC_ - the search will be made among both of the tags.
- UDM_LIMIT_LANG - defines document language limitations.
- UDM_LIMIT_CAT - defines document category limitations. Categories are similar to tag feature, but nested. So you can have one category inside another and so on. You have to use two characters for each level. Use a hex number going from 0-F or a 36 base number going from 0-Z. Therefore a top-level category like 'Auto' would be 01. If it has a subcategory like 'Ford', then it would be 01 (the parent category) and then 'Ford' which we will give 01. Put those together and you get 0101. If 'Auto' had another subcategory named 'VW', then it's id would be 01 because it belongs to the 'Ford' category and then 02 because it's the next category. So it's id would be 0102. If VW had a sub category called 'Engine' then it's id would start at 01 again and it would get the 'VW' id 02 and 'Auto' id of 01, making it 010201. If you want to search for sites under that category then you pass it cat=010201 in the url.

- UDM_LIMIT_DATE - defines limitation by date document was modified.

  Format of parameter value: a string with first character < or >, then with no space - date in unixtime format, for example:

  Udm_Add_Search_Limit($udm,UDM_LIMIT_DATE,"<908012006");

  If > character is used, then search will be restricted to those documents having modification date greater than entered. If <, then smaller.

udm_alloc_agent

(PHP 4 >= 4.0.5)

udm_alloc_agent -- Allocate mnoGoSearch session

Description

int **udm_alloc_agent** (string dbaddr [, string dbmode])

udm_alloc_agent() returns mnogosearch agent identifier on success, **FALSE** on error. This function creates a session with database parameters.

dbaddr - URL-style database description. Options (type, host, database name, port, user and password) to connect to SQL database. Do not matter for built-in text files support. Format: DBAddr DBType:[//[DBUser[:DBPass]@]DBHost[:DBPort]]/DBName/ Currently supported DBType values are: mysql, pgsql, msql, solid, mssql, oracle, ibase. Actually, it does not matter for native libraries support. But ODBC users should specify one of supported values. If your database type is not supported, you may use "unknown" instead.

dbmode - You may select SQL database mode of words storage. When "single" is specified, all words are stored in the same table. If "multi" is selected, words will be located in different tables depending of their lengths. "multi" mode is usually faster but requires more tables in database. If "crc" mode is selected, mnoGoSearch will store 32 bit integer word IDs calculated by CRC32 algorythm instead of words. This mode requres less disk space and it is faster comparing with "single" and "multi" modes. "crc-multi" uses the same storage structure with the "crc" mode, but also stores words in different tables depending on words lengths like "multi" mode. Format: DBMode single/multi/crc/crc-multi

> **Note:** *dbaddr* and *dbmode* must match those used during indexing.
>
> **Note:** In fact this function does not open connection to database and thus does not check entered login and password. Actual connection to database and login/password verification is done by **udm_find()**.

udm_api_version

(PHP 4 >= 4.0.5)

udm_api_version -- Get mnoGoSearch API version.

Description

int **udm_api_version** (void)

udm_api_version() returns mnoGoSearch API version number. E.g. if mnoGoSearch 3.1.10 API is used, this function will return `30110`.

This function allows user to identify which API functions are available, e.g. **udm_get_doc_count()** function is only available in mnoGoSearch 3.1.11 or later.

Example:

```
if (udm_api_version() >= 30111) {
   print  "Total number of urls in database: ".udm_get_doc_count($udm)."<br>\n";
   }
```

udm_cat_list

(PHP 4 >= 4.0.6)

udm_cat_list -- Get all the categories on the same level with the current one.

Description

array **udm_cat_list** (int agent, string category)

udm_cat_list() returns array listing all categories of the same level as current category in the categories tree.

The function can be useful for developing categories tree browser.

Returns array with the following format:

The array consists of pairs. Elements with even index numbers contain category paths, odd elements contain corresponding category names.

$array[0] will contain '020300'
$array[1] will contain 'Audi'
$array[2] will contain '020301'
$array[3] will contain 'BMW'
$array[4] will contain '020302'
$array[5] will contain 'Opel'
...
etc.

Following is an example of displaying links of the current level in format:
Audi
BMW
Opel
...

```
<?php
 $cat_list_arr = udm_cat_list($udm_agent,$cat);
 $cat_list = '';
 for ($i=0; $i<count($cat_list_arr); $i+=2) {
   $path = $cat_list_arr[$i];
   $name = $cat_list_arr[$i+1];
   $cat_list .= "<a href=\"$PHP_SELF?cat=$path\">$name</a><br>";
 }
?>
```

udm_cat_path

(PHP 4 >= 4.0.6)

udm_cat_path -- Get the path to the current category.

Description

array **udm_cat_path** (int agent, string category)

udm_cat_path() returns array describing path in the categories tree from the tree root to the current category.

agent - agent link identifier.

category - current category - the one to get path to.

Returns array with the following format:

The array consists of pairs. Elements with even index numbers contain category paths, odd elements contain corresponding category names.

For example, the call `$array=udm_cat_path($agent, '02031D');` may return the following array:

$array[0] will contain ''
$array[1] will contain 'Root'
$array[2] will contain '02'
$array[3] will contain 'Sport'
$array[4] will contain '0203'
$array[5] will contain 'Auto'
$array[4] will contain '02031D'
$array[5] will contain 'Ferrari'

Example 1. Specifying path to the current category in the following format: '> Root > Sport > Auto > Ferrari'

```
<?php
  $cat_path_arr = udm_cat_path($udm_agent,$cat);
  $cat_path = '';
  for ($i=0; $i<count($cat_path_arr); $i+=2) {
    $path = $cat_path_arr[$i];
    $name = $cat_path_arr[$i+1];
    $cat_path .= " > <a href=\"$PHP_SELF?cat=$path\">$name</a> ";
  }
?>
```

udm_check_charset

(PHP 4 >= 4.2.0)

udm_check_charset -- Check if the given charset is known to mnogosearch

Description

int **udm_check_charset** (int agent, string charset)

Warning
This function is currently not documented; only the argument list is available.

udm_check_stored

(PHP 4 >= 4.2.0)

udm_check_stored -- Check connection to stored

Description

int **udm_check_stored** (int agent, int link, string doc_id)

Warning
This function is currently not documented; only the argument list is available.

udm_clear_search_limits

(PHP 4 >= 4.0.5)

udm_clear_search_limits -- Clear all mnoGoSearch search restrictions

Description

int **udm_clear_search_limits** (int agent)

udm_clear_search_limits() resets defined search limitations and returns **TRUE**.

udm_close_stored

(PHP 4 >= 4.2.0)

udm_close_stored -- Close connection to stored

Description

int **udm_close_stored** (int agent, int link)

Warning
This function is currently not documented; only the argument list is available.

udm_crc32

(PHP 4 >= 4.2.0)

udm_crc32 -- Return CRC32 checksum of gived string

Description

int **udm_crc32** (int agent, string str)

Warning
This function is currently not documented; only the argument list is available.

udm_errno

(PHP 4 >= 4.0.5)

udm_errno -- Get mnoGoSearch error number

Description

int **udm_errno** (int agent)

udm_errno() returns mnoGoSearch error number, zero if no error.

agent - link to agent identifier, received after call to **udm_alloc_agent()**.

Receiving numeric agent error code.

udm_error

(PHP 4 >= 4.0.5)

udm_error -- Get mnoGoSearch error message

Description

string **udm_error** (int agent)

udm_error() returns mnoGoSearch error message, empty string if no error.

agent - link to agent identifier, received after call to **udm_alloc_agent()**.

Receiving agent error message.

udm_find

(PHP 4 >= 4.0.5)

udm_find -- Perform search

Description

int **udm_find** (int agent, string query)

udm_find() returns result link identifier on success, **FALSE** on error.

The search itself. The first argument - session, the next one - query itself. To find something just type words you want to find and press SUBMIT button. For example, "mysql odbc". You should not use quotes " in query, they are written here only to divide a query from other text. mnoGoSearch will find all documents that contain word "mysql" and/or word "odbc". Best documents having bigger weights will be displayed first. If you use search mode ALL, search will return documents that contain both (or more) words you entered. In case you use mode ANY, the search will return list of documents that contain any of the words you entered. If you want more advanced results you may use query language. You should select "bool" match mode in the search from.

mnoGoSearch understands the following boolean operators:

& - logical AND. For example, "mysql & odbc". mnoGoSearch will find any URLs that contain both "mysql" and "odbc".

| - logical OR. For example "mysql|odbc". mnoGoSearch will find any URLs, that contain word "mysql" or word "odbc".

~ - logical NOT. For example "mysql & ~odbc". mnoGoSearch will find URLs that contain word "mysql" and do not contain word "odbc" at the same time. Note that ~ just excludes given word from results. Query "~odbc" will find nothing!

() - group command to compose more complex queries. For example "(mysql | msql) & ~postgres". Query language is simple and powerful at the same time. Just consider query as usual boolean expression.

udm_free_agent

(PHP 4 >= 4.0.5)

udm_free_agent -- Free mnoGoSearch session

Description

int **udm_free_agent** (int agent)

udm_free_agent() returns **TRUE** on success, **FALSE** on error.

agent - link to agent identifier, received ` after call to **udm_alloc_agent()**.

Freeing up memory allocated for agent session.

udm_free_ispell_data

(PHP 4 >= 4.0.5)

udm_free_ispell_data -- Free memory allocated for ispell data

Description

int **udm_free_ispell_data** (int agent)

udm_free_ispell_data() always returns **TRUE**.

agent - agent link identifier, received after call to **udm_alloc_agent()**.

> **Note:** This function is supported beginning from version 3.1.12 of mnoGoSearch and it does not do anything in previous versions.

udm_free_res

(PHP 4 >= 4.0.5)

udm_free_res -- Free mnoGoSearch result

Description

int **udm_free_res** (int res)

udm_free_res() returns **TRUE** on success, **FALSE** on error.

res - a link to result identifier, received after call to **udm_find()**.

Freeing up memory allocated for results.

udm_get_doc_count

(PHP 4 >= 4.0.5)

udm_get_doc_count -- Get total number of documents in database.

Description

int **udm_get_doc_count** (int agent)

udm_get_doc_count() returns number of documents in database.

agent - link to agent identifier, received after call to **udm_alloc_agent()**.

Note: This function is supported only in mnoGoSearch 3.1.11 or later.

udm_get_res_field

(PHP 4 >= 4.0.5)

udm_get_res_field -- Fetch mnoGoSearch result field

Description

string **udm_get_res_field** (int res, int row, int field)

udm_get_res_field() returns result field value on success, **FALSE** on error.

res - a link to result identifier, received after call to **udm_find()**.

row - the number of the link on the current page. May have values from 0 to *UDM_PARAM_NUM_ROWS-1*.

field - field identifier, may have the following values:

- UDM_FIELD_URL - document URL field
- UDM_FIELD_CONTENT - document Content-type field (for example, text/html).
- UDM_FIELD_CATEGORY - document category field. Use **udm_cat_path()** to get full path to current category from the categories tree root. (This parameter is available only in PHP 4.0.6 or later).
- UDM_FIELD_TITLE - document title field.
- UDM_FIELD_KEYWORDS - document keywords field (from META KEYWORDS tag).
- UDM_FIELD_DESC - document description field (from META DESCRIPTION tag).
- UDM_FIELD_TEXT - document body text (the first couple of lines to give an idea of what the document is about).

- UDM_FIELD_SIZE - document size.
- UDM_FIELD_URLID - unique URL ID of the link.
- UDM_FIELD_RATING - page rating (as calculated by mnoGoSearch).
- UDM_FIELD_MODIFIED - last-modified field in unixtime format.
- UDM_FIELD_ORDER - the number of the current document in set of found documents.
- UDM_FIELD_CRC - document CRC.

udm_get_res_param

(PHP 4 >= 4.0.5)

udm_get_res_param -- Get mnoGoSearch result parameters

Description

string **udm_get_res_param** (int res, int param)

udm_get_res_param() returns result parameter value on success, **FALSE** on error.

res - a link to result identifier, received after call to **udm_find()**.

param - parameter identifier, may have the following values:

- UDM_PARAM_NUM_ROWS - number of received found links on the current page. It is equal to UDM_PARAM_PAGE_SIZE for all search pages, on the last page - the rest of links.
- UDM_PARAM_FOUND - total number of results matching the query.
- UDM_PARAM_WORDINFO - information on the words found. E.g. search for "a good book" will return "a: stopword, good:5637, book: 120"
- UDM_PARAM_SEARCHTIME - search time in seconds.
- UDM_PARAM_FIRST_DOC - the number of the first document displayed on current page.
- UDM_PARAM_LAST_DOC - the number of the last document displayed on current page.

udm_load_ispell_data

(PHP 4 >= 4.0.5)

udm_load_ispell_data -- Load ispell data

Description

int **udm_load_ispell_data** (int agent, int var, string val1, string val2, int flag)

udm_load_ispell_data() loads ispell data. Returns **TRUE** on success, **FALSE** on error.

agent - agent link identifier, received after call to **udm_alloc_agent()**.

var - parameter, indicating the source for ispell data. May have the following values:

After using this function to free memory allocated for ispell data, please use **udm_free_ispell_data()**, even if you use UDM_ISPELL_TYPE_SERVER mode.

The fastest mode is UDM_ISPELL_TYPE_SERVER. UDM_ISPELL_TYPE_TEXT is slower and UDM_ISPELL_TYPE_DB is the slowest. The above pattern is **TRUE** for mnoGoSearch 3.1.10 - 3.1.11. It is planned to speed up DB mode in future versions and it is going to be faster than TEXT mode.

- UDM_ISPELL_TYPE_DB - indicates that ispell data should be loaded from SQL. In this case, parameters *val1* and *val2* are ignored and should be left blank. *flag* should be equal to 1.

Note: *flag* indicates that after loading ispell data from defined source it sould be sorted (it is necessary for correct functioning of ispell). In case of loading ispell data from files there may be several calls to **udm_load_ispell_data()**, and there is no sense to sort data after every call, but only after the last one. Since in db mode all the data is loaded by one call, this parameter should have the value 1. In this mode in case of error, e.g. if ispell tables are absent, the function will return **FALSE** and code and error message will be accessible through udm_error() and udm_errno().

Example:

```
if (! udm_load_ispell_data($udm,UDM_ISPELL_TYPE_DB,'','',1)) {
  printf("Error #%d: '%s'\n", udm_errno($udm), udm_error($udm));
  exit;
}
```

- UDM_ISPELL_TYPE_AFFIX - indicates that ispell data should be loaded from file and initiates loading affixes file. In this case *val1* defines double letter language code for which affixes are loaded, and *val2* - file path. Please note, that if a relative path entered, the module looks for the file not in UDM_CONF_DIR, but in relation to current path, i.e. to the path where the script is executed. In case of error in this mode, e.g. if file is absent, the function will return **FALSE**, and an error message will be displayed. Error message text cannot be accessed through udm_error() and udm_errno(), since those functions can only return messages associated with SQL. Please, see *flag* parameter description in UDM_ISPELL_TYPE_DB.

  Example:

```
if ((! udm_load_ispell_data($udm,UDM_ISPELL_TYPE_AFFIX,'en','/opt/ispell/en.aff',0)) ||
        (! udm_load_ispell_data($udm,UDM_ISPELL_TYPE_AFFIX,'ru','/opt/ispell/ru.aff',0)) ||
        (! udm_load_ispell_data($udm,UDM_ISPELL_TYPE_SPELL,'en','/opt/ispell/en.dict',0)) ||
        (! udm_load_ispell_data($udm,UDM_ISPELL_TYPE_SPELL,'ru','/opt/ispell/ru.dict',1))) {
     exit;
     }
```

  Note: *flag* is equal to 1 only in the last call.

- UDM_ISPELL_TYPE_SPELL - indicates that ispell data should be loaded from file and initiates loading of ispell dictionary file. In this case *val1* defines double letter language code for which affixes are loaded, and *val2* - file path. Please note, that if a relative path entered, the module looks for the file not in UDM_CONF_DIR, but in relation to current path, i.e. to the path where the script is executed. In case of error in this mode, e.g. if file is absent, the function will return **FALSE**, and an error message will be displayed. Error message text cannot be accessed through udm_error() and udm_errno(), since those functions can only return messages associated with SQL. Please, see *flag* parameter description in UDM_ISPELL_TYPE_DB.

```
if ((! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_AFFIX,'en','/opt/ispell/en.aff',0)) ||
        (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_AFFIX,'ru','/opt/ispell/ru.aff',0)) ||
        (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_SPELL,'en','/opt/ispell/en.dict',0)) ||
        (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_SPELL,'ru','/opt/ispell/ru.dict',1))) {
      exit;
      }
```

  Note: *flag* is equal to 1 only in the last call.

- UDM_ISPELL_TYPE_SERVER - enables spell server support. *val1* parameter indicates address of the host running spell server. *val2*` is not used yet, but in future releases it is going to indicate number of port used by spell server. *flag* parameter in this case is not needed since ispell data is stored on spellserver already sorted.

  Spelld server reads spell-data from a separate configuration file (/usr/local/mnogosearch/etc/spelld.conf by default), sorts it and stores in memory. With clients server communicates in two ways: to indexer all the data is transferred (so that indexer starts faster), from search.cgi server receives word to normalize and then passes over to client (search.cgi) list of normalized word forms. This allows fastest, compared to db and text modes processing of search queries (by omitting loading and sorting all the spell data).

  udm_load_ispell_data() function in UDM_ISPELL_TYPE_SERVER mode does not actually load ispell data, but only defines server address. In fact, server is automatically used by udm_find() function when performing search. In case of errors, e.g. if spellserver is not running or invalid host indicated, there are no messages returned and ispell conversion does not work.

  Note: This function is available in mnoGoSearch 3.1.12 or later.

  Example:

```
if (!udm_load_ispell_data($udm,UDM_ISPELL_TYPE_SERVER,'','',1)) {
    printf("Error loading ispell data from server<br>\n");
    exit;
}
```

udm_open_stored

(PHP 4 >= 4.2.0)

udm_open_stored -- Open connection to stored

Description

int **udm_open_stored** (int agent, string storedaddr)

Warning
This function is currently not documented; only the argument list is available.

udm_set_agent_param

(PHP 4 >= 4.0.5)

udm_set_agent_param -- Set mnoGoSearch agent session parameters

Description

int **udm_set_agent_param** (int agent, int var, string val)

udm_set_agent_param() returns **TRUE** on success, **FALSE** on error. Defines mnoGoSearch session parameters.

The following parameters and their values are available:

- UDM_PARAM_PAGE_NUM - used to choose search results page number (results are returned by pages beginning from 0, with UDM_PARAM_PAGE_SIZE results per page).
- UDM_PARAM_PAGE_SIZE - number of search results displayed on one page.
- UDM_PARAM_SEARCH_MODE - search mode. The following values available: UDM_MODE_ALL - search for all words; UDM_MODE_ANY - search for any word; UDM_MODE_PHRASE - phrase search; UDM_MODE_BOOL - boolean search. See **udm_find()** for details on boolean search.
- UDM_PARAM_CACHE_MODE - turns on or off search result cache mode. When enabled, the search engine will store search results to disk. In case a similar search is performed later, the engine will take results from the cache for faster performance. Available values: UDM_CACHE_ENABLED, UDM_CACHE_DISABLED.
- UDM_PARAM_TRACK_MODE - turns on or off trackquery mode. Since version 3.1.2 mnoGoSearch has a query tracking support. Note that tracking is implemented in SQL version only and not available in built-in database. To use tracking, you have to create tables for tracking support. For MySQL, use create/mysql/track.txt. When doing a search, front-end uses those tables to store query words, a number of found documents and current UNIX timestamp in seconds. Available values: UDM_TRACK_ENABLED, UDM_TRACK_DISABLED.
- UDM_PARAM_PHRASE_MODE - defines whether index database using phrases ("phrase" parameter in indexer.conf). Possible values: UDM_PHRASE_ENABLED and UDM_PHRASE_DISABLED. Please note, that if phrase search is enabled (UDM_PHRASE_ENABLED), it is still possible to do search in any mode (ANY, ALL, BOOL or PHRASE). In 3.1.10 version of mnoGoSearch phrase search is supported only in sql and built-in database modes, while beginning with 3.1.11 phrases are supported in cachemode as well.

  Examples of phrase search:

  "Arizona desert" - This query returns all indexed documents that contain "Arizona desert" as a phrase. Notice that you need to put double quotes around the phrase
- UDM_PARAM_CHARSET - defines local charset. Available values: set of charsets supported by mnoGoSearch, e.g. koi8-r, cp1251, ...
- UDM_PARAM_STOPFILE - Defines name and path to stopwords file. (There is a small difference with mnoGoSearch - while in mnoGoSearch if relative path or no path entered, it looks for this file in relation to UDM_CONF_DIR, the module looks for the file in relation to current path, i.e. to the path where the php script is executed.)
- UDM_PARAM_STOPTABLE - Load stop words from the given SQL table. You may use several StopwordTable commands. This command has no effect when compiled without SQL database support.
- UDM_PARAM_WEIGHT_FACTOR - represents weight factors for specific document parts. Currently body, title, keywords, description, url are supported. To activate this feature please use degrees of 2 in *Weight commands of the indexer.conf.

Let's imagine that we have these weights:

```
URLWeight     1
BodyWeight    2
TitleWeight   4
KeywordWeight 8
DescWeight    16
```

As far as indexer uses bit OR operation for word weights when some word presents several time in the same document, it is possible at search time to detect word appearance in different document parts. Word which appears only in the body will have 00000010 argegate weight (in binary notation). Word used in all document parts will have 00011111 aggregate weight.

This parameter's value is a string of hex digits ABCDE. Each digit is a factor for corresponding bit in word weight. For the given above weights configuration:

```
E is a factor for weight 1  (URL Weight bit)
D is a factor for weight 2  (BodyWeight bit)
C is a factor for weight 4  (TitleWeight bit)
B is a factor for weight 8  (KeywordWeight bit)
A is a factor for weight 16 (DescWeight bit)
```

Examples:

UDM_PARAM_WEIGHT_FACTOR=00001 will search through URLs only.

UDM_PARAM_WEIGHT_FACTOR=00100 will search through Titles only.

UDM_PARAM_WEIGHT_FACTOR=11100 will search through Title,Keywords,Description but not through URL and Body.

UDM_PARAM_WEIGHT_FACTOR=F9421 will search through:

```
Description with factor 15  (F hex)
Keywords with factor 9
Title with factor  4
Body with factor 2
URL with factor 1
```

If UDM_PARAM_WEIGHT_FACTOR variable is ommited, original weight value is taken to sort results. For a given above weight configuration it means that document description has a most big weight 16.

- UDM_PARAM_WORD_MATCH - word match. You may use this parameter to choose word match type. This feature works only in "single" and "multi" modes using SQL based and built-in database. It does not work in cachemode and other modes since they use word CRC and do not support substring search. Available values:

  UDM_MATCH_BEGIN - word beginning match;

  UDM_MATCH_END - word ending match;

  UDM_MATCH_WORD - whole word match;

  UDM_MATCH_SUBSTR - word substring match.

- UDM_PARAM_MIN_WORD_LEN - defines minimal word length. Any word shorter this limit is considered to be a stopword. Please note that this parameter value is inclusive, i.e. if UDM_PARAM_MIN_WORD_LEN=3, a word 3 characters long will not be considered a stopword, while a word 2 characters long will be. Default value is 1.

- UDM_PARAM_ISPELL_PREFIXES - Possible values: UDM_PREFIXES_ENABLED and UDM_PREFIXES_DISABLED, that respectively enable or disable using prefixes. E.g. if a word "tested" is in search query, also words like "test", "testing", etc. Only suffixes are supported by default. Prefixes usually change word meanings, for example if somebody is searching for the word "tested" one hardly wants "untested" to be found. Prefixes support may also be found useful for site's spelling checking purposes. In order to enable ispell, you have to load ispell data with **udm_load_ispell_data()**.

- UDM_PARAM_CROSS_WORDS - enables or disables crosswords support. Possible values: UDM_CROSS_WORDS_ENABLED and UDM_CROSS_WORDS_DISABLED.

  The corsswords feature allows to assign words between and also to a document this link leads to. It

works in SQL database mode and is not supported in built-in database and Cachemode.

Note: Crosswords are supported only in mnoGoSearch 3.1.11 or later.

- UDM_PARAM_VARDIR - specifies a custom path to directory where indexer stores data when using built-in database and in cache mode. By default `/var` directory of mnoGoSearch installation is used. Can have only string values. The parameter is available in PHP 4.1.0 or later.
- UDM_PARAM_VARDIR - specifies a custom path to directory where indexer stores data when using built-in database and in cache mode. By default `/var` directory of mnoGoSearch installation is used. Can have only string values. The parameter is available in PHP 4.1.0 or later.

LXII. mSQL functions

Introduction

These functions allow you to access mSQL database servers. More information about mSQL can be found at http://www.hughes.com.au/.

Requirements

Installation

In order to have these functions available, you must compile PHP with msql support by using the `--with-msql[=DIR]` option. DIR is the mSQL base install directory, defaults to `/usr/local/Hughes`.

Note to Win32 Users: In order to enable this module on a Windows environment, you must copy *msql.dll* from the DLL folder of the PHP/Win32 binary package to the SYSTEM32 folder of your windows machine. (Ex: C:\WINNT\SYSTEM32 or C:\WINDOWS\SYSTEM32)

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. mSQL configuration options

Name	Default	Changeable
msql.allow_persistent	"On"	PHP_INI_SYSTEM
msql.max_persistent	"-1"	PHP_INI_SYSTEM
msql.max_links	"-1"	PHP_INI_SYSTEM

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Here is a short explanation of the configuration directives.

`msql.allow_persistent` **boolean**

Whether to allow persistent mSQL connections.

`msql.max_persistent` **integer**

The maximum number of persistent mSQL connections per process.

`msql.max_links` **integer**

The maximum number of mSQL connections per process, including persistent connections.

Resource Types

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

MSQL_ASSOC (integer)

MSQL_NUM (integer)

MSQL_BOTH (integer)

Table of Contents

msql_affected_rows

(PHP 3>= 3.0.6, PHP 4)

msql_affected_rows -- Returns number of affected rows

Description

int **msql_affected_rows** (int query_identifier)

Returns number of affected ("touched") rows by a specific query (i.e. the number of rows returned by a SELECT, the number of rows modified by an update, or the number of rows removed by a delete).

See also: **msql_query()**.

msql_close

(PHP 3, PHP 4)

msql_close -- Close mSQL connection

Description

int **msql_close** (int link_identifier)

Returns **TRUE** on success, **FALSE** on error.

msql_close() closes the link to a mSQL database that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

Note that this isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

msql_close() will not close persistent links generated by **msql_pconnect()**.

See also: **msql_connect()** and **msql_pconnect()**.

msql_connect

(PHP 3, PHP 4)

msql_connect -- Open mSQL connection

Description

int **msql_connect** ([string hostname [, string server [, string username [, string password]]]])

Returns a positive mSQL link identifier on success, or **FALSE** on error.

msql_connect() establishes a connection to a mSQL server. The *server* parameter can also include a port number. eg. "hostname:port". It defaults to 'localhost'.

In case a second call is made to **msql_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling **msql_close()**.

See also **msql_pconnect()**, **msql_close()**.

msql_create_db

(PHP 3, PHP 4)

msql_create_db -- Create mSQL database

Description

int **msql_create_db** (string database name [, int link_identifier])

msql_create_db() attempts to create a new database on the server associated with the specified link identifier.

See also: **msql_drop_db()**.

msql_createdb

(PHP 3, PHP 4)

msql_createdb -- Create mSQL database

Description

int **msql_createdb** (string database name [, int link_identifier])

Identical to **msql_create_db()**.

msql_data_seek

(PHP 3, PHP 4)

msql_data_seek -- Move internal row pointer

Description

int **msql_data_seek** (int query_identifier, int row_number)

Returns **TRUE** on success or **FALSE** on failure.

msql_data_seek() moves the internal row pointer of the mSQL result associated with the specified query identifier to point to the specifyed row number. The next call to **msql_fetch_row()** would return that row.

See also: **msql_fetch_row()**.

msql_dbname

(PHP 3, PHP 4)

msql_dbname -- Get current mSQL database name

Description

string **msql_dbname** (int query_identifier, int i)

msql_dbname() returns the database name stored in position *i* of the result pointer returned from the **msql_listdbs()** function. The **msql_numrows()** function can be used to determine how many database names are available.

msql_drop_db

(PHP 3, PHP 4)

msql_drop_db -- Drop (delete) mSQL database

Description

int **msql_drop_db** (string database_name, int link_identifier)

Returns **TRUE** on success or **FALSE** on failure.

msql_drop_db() attempts to drop (remove) an entire database from the server associated with the specified link identifier.

See also: **msql_create_db()**.

msql_dropdb

msql_dropdb -- Drop (delete) mSQL database

Description

See **msql_drop_db()**.

msql_error

(PHP 3, PHP 4)

msql_error -- Returns error message of last msql call

Description

string **msql_error** ([int link_identifier])

Errors coming back from the mSQL database backend no longer issue warnings. Instead, use these functions to retrieve the error string.

msql_fetch_array

(PHP 3, PHP 4)

msql_fetch_array -- Fetch row as array

Description

int **msql_fetch_array** (int query_identifier [, int result_type])

Returns an array that corresponds to the fetched row, or **FALSE** if there are no more rows.

msql_fetch_array() is an extended version of **msql_fetch_row()**. In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

The second optional argument *result_type* in **msql_fetch_array()** is a constant and can take the following values: MSQL_ASSOC, MSQL_NUM, and MSQL_BOTH.

Be careful if you are retrieving results from a query that may return a record that contains only one field that has a value of o (or an empty string, or **NULL**).

An important thing to note is that using **msql_fetch_array()** is NOT significantly slower than using **msql_fetch_row()**, while it provides a significant added value.

For further details, also see **msql_fetch_row()**.

msql_fetch_field

(PHP 3, PHP 4)

msql_fetch_field -- Get field information

Description

object **msql_fetch_field** (int query_identifier, int field_offset)

Returns an object containing field information

msql_fetch_field() can be used in order to obtain information about fields in a certain query result. If the field offset isn't specified, the next field that wasn't yet retreived by **msql_fetch_field()** is retreived.

The properties of the object are:

- name - column name

- table - name of the table the column belongs to
- not_null - 1 if the column cannot be **NULL**
- primary_key - 1 if the column is a primary key
- unique - 1 if the column is a unique key
- type - the type of the column

See also **msql_field_seek()**.

msql_fetch_object

(PHP 3, PHP 4)

msql_fetch_object -- Fetch row as object

Description

int **msql_fetch_object** (int query_identifier [, int result_type])

Returns an object with properties that correspond to the fetched row, or **FALSE** if there are no more rows.

msql_fetch_object() is similar to **msql_fetch_array()**, with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

The optional second argument *result_type* in **msql_fetch_array()** is a constant and can take the following values: MSQL_ASSOC, MSQL_NUM, and MSQL_BOTH.

Speed-wise, the function is identical to **msql_fetch_array()**, and almost as quick as **msql_fetch_row()** (the difference is insignificant).

See also: **msql_fetch_array()** and **msql_fetch_row()**.

msql_fetch_row

(PHP 3, PHP 4)

msql_fetch_row -- Get row as enumerated array

Description

array **msql_fetch_row** (int query_identifier)

Returns an array that corresponds to the fetched row, or **FALSE** if there are no more rows.

msql_fetch_row() fetches one row of data from the result associated with the specified query identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to **msql_fetch_row()** would return the next row in the result set, or **FALSE** if there are no more rows.

See also: **msql_fetch_array()**, **msql_fetch_object()**, **msql_data_seek()**, and **msql_result()**.

msql_field_seek

(PHP 3, PHP 4)

msql_field_seek -- Set field offset

Description

int **msql_field_seek** (int query_identifier, int field_offset)

Seeks to the specified field offset. If the next call to **msql_fetch_field()** won't include a field offset, this field would be returned.

See also: **msql_fetch_field()**.

msql_fieldflags

(PHP 3, PHP 4)

msql_fieldflags -- Get field flags

Description

string **msql_fieldflags** (int query_identifier, int i)

msql_fieldflags() returns the field flags of the specified field. Currently this is either, "not `NULL`", "primary key", a combination of the two or "" (an empty string).

msql_fieldlen

(PHP 3, PHP 4)

msql_fieldlen -- Get field length

Description

int **msql_fieldlen** (int query_identifier, int i)

msql_fieldlen() returns the length of the specified field.

msql_fieldname

(PHP 3, PHP 4)

msql_fieldname -- Get field name

Description

string **msql_fieldname** (int query_identifier, int field)

msql_fieldname() returns the name of the specified field. *`query_identifier`* is the query identifier, and *`field`* is the field index. `msql_fieldname($result, 2);` will return the name of the second field in the result associated with the result identifier.

msql_fieldtable

(PHP 3, PHP 4)

msql_fieldtable -- Get table name for field

Description

int **msql_fieldtable** (int query_identifier, int field)

Returns the name of the table *`field`* was fetched from.

msql_fieldtype

(PHP 3, PHP 4)

msql_fieldtype -- Get field type

Description

string **msql_fieldtype** (int query_identifier, int i)

msql_fieldtype() is similar to the **msql_fieldname()** function. The arguments are identical, but the field type is returned. This will be one of "int", "char" or "real".

msql_free_result

(PHP 3, PHP 4)

msql_free_result -- Free result memory

Description

int **msql_free_result** (int query_identifier)

msql_free_result() frees the memory associated with *query_identifier*. When PHP completes a request, this memory is freed automatically, so you only need to call this function when you want to make sure you don't use too much memory while the script is running.

msql_freeresult

msql_freeresult -- Free result memory

Description

See **msql_free_result()**

msql_list_dbs

(PHP 3, PHP 4)

msql_list_dbs -- List mSQL databases on server

Description

int **msql_list_dbs** (void)

msql_list_dbs() will return a result pointer containing the databases available from the current msql daemon. Use the **msql_dbname()** function to traverse this result pointer.

msql_list_fields

(PHP 3, PHP 4)

msql_list_fields -- List result fields

Description

int **msql_list_fields** (string database, string tablename)

msql_list_fields() retrieves information about the given tablename. Arguments are the database name and the table name. A result pointer is returned which can be used with **msql_fieldflags()**, **msql_fieldlen()**, **msql_fieldname()**, and **msql_fieldtype()**. A query identifier is a positive integer. The function returns `-1` if a error occurs. A string describing the error will be placed in `$phperrmsg`, and unless the function was called as `@msql_list_fields()` then this error string will also be printed out.

See also **msql_error()**.

msql_list_tables

(PHP 3, PHP 4)

msql_list_tables -- List tables in an mSQL database

Description

int **msql_list_tables** (string database)

msql_list_tables() takes a database name and result pointer much like the **msql()** function. The **msql_tablename()** function should be used to extract the actual table names from the result pointer.

msql_listdbs

msql_listdbs -- List mSQL databases on server

Description

See **msql_list_dbs()**.

msql_listfields

msql_listfields -- List result fields

Description

See **msql_list_fields()**.

msql_listtables

msql_listtables -- List tables in an mSQL database

Description

See **msql_list_tables()**.

msql_num_fields

(PHP 3, PHP 4)

msql_num_fields -- Get number of fields in result

Description

int **msql_num_fields** (int query_identifier)

msql_num_fields() returns the number of fields in a result set.

See also: **msql()**, **msql_query()**, **msql_fetch_field()**, and **msql_num_rows()**.

msql_num_rows

(PHP 3, PHP 4)

msql_num_rows -- Get number of rows in result

Description

int **msql_num_rows** (int query_identifier)

msql_num_rows() returns the number of rows in a result set.

See also: **msql()**, **msql_query()**, and **msql_fetch_row()**.

msql_numfields

(PHP 3, PHP 4)

msql_numfields -- Get number of fields in result

Description

int **msql_numfields** (int query_identifier)

Identical to **msql_num_fields()**.

msql_numrows

(PHP 3, PHP 4)

msql_numrows -- Get number of rows in result

Description

int **msql_numrows** (void)

Identical to **msql_num_rows()**.

msql_pconnect

(PHP 3, PHP 4)

msql_pconnect -- Open persistent mSQL connection

Description

int **msql_pconnect** ([string server [, string username [, string password]]])

Returns a positive mSQL persistent link identifier on success, or **FALSE** on error.

msql_pconnect() acts very much like **msql_connect()** with two major differences.

First, when connecting, the function would first try to find a (persistent) link that's already open with the same host. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (**msql_close()** will not close links established by **msql_pconnect()**).

This type of links is therefore called 'persistent'.

msql_query

(PHP 3, PHP 4)

msql_query -- Send mSQL query

Description

int **msql_query** (string query, int link_identifier)

msql_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if **msql_connect()** was called, and use it.

Returns a positive mSQL query identifier on success, or **FALSE** on error.

Example 1. msql_query()

```
<?php
$link = msql_connect("dbserver")
   or die("unable to connect to msql server: ".msql_error());
msql_select_db("db", $link)
   or die("unable to select database 'db': ".msql_error());

$result = msql_query("SELECT * FROM table WHERE id=1", $link);
if (!$result) {
   die("query failed: ".msql_error());
}

while ($row = msql_fetch_array($result)) {
    echo $row["id"];
}
?>
```

See also: **msql()**, **msql_select_db()**, and **msql_connect()**.

msql_regcase

msql_regcase -- Make regular expression for case insensitive match

Description

See **sql_regcase()**.

msql_result

(PHP 3, PHP 4)

msql_result -- Get result data

Description

int **msql_result** (int query_identifier, int i, mixed field)

Returns the contents of the cell at the row and offset in the specified mSQL result set.

msql_result() returns the contents of one cell from a mSQL result set. The field argument can be the field's offset, or the field's name, or the field's table dot field's name (fieldname.tablename). If the column name has been aliased ('select foo as bar from ...'), use the alias instead of the column name.

When working on large result sets, you should consider using one of the functions that fetch an entire row (specified below). As these functions return the contents of multiple cells in one function call, they're MUCH quicker than **msql_result()**. Also, note that specifying a numeric offset for the field argument is much quicker than specifying a fieldname or tablename.fieldname argument.

Recommended high-performance alternatives: **msql_fetch_row()**, **msql_fetch_array()**, and **msql_fetch_object()**.

msql_select_db

(PHP 3, PHP 4)

msql_select_db -- Select mSQL database

Description

int **msql_select_db** (string database_name, int link_identifier)

Returns **TRUE** on success, **FALSE** on error.

msql_select_db() sets the current active database on the server that's associated with the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if **msql_connect()** was called, and use it.

Every subsequent call to **msql_query()** will be made on the active database.

See also: **msql_connect()**, **msql_pconnect()**, and **msql_query()**.

msql_selectdb

msql_selectdb -- Select mSQL database

Description

See **msql_select_db()**.

msql_tablename

(PHP 3, PHP 4)

msql_tablename -- Get table name of field

Description

string **msql_tablename** (int query_identifier, int field)

msql_tablename() takes a result pointer returned by the **msql_list_tables()** function as well as an integer index and returns the name of a table. The **msql_numrows()** function may be used to determine the number of tables in the result pointer.

Example 1. msql_tablename() example

```
<?php
msql_connect ("localhost");
$result = msql_list_tables ("wisconsin");
$i = 0;
while ($i < msql_numrows ($result)) {
    $tb_names[$i] = msql_tablename ($result, $i);
    echo $tb_names[$i] . "<BR>";
    $i++;
}
?>
```

msql

(PHP 3, PHP 4)

msql -- Send mSQL query

Description

int **msql** (string database, string query, int link_identifier)

Returns a positive mSQL query identifier to the query result, or **FALSE** on error.

msql() selects a database and executes a query on it. If the optional link identifier isn't specified, the function will try to find an open link to the mSQL server and if no such link is found it'll try to create one as if **msql_connect()** was called with no arguments (see **msql_connect()**).

LXIII. MySQL Functions

Introduction

These functions allow you to access MySQL database servers. More information about MySQL can be found at http://www.mysql.com/.

Documentation for MySQL can be found at http://www.mysql.com/documentation/.

Requirements

In order to have these functions available, you must compile PHP with MySQL support.

Installation

By using the `--with-mysql[=DIR]` configuration option you enable PHP to access MySQL databases. If you use this option without specifying the path to MySQL, PHP will use the built-in MySQL client libraries. With PHP 4 MySQL support is always enabled; if you don't specify the configure option, the bundled libraries are used. Users who run other applications that use MySQL (for example, running PHP 3 and PHP 4 as concurrent apache modules, or auth-mysql) should always specify the path to MySQL: `--with-mysql=/path/to/mysql`. This will force PHP to use the client libraries installed by MySQL, avoiding any conflicts.

The windows version of PHP has built in support for this extension. You do not need to load any additional extension in order to use these functions.

Warning
Crashes and startup problems of PHP may be encountered when loading this extension in conjunction with the recode extension. See the recode extension for more information.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. MySQL Configuration Options

Name	Default	Changeable
mysql.allow_persistent	"On"	PHP_INI_SYSTEM
mysql.max_persistent	"-1"	PHP_INI_SYSTEM
mysql.max_links	"-1"	PHP_INI_SYSTEM
mysql.default_port	NULL	PHP_INI_ALL
mysql.default_socket	NULL	PHP_INI_ALL
mysql.default_host	NULL	PHP_INI_ALL
mysql.default_user	NULL	PHP_INI_ALL
mysql.default_password	NULL	PHP_INI_ALL
mysql.connect_timeout	"0"	PHP_INI_SYSTEM

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Here is a short explanation of the configuration directives.

`mysql.allow_persistent` **boolean**

Whether to allow persistent connections to MySQL.

`mysql.max_persistent` **integer**

The maximum number of persistent MySQL connections per process.

mysql.max_links **integer**

The maximum number of MySQL connections per process, including persistent connections.

mysql.default_port **string**

The default TCP port number to use when connecting to the database server if no other port is specified. If no default is specified, the port will be obtained from the `MYSQL_TCP_PORT` environment variable, the `mysql-tcp` entry in `/etc/services` or the compile-time `MYSQL_PORT` constant, in that order. Win32 will only use the `MYSQL_PORT` constant.

mysql.default_socket **string**

The default socket name to use when connecting to a local database server if no other socket name is specified.

mysql.default_host **string**

The default server host to use when connecting to the database server if no other host is specified. Doesn't apply in safe mode.

mysql.default_user **string**

The default user name to use when connecting to the database server if no other name is specified. Doesn't apply in safe mode.

mysql.default_password **string**

The default password to use when connecting to the database server if no other password is specified. Doesn't apply in safe mode.

mysql.connect_timeout **integer**

Connect timeout in seconds. On Linux this timeout is also used for waiting for the first answer from the server.

Resource Types

There are two resource types used in the MySQL module. The first one is the link identifier for a database connection, the second a resource which holds the result of a query.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

Since PHP 4.3.0 it is possible to specify additional client flags for the **mysql_connect()** and **mysql_pconnect()** functions. The following constants are defined:

Table 2. MySQL client constants

constant	description
MYSQL_CLIENT_COMPRESS	use compression protocol
MYSQL_CLIENT_IGNORE_SPACE	Allow space after function names
MYSQL_CLIENT_INTERACTIVE	Allow interactive_timeout seconds (instead of wait_timeout) of inactivity before closing the connection.

The function **mysql_fetch_array()** uses a constant for the different types of result arrays. The following constants are defined:

Table 3. MySQL fetch constants

constant	description
MYSQL_ASSOC	Columns are returned into the array having the fieldname as the array index.
MYSQL_BOTH	Columns are returned into the array having both a numerical index and the fieldname as the array index.
MYSQL_NUM	Columns are returned into the array having a numerical index to the fields. This index starts with 0, the first field in the result.

Examples

This simple example shows how to connect, execute a query, print resulting rows and disconnect from a MySQL database.

Example 1. MySQL extension overview example

```
<?php
    /* Connecting, selecting database */
    $link = mysql_connect("mysql_host", "mysql_user", "mysql_password")
        or die("Could not connect");
    print "Connected successfully";
    mysql_select_db("my_database") or die("Could not select database");

    /* Performing SQL query */
    $query = "SELECT * FROM my_table";
    $result = mysql_query($query) or die("Query failed");

    /* Printing results in HTML */
    print "<table>\n";
    while ($line = mysql_fetch_array($result, MYSQL_ASSOC)) {
        print "\t<tr>\n";
        foreach ($line as $col_value) {
            print "\t\t<td>$col_value</td>\n";
        }
        print "\t</tr>\n";
    }
    print "</table>\n";

    /* Free resultset */
    mysql_free_result($result);

    /* Closing connection */
    mysql_close($link);
?>
```

Table of Contents

mysql_pconnect -- Open a persistent connection to a MySQL server
mysql_ping -- Ping a server connection or reconnect if there is no connection
mysql_query -- Send a MySQL query
mysql_real_escape_string -- Escapes special characters in a string for use in a SQL statement, taking into account the current charset of the connection.
mysql_result -- Get result data
mysql_select_db -- Select a MySQL database
mysql_stat -- Get current system status
mysql_tablename -- Get table name of field
mysql_thread_id -- Return the current thread ID
mysql_unbuffered_query -- Send an SQL query to MySQL, without fetching and buffering the result rows

mysql_affected_rows

(PHP 3, PHP 4)

mysql_affected_rows -- Get number of affected rows in previous MySQL operation

Description

int **mysql_affected_rows** ([resource link_identifier])

mysql_affected_rows() returns the number of rows affected by the last INSERT, UPDATE or DELETE query associated with *link_identifier*. If the link identifier isn't specified, the last link opened by **mysql_connect()** is assumed.

> **Note:** If you are using transactions, you need to call **mysql_affected_rows()** after your INSERT, UPDATE, or DELETE query, not after the commit.

If the last query was a DELETE query with no WHERE clause, all of the records will have been deleted from the table but this function will return zero.

> **Note:** When using UPDATE, MySQL will not update columns where the new value is the same as the old value. This creates the possiblity that **mysql_affected_rows()** may not actually equal the number of rows matched, only the number of rows that were literally affected by the query.

mysql_affected_rows() does not work with SELECT statements; only on statements which modify records. To retrieve the number of rows returned by a SELECT, use **mysql_num_rows()**.

If the last query failed, this function will return -1.

Example 1. Delete-Query

```
<?php
    /* connect to database */
    mysql_pconnect("localhost", "mysql_user", "mysql_password") or
        die("Could not connect: " . mysql_error());

    /* this should return the correct numbers of deleted records */
    mysql_query("DELETE FROM mytable WHERE id < 10");
    printf ("Records deleted: %d\n", mysql_affected_rows());

    /* without a where clause in a delete statement, it should return 0 */
    mysql_query("DELETE FROM mytable");
    printf ("Records deleted: %d\n", mysql_affected_rows());
?>
```

The above example would produce the following output:

```
Records deleted: 10
Records deleted: 0
```

Example 2. Update-Query

```
<?php
    /* connect to database */
    mysql_pconnect("localhost", "mysql_user", "mysql_password") or
        die("Could not connect: " . mysql_error());

    /* Update records */
    mysql_query("UPDATE mytable SET used=1 WHERE id < 10");
    printf ("Updated records: %d\n", mysql_affected_rows());
mysql_query("COMMIT");
?>
```

The above example would produce the following output:

```
Updated Records: 10
```

See also: mysql_num_rows(), mysql_info().

mysql_change_user

(PHP 3>= 3.0.13)

mysql_change_user -- Change logged in user of the active connection

Description

int **mysql_change_user** (string user, string password [, string database [, resource link_identifier]])

mysql_change_user() changes the logged in user of the current active connection, or the connection given by the optional *link_identifier* parameter. If a database is specified, this will be the current database after the user has been changed. If the new user and password authorization fails, the current connected user stays active. Returns **TRUE** on success or **FALSE** on failure.

Note: This function was introduced in PHP 3.0.13 and requires MySQL 3.23.3 or higher. It is not available in PHP 4.

mysql_client_encoding

(PHP 4 >= 4.3.0)

mysql_client_encoding -- Returns the name of the character set

Description

int **mysql_client_encoding** ([resource link_identifier])

mysql_client_encoding() returns the default character set name for the current connection.

Example 1. mysql_client_encoding() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$charset = mysql_client_encoding($link);
printf ("current character set is %s\n", $charset);
?>
```

The above example would produce the following output:

```
current character set is latin1
```

See also: mysql_real_escape_string()

mysql_close

(PHP 3, PHP 4)

mysql_close -- Close MySQL connection

Description

bool **mysql_close** ([resource link_identifier])

Returns **TRUE** on success or **FALSE** on failure.

mysql_close() closes the connection to the MySQL server that's associated with the specified link identifier. If *link_identifier* isn't specified, the last opened link is used.

Using **mysql_close()** isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution. See also freeing resources.

> **Note: mysql_close()** will not close persistent links created by **mysql_pconnect()**.

Example 1. MySQL close example

```
<?php
    $link = mysql_connect("localhost", "mysql_user", "mysql_password")
        or die("Could not connect: " . mysql_error());
    print ("Connected successfully");
    mysql_close($link);
?>
```

See also: **mysql_connect()**, and **mysql_pconnect()**.

mysql_connect

(PHP 3, PHP 4)

mysql_connect -- Open a connection to a MySQL Server

Description

resource **mysql_connect** ([string server [, string username [, string password [, bool new_link [, int client_flags]]]]])

Returns a MySQL link identifier on success, or **FALSE** on failure.

mysql_connect() establishes a connection to a MySQL server. The following defaults are assumed for missing optional parameters: *server* = 'localhost:3306', *username* = name of the user that owns the server process and *password* = empty password.

The *server* parameter can also include a port number. eg. "hostname:port" or a path to a socket eg. ":/path/to/socket" for the localhost.

> **Note:** Support for ":port" was added in PHP 3.0B4.
>
> Support for ":/path/to/socket" was added in PHP 3.0.10.
>
> You can suppress the error message on failure by prepending a @ to the function name.

If a second call is made to **mysql_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned. The *new_link* parameter modifies this behavior and makes **mysql_connect()** always open a new link, even if **mysql_connect()** was called before with the same parameters. The *client_flags* parameter can be a combination of the constants MYSQL_CLIENT_SSL, MYSQL_CLIENT_COMPRESS, MYSQL_CLIENT_IGNORE_SPACE or MYSQL_CLIENT_INTERACTIVE.

> **Note:** The *new_link* parameter became available in PHP 4.2.0
>
> The *client_flags* parameter became available in PHP 4.3.0

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling **mysql_close()**.

Example 1. MySQL connect example

```
<?php
    $link = mysql_connect("localhost", "mysql_user", "mysql_password")
        or die("Could not connect: " . mysql_error());
    print ("Connected successfully");
    mysql_close($link);
?>
```

See also **mysql_pconnect()** and **mysql_close()**.

mysql_create_db

(PHP 3, PHP 4)

mysql_create_db -- Create a MySQL database

Description

bool **mysql_create_db** (string database name [, resource link_identifier])

mysql_create_db() attempts to create a new database on the server associated with the specified link identifier.

Returns **TRUE** on success or **FALSE** on failure.

Example 1. MySQL create database example

```
<?php
    $link = mysql_pconnect("localhost", "mysql_user", "mysql_password")
        or die("Could not connect: " . mysql_error());

    if (mysql_create_db("my_db")) {
        print ("Database created successfully\n");
    } else {
        printf ("Error creating database: %s\n", mysql_error());
    }
?>
```

For downwards compatibility **mysql_createdb()** can also be used. This is deprecated, however.

> **Note:** The function **mysql_create_db()** is deprecated. It is preferable to use **mysql_query()** to issue a SQL CREATE DATABASE Statement instead.

See also: **mysql_drop_db()**, **mysql_query()**.

mysql_data_seek

(PHP 3, PHP 4)

mysql_data_seek -- Move internal result pointer

Description

bool **mysql_data_seek** (resource result_identifier, int row_number)

Returns **TRUE** on success or **FALSE** on failure.

mysql_data_seek() moves the internal row pointer of the MySQL result associated with the specified result identifier to point to the specified row number. The next call to **mysql_fetch_row()** would return that row.

Row_number starts at 0. The *row_number* should be a value in the range from 0 to mysql_num_rows - 1.

> **Note:** The function **mysql_data_seek()** can be used in conjunction only with **mysql_query()**, not with **mysql_unbuffered_query()**.

Example 1. MySQL data seek example

```
<?php
    $link = mysql_pconnect("localhost", "mysql_user", "mysql_password")
        or die("Could not connect: " . mysql_error());

    mysql_select_db("samp_db")
        or die("Could not select database: " . mysql_error());

    $query = "SELECT last_name, first_name FROM friends";
    $result = mysql_query($query)
        or die("Query failed: " . mysql_error());

    /* fetch rows in reverse order */
    for ($i = mysql_num_rows($result) - 1; $i >= 0; $i--) {
        if (!mysql_data_seek($result, $i)) {
            echo "Cannot seek to row $i: " . mysql_error() . "\n";
            continue;
        }

        if(!($row = mysql_fetch_object($result)))
            continue;

        echo "$row->last_name $row->first_name<br />\n";
    }

    mysql_free_result($result);
```

```
?>
```

See also: **mysql_query()**, **mysql_num_rows()**.

mysql_db_name

(PHP 3>= 3.0.6, PHP 4)

mysql_db_name -- Get result data

Description

string **mysql_db_name** (resource result, int row [, mixed field])

mysql_db_name() takes as its first parameter the result pointer from a call to **mysql_list_dbs()**. The *row* parameter is an index into the result set.

If an error occurs, **FALSE** is returned. Use **mysql_errno()** and **mysql_error()** to determine the nature of the error.

Example 1. mysql_db_name() example

```
<?php
    error_reporting(E_ALL);

    mysql_connect('dbhost', 'username', 'password');
    $db_list = mysql_list_dbs();

    $i = 0;
    $cnt = mysql_num_rows($db_list);
    while ($i < $cnt) {
        echo mysql_db_name($db_list, $i) . "\n";
        $i++;
    }
?>
```

For backward compatibility, **mysql_dbname()** is also accepted. This is deprecated, however.

mysql_db_query

(PHP 3, PHP 4)

mysql_db_query -- Send a MySQL query

Description

resource **mysql_db_query** (string database, string query [, resource link_identifier])

Returns a positive MySQL result resource to the query result, or **FALSE** on error. The function also returns **TRUE**/**FALSE** for `INSERT`/`UPDATE`/`DELETE` queries to indicate success/failure.

mysql_db_query() selects a database and executes a query on it. If the optional link identifier isn't specified, the function will try to find an open link to the MySQL server and if no such link is found it'll try to create one as if **mysql_connect()** was called with no arguments.

Be aware that this function does **NOT** switch back to the database you were connected before. In other words, you can't use this function to *temporarily* run a sql query on another database, you would have to manually switch back. Users are strongly encouraged to use the `database.table` syntax in their sql queries instead of this function.

See also **mysql_connect()** and **mysql_query()**.

> **Note:** This function has been deprecated since PHP 4.0.6. Do not use this function. Use **mysql_select_db()** and **mysql_query()** instead.

mysql_drop_db

(PHP 3, PHP 4)

mysql_drop_db -- Drop (delete) a MySQL database

Description

bool **mysql_drop_db** (string database_name [, resource link_identifier])

Returns **TRUE** on success or **FALSE** on failure.

mysql_drop_db() attempts to drop (remove) an entire database from the server associated with the specified link identifier.

For downward compatibility **mysql_dropdb()** can also be used. This is deprecated, however.

> **Note:** The function **mysql_drop_db()** is deprecated. It is preferable to use **mysql_query()** to issue a SQL DROP DATABASE statement instead.

See also: **mysql_create_db()**, **mysql_query()**.

mysql_errno

(PHP 3, PHP 4)

mysql_errno -- Returns the numerical value of the error message from previous MySQL operation

Description

int **mysql_errno** ([resource link_identifier])

Returns the error number from the last MySQL function, or `0` (zero) if no error occurred.

Errors coming back from the MySQL database backend no longer issue warnings. Instead, use **mysql_errno()** to retrieve the error code. Note that this function only returns the error code from the most recently executed MySQL function (not including **mysql_error()** and **mysql_errno()**), so if you want to use it, make sure you check the value before calling another MySQL function.

Example 1. mysql_errno Example

```
<?php
    mysql_connect("localhost", "mysql_user", "mysql_password");

    mysql_select_db("nonexistentdb");
    echo mysql_errno() . ": " . mysql_error(). "\n";

    mysql_select_db("kossu");
    mysql_query("SELECT * FROM nonexistenttable");
    echo mysql_errno() . ": " . mysql_error() . "\n";
?>
```

The above example would produce the following output:

```
1049: Unknown database 'nonexistentdb'
1146: Table 'kossu.nonexistenttable' doesn't exist
```

> **Note:** If the optional argument is specified the given link is used to retrieve the error code. If not, the last opened link is used.

See also: **mysql_error()**

mysql_error

(PHP 3, PHP 4)

mysql_error -- Returns the text of the error message from previous MySQL operation

Description

string **mysql_error** ([resource link_identifier])

Returns the error text from the last MySQL function, or '' (the empty string) if no error occurred.

Errors coming back from the MySQL database backend no longer issue warnings. Instead, use **mysql_error()** to retrieve the error text. Note that this function only returns the error text from the most recently executed MySQL function (not including **mysql_error()** and mysql_errno()), so if you want to use it, make sure you check the value before calling another MySQL function.

Example 1. mysql_error Example

```
<?php
    mysql_connect("localhost", "mysql_user", "mysql_password");

    mysql_select_db("nonexistentdb");
    echo mysql_errno() . ": " . mysql_error(). "\n";

    mysql_select_db("kossu");
    mysql_query("SELECT * FROM nonexistenttable");
    echo mysql_errno() . ": " . mysql_error() . "\n";
?>
```

The above example would produce the following output:

```
1049: Unknown database 'nonexistentdb'
1146: Table 'kossu.nonexistenttable' doesn't exist
```

> **Note:** If the optional argument is specified the given link is used to retrieve the error message. If not, the last opened link is used.

See also: mysql_errno()

mysql_escape_string

(PHP 4 >= 4.0.3)

mysql_escape_string -- Escapes a string for use in a mysql_query.

Description

string **mysql_escape_string** (string unescaped_string)

This function will escape the *unescaped_string*, so that it is safe to place it in a mysql_query().

> **Note: mysql_escape_string()** does not escape % and _.
>
> This function is identical to mysql_real_escape_string() except that mysql_real_escape_string() takes a connection handler and escapes the string according to the current character set. **mysql_escape_string()** does not take a connection argument and does not respect the current charset setting.

Example 1. mysql_escape_string() example

```
<?php
    $item = "Zak's Laptop";
    $escaped_item = mysql_escape_string($item);
    printf ("Escaped string: %s\n", $escaped_item);
?>
```

The above example would produce the following output:

```
Escaped string: Zak\'s Laptop
```

See also: mysql_real_escape_string(), addslashes(), and the magic_quotes_gpc directive.

mysql_fetch_array

(PHP 3, PHP 4)

mysql_fetch_array -- Fetch a result row as an associative array, a numeric array, or both.

Description

array **mysql_fetch_array** (resource result [, int result_type])

Returns an array that corresponds to the fetched row, or **FALSE** if there are no more rows.

mysql_fetch_array() is an extended version of **mysql_fetch_row()**. In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you must use the numeric index of the column or make an alias for the column. For aliased columns, you cannot access the contents with the original column name (by using `'field'` in this example).

Example 1. Query with duplicate field names

```
select table1.field as foo table2.field as bar from table1, table2
```

An important thing to note is that using **mysql_fetch_array()** is *not significantly* slower than using **mysql_fetch_row()**, while it provides a significant added value.

The optional second argument *result_type* in **mysql_fetch_array()** is a constant and can take the following values: MYSQL_ASSOC, MYSQL_NUM, and MYSQL_BOTH. This feature was added in PHP 3.0.7. MYSQL_BOTH is the default for this argument.

By using MYSQL_BOTH, you'll get an array with both associative and number indices. Using MYSQL_ASSOC, you only get associative indices (as **mysql_fetch_assoc()** works), using MYSQL_NUM, you only get number indices (as **mysql_fetch_row()** works).

Example 2. mysql_fetch_array with MYSQL_NUM

```
<?php
    mysql_connect("localhost", "mysql_user", "mysql_password") or
        die("Could not connect: " . mysql_error());
    mysql_select_db("mydb");

    $result = mysql_query("SELECT id, name FROM mytable");

    while ($row = mysql_fetch_array($result, MYSQL_NUM)) {
        printf ("ID: %s  Name: %s", $row[0], $row[1]);
    }

    mysql_free_result($result);
?>
```

Example 3. mysql_fetch_array with MYSQL_ASSOC

```
<?php
    mysql_connect("localhost", "mysql_user", "mysql_password") or
        die("Could not connect: " . mysql_error());
    mysql_select_db("mydb");

    $result = mysql_query("SELECT id, name FROM mytable");

    while ($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
        printf ("ID: %s  Name: %s", $row["id"], $row["name"]);
    }

    mysql_free_result($result);
?>
```

Example 4. mysql_fetch_array with MYSQL_BOTH

```
<?php
    mysql_connect("localhost", "mysql_user", "mysql_password") or
        die("Could not connect: " . mysql_error());
    mysql_select_db("mydb");

    $result = mysql_query("SELECT id, name FROM mytable");

    while ($row = mysql_fetch_array($result, MYSQL_BOTH)) {
        printf ("ID: %s  Name: %s", $row[0], $row["name"]);
    }

    mysql_free_result($result);
?>
```

For further details, see also **mysql_fetch_row()** and **mysql_fetch_assoc()**.

mysql_fetch_assoc

(PHP 4 >= 4.0.3)

mysql_fetch_assoc -- Fetch a result row as an associative array

Description

array **mysql_fetch_assoc** (resource result)

Returns an associative array that corresponds to the fetched row, or **FALSE** if there are no more rows.

mysql_fetch_assoc() is equivalent to calling **mysql_fetch_array()** with MYSQL_ASSOC for the optional second parameter. It only returns an associative array. This is the way **mysql_fetch_array()** originally worked. If you need the numeric indices as well as the associative, use **mysql_fetch_array()**.

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you either need to access the result with numeric indices by using **mysql_fetch_row()** or add alias names. See the example at the **mysql_fetch_array()** description about aliases.

An important thing to note is that using **mysql_fetch_assoc()** is *not significantly* slower than using **mysql_fetch_row()**, while it provides a significant added value.

Example 1. An expanded mysql_fetch_assoc() example

```
<?php

    $conn = mysql_connect("localhost", "mysql_user", "mysql_password");

    if (!$conn) {
        echo "Unable to connect to DB: " . mysql_error();
        exit;
    }

    if (!mysql_select_db("mydbname")) {
        echo "Unable to select mydbname: " . mysql_error();
        exit;
    }

    $sql = "SELECT id as userid, fullname, userstatus
            FROM   sometable
            WHERE  userstatus = 1";

    $result = mysql_query($sql);

    if (!$result) {
        echo "Could not successfully run query ($sql) from DB: " . mysql_error();
        exit;
    }

    if (mysql_num_rows($result) == 0) {
        echo "No rows found, nothing to print so am exiting";
        exit;
    }

    // While a row of data exists, put that row in $row as an associative array
    // Note: If you're expecting just one row, no need to use a loop
    // Note: If you put extract($row); inside the following loop, you'll
    //       then create $userid, $fullname, and $userstatus
    while ($row = mysql_fetch_assoc($result)) {
        echo $row["userid"];
        echo $row["fullname"];
        echo $row["userstatus"];
    }

    mysql_free_result($result);

?>
```

See also **mysql_fetch_row()**, **mysql_fetch_array()**, **mysql_query()**, and **mysql_error()**.

mysql_fetch_field

(PHP 3, PHP 4)

mysql_fetch_field -- Get column information from a result and return as an object

Description

object **mysql_fetch_field** (resource result [, int field_offset])

Returns an object containing field information.

mysql_fetch_field() can be used in order to obtain information about fields in a certain query result. If the field offset isn't specified, the next field that wasn't yet retrieved by **mysql_fetch_field()** is retrieved.

The properties of the object are:

- name - column name
- table - name of the table the column belongs to
- max_length - maximum length of the column
- not_null - 1 if the column cannot be **NULL**
- primary_key - 1 if the column is a primary key
- unique_key - 1 if the column is a unique key
- multiple_key - 1 if the column is a non-unique key
- numeric - 1 if the column is numeric
- blob - 1 if the column is a BLOB
- type - the type of the column
- unsigned - 1 if the column is unsigned
- zerofill - 1 if the column is zero-filled

Example 1. mysql_fetch_field()

```
<?php
mysql_connect('localhost:3306', $user, $password)
    or die("Could not connect: " . mysql_error());
mysql_select_db("database");
$result = mysql_query("select * from table")
    or die("Query failed: " . mysql_error());
/* get column metadata */
$i = 0;
while ($i < mysql_num_fields($result)) {
    echo "Information for column $i:<br />\n";
    $meta = mysql_fetch_field($result);
    if (!$meta) {
        echo "No information available<br />\n";
    }
    echo "<pre>
blob:         $meta->blob
max_length:   $meta->max_length
multiple_key: $meta->multiple_key
name:         $meta->name
not_null:     $meta->not_null
numeric:      $meta->numeric
primary_key:  $meta->primary_key
table:        $meta->table
type:         $meta->type
unique_key:   $meta->unique_key
unsigned:     $meta->unsigned
zerofill:     $meta->zerofill
</pre>";
    $i++;
}
mysql_free_result($result);
?>
```

See also **mysql_field_seek()**.

mysql_fetch_lengths

(PHP 3, PHP 4)

mysql_fetch_lengths -- Get the length of each output in a result

Description

array **mysql_fetch_lengths** (resource result)

Returns an array that corresponds to the lengths of each field in the last row fetched by **mysql_fetch_row()**, or **FALSE** on error.

mysql_fetch_lengths() stores the lengths of each result column in the last row returned by **mysql_fetch_row()**, **mysql_fetch_array()**, and **mysql_fetch_object()** in an array, starting at offset 0.

See also: **mysql_fetch_row()**.

mysql_fetch_object

(PHP 3, PHP 4)

mysql_fetch_object -- Fetch a result row as an object

Description

object **mysql_fetch_object** (resource result)

Returns an object with properties that correspond to the fetched row, or **FALSE** if there are no more rows.

mysql_fetch_object() is similar to **mysql_fetch_array()**, with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

```
<?php

/* this is valid */
echo $row->field;
/* this is invalid */
echo $row->0;

?>
```

Speed-wise, the function is identical to **mysql_fetch_array()**, and almost as quick as **mysql_fetch_row()** (the difference is insignificant).

Example 1. mysql_fetch_object() example

```
<?php
mysql_connect("hostname", "user", "password");
mysql_select_db($db);
$result = mysql_query("select * from table");
while ($row = mysql_fetch_object($result)) {
    echo $row->user_id;
    echo $row->fullname;
}
mysql_free_result($result);
?>
```

See also: **mysql_fetch_array()** and **mysql_fetch_row()**.

mysql_fetch_row

(PHP 3, PHP 4)

mysql_fetch_row -- Get a result row as an enumerated array

Description

array **mysql_fetch_row** (resource result)

Returns an array that corresponds to the fetched row, or **FALSE** if there are no more rows.

mysql_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to **mysql_fetch_row()** would return the next row in the result set, or **FALSE** if there are no more rows.

See also: **mysql_fetch_array()**, **mysql_fetch_object()**, **mysql_data_seek()**, **mysql_fetch_lengths()**, and **mysql_result()**.

mysql_field_flags

(PHP 3, PHP 4)

mysql_field_flags -- Get the flags associated with the specified field in a result

Description

string **mysql_field_flags** (resource result, int field_offset)

mysql_field_flags() returns the field flags of the specified field. The flags are reported as a single word per flag separated by a single space, so that you can split the returned value using **explode()**.

The following flags are reported, if your version of MySQL is current enough to support them: "not_null", "primary_key", "unique_key", "multiple_key", "blob", "unsigned", "zerofill", "binary", "enum", "auto_increment", "timestamp".

For downward compatibility **mysql_fieldflags()** can also be used. This is deprecated, however.

mysql_field_len

(PHP 3, PHP 4)

mysql_field_len -- Returns the length of the specified field

Description

int **mysql_field_len** (resource result, int field_offset)

mysql_field_len() returns the length of the specified field.

For downward compatibility **mysql_fieldlen()** can also be used. This is deprecated, however.

mysql_field_name

(PHP 3, PHP 4)

mysql_field_name -- Get the name of the specified field in a result

Description

string **mysql_field_name** (resource result, int field_index)

mysql_field_name() returns the name of the specified field index. *result* must be a valid result identifier and *field_index* is the numerical offset of the field.

> **Note:** *field_index* starts at 0.
>
> e.g. The index of the third field would actually be 2, the index of the fourth field would be 3 and so on.

Example 1. mysql_field_name() example

```
/* The users table consists of three fields:
 *   user_id
 *   username
 *   password.
 */
$link = mysql_connect('localhost', "mysql_user", "mysql_password");
mysql_select_db($dbname, $link)
    or die("Could not set $dbname: " . mysql_error());
$res = mysql_query("select * from users", $link);

echo mysql_field_name($res, 0) . "\n";
echo mysql_field_name($res, 2);
```

The above example would produce the following output:

```
user_id
password
```

For downwards compatibility **mysql_fieldname()** can also be used. This is deprecated, however.

mysql_field_seek

(PHP 3, PHP 4)

mysql_field_seek -- Set result pointer to a specified field offset

Description

int **mysql_field_seek** (resource result, int field_offset)

Seeks to the specified field offset. If the next call to **mysql_fetch_field()** doesn't include a field offset, the field offset specified in **mysql_field_seek()** will be returned.

See also: **mysql_fetch_field()**.

mysql_field_table

(PHP 3, PHP 4)

mysql_field_table -- Get name of the table the specified field is in

Description

string **mysql_field_table** (resource result, int field_offset)

Returns the name of the table that the specifed field is in.

For downward compatibility **mysql_fieldtable()** can also be used. This is deprecated, however.

mysql_field_type

(PHP 3, PHP 4)

mysql_field_type -- Get the type of the specified field in a result

Description

string **mysql_field_type** (resource result, int field_offset)

mysql_field_type() is similar to the **mysql_field_name()** function. The arguments are identical, but the field type is returned instead. The field type will be one of "int", "real", "string", "blob", and others as detailed in the MySQL documentation.

Example 1. MySQL field types

```
<?php
    mysql_connect("localhost", "mysql_username", "mysql_password");
    mysql_select_db("mysql");
    $result = mysql_query("SELECT * FROM func");
    $fields = mysql_num_fields($result);
    $rows   = mysql_num_rows($result);
    $table = mysql_field_table($result, 0);
    echo "Your '".$table."' table has ".$fields." fields and ".$rows." record(s)\n";
    echo "The table has the following fields:\n";
    for ($i=0; $i < $fields; $i++) {
        $type  = mysql_field_type($result, $i);
        $name  = mysql_field_name($result, $i);
        $len   = mysql_field_len($result, $i);
        $flags = mysql_field_flags($result, $i);
        echo $type." ".$name." ".$len." ".$flags."\n";
    }
```

```
    mysql_free_result($result);
    mysql_close();
?>
```

The above example would produce the following output:

```
Your 'func' table has 4 fields and 1 record(s)
The table has the following fields:
string name 64 not_null primary_key binary
int ret 1 not_null
string dl 128 not_null
string type 9 not_null enum
```

For downward compatibility **mysql_fieldtype()** can also be used. This is deprecated, however.

mysql_free_result

(PHP 3, PHP 4)

mysql_free_result -- Free result memory

Description

bool **mysql_free_result** (resource result)

mysql_free_result() will free all memory associated with the result identifier *result*.

mysql_free_result() only needs to be called if you are concerned about how much memory is being used for queries that return large result sets. All associated result memory is automatically freed at the end of the script's execution.

Returns **TRUE** on success or **FALSE** on failure.

For downward compatibility **mysql_freeresult()** can also be used. This is deprecated, however.

mysql_get_client_info

(PHP 4 >= 4.0.5)

mysql_get_client_info -- Get MySQL client info

Description

string **mysql_get_client_info** (void)

mysql_get_client_info() returns a string that represents the client library version.

Example 1. mysql_get_client_info Example

```
<?php
    printf ("MySQL client info: %s\n", mysql_get_client_info());
?>
```

The above example would produce the following output:

```
MySQL client info: 3.23.39
```

See also: **mysql_get_host_info()**, **mysql_get_proto_info()** and **mysql_get_server_info()**.

mysql_get_host_info

(PHP 4 >= 4.0.5)

mysql_get_host_info -- Get MySQL host info

Description

string **mysql_get_host_info** ([resource link_identifier])

mysql_get_host_info() returns a string describing the type of connection in use for the connection *link_identifier*, including the server host name. If *link_identifier* is omitted, the last opened connection will be used.

Example 1. mysql_get_host_info Example

```
<?php
    mysql_connect("localhost", "mysql_user", "mysql_password") or
        die("Could not connect: " . mysql_error());
    printf ("MySQL host info: %s\n", mysql_get_host_info());
?>
```

The above example would produce the following output:

```
MySQL host info: Localhost via UNIX socket
```

See also: **mysql_get_client_info()**, **mysql_get_proto_info()** and **mysql_get_server_info()**.

mysql_get_proto_info

(PHP 4 >= 4.0.5)

mysql_get_proto_info -- Get MySQL protocol info

Description

int **mysql_get_proto_info** ([resource link_identifier])

mysql_get_proto_info() returns the protocol version used by connection *link_identifier*. If *link_identifier* is omitted, the last opened connection will be used.

Example 1. mysql_get_proto_info Example

```
<?php
    mysql_connect("localhost", "mysql_user", "mysql_password") or
        die("Could not connect: " . mysql_error());
    printf ("MySQL protocol version: %s\n", mysql_get_proto_info());
?>
```

The above example would produce the following output:

```
MySQL protocol version: 10
```

See also: **mysql_get_client_info()**, **mysql_get_host_info()** and **mysql_get_server_info()**.

mysql_get_server_info

(PHP 4 >= 4.0.5)

mysql_get_server_info -- Get MySQL server info

Description

string **mysql_get_server_info** ([resource link_identifier])

mysql_get_server_info() returns the server version used by connection *link_identifier*. If *link_identifier* is omitted, the last opened connection will be used.

Example 1. mysql_get_server_info Example

```
<?php
    mysql_connect("localhost", "mysql_user", "mysql_password") or
        die("Could not connect: " . mysql_error());
    printf ("MySQL server version: %s\n", mysql_get_server_info());
?>
```

The above example would produce the following output:

```
MySQL server version: 4.0.1-alpha
```

See also: **mysql_get_client_info()**, **mysql_get_host_info()** and **mysql_get_proto_info()**.

mysql_info

(PHP 4 >= 4.3.0)

mysql_info -- Get information about the most recent query

Description

string **mysql_info** ([resource link_identifier])

mysql_info() returns detailed information about the last query using the given *link_identifier*. If *link_identifier* isn't specified, the last opened link is assumed.

mysql_info() returns a string for all statements listed below. For all other **FALSE**. The string format depends on the given statement.

Example 1. Relevant MySQL Statements

```
INSERT INTO ... SELECT ...
String format: Records: 23 Duplicates: 0 Warnings: 0
INSERT INTO ... VALUES (...),(...),(...)...
String format: Records: 37 Duplicates: 0 Warnings: 0
LOAD DATA INFILE ...
String format: Records: 42 Deleted: 0 Skipped: 0 Warnings: 0
ALTER TABLE
String format: Records: 60 Duplicates: 0 Warnings: 0
UPDATE
String format: Rows matched: 65 Changed: 65 Warnings: 0
```

The numbers are only for illustrating purpose; their values will correspond to the query.

> **Note: mysql_info()** returns a non-**FALSE** value for the INSERT ... VALUES statement only if multiple value lists are specified in the statement.

See also: **mysql_affected_rows()**

mysql_insert_id

(PHP 3, PHP 4)

mysql_insert_id -- Get the ID generated from the previous INSERT operation

Description

int **mysql_insert_id** ([resource link_identifier])

mysql_insert_id() returns the ID generated for an AUTO_INCREMENT column by the previous INSERT query using the given *link_identifier*. If *link_identifier* isn't specified, the last opened link is assumed.

mysql_insert_id() returns 0 if the previous query does not generate an AUTO_INCREMENT value. If you need to save the value for later, be sure to call **mysql_insert_id()** immediately after the query that generates the value.

> **Note:** The value of the MySQL SQL function `LAST_INSERT_ID()` always contains the most recently generated AUTO_INCREMENT value, and is not reset between queries.

Warning

mysql_insert_id() converts the return type of the native MySQL C API function `mysql_insert_id()` to a type of `long` (named **int** in PHP). If your AUTO_INCREMENT column has a column type of BIGINT, the value returned by **mysql_insert_id()** will be incorrect. Instead, use the internal MySQL SQL function `LAST_INSERT_ID()` in an SQL query.

Example 1. mysql_insert_id Example

```
<?php
    mysql_connect("localhost", "mysql_user", "mysql_password") or
        die("Could not connect: " . mysql_error());
    mysql_select_db("mydb");

    mysql_query("INSERT INTO mytable (product) values ('kossu')");
    printf ("Last inserted record has id %d\n", mysql_insert_id());
?>
```

See also: **mysql_query()**.

mysql_list_dbs

(PHP 3, PHP 4)

mysql_list_dbs -- List databases available on a MySQL server

Description

resource **mysql_list_dbs** ([resource link_identifier])

mysql_list_dbs() will return a result pointer containing the databases available from the current mysql daemon. Use the **mysql_tablename()** function to traverse this result pointer, or any function for result tables.

Example 1. mysql_list_dbs() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$db_list = mysql_list_dbs($link);

while ($row = mysql_fetch_object($db_list)) {
    echo $row->Database . "\n";
}
?>
```

The above example would produce the following output:

```
database1
database2
database3
...
```

Note: The above code would just as easily work with **mysql_fetch_row()** or other similar functions.

For downward compatibility **mysql_listdbs()** can also be used. This is deprecated however.

See also **mysql_db_name()**.

mysql_list_fields

(PHP 3, PHP 4)

mysql_list_fields -- List MySQL result fields

Description

resource **mysql_list_fields** (string database_name, string table_name [, resource link_identifier])

mysql_list_fields() retrieves information about the given table name. Arguments are the database name and the table name. A result pointer is returned which can be used with **mysql_field_flags()**, **mysql_field_len()**, **mysql_field_name()**, and **mysql_field_type()**.

Example 1. mysql_list_fields() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');

$fields = mysql_list_fields("database1", "table1", $link);
$columns = mysql_num_fields($fields);

for ($i = 0; $i < $columns; $i++) {
    echo mysql_field_name($fields, $i) . "\n";
}
```

The above example would produce the following output:

```
field1
field2
field3
...
```

For downward compatibility **mysql_listfields()** can also be used. This is deprecated however.

mysql_list_processes

(PHP 4 >= 4.3.0)

mysql_list_processes -- List MySQL processes

Description

resource **mysql_list_processes** ([resource link_identifier])

mysql_list_processes() returns a result pointer describing the current server threads.

Example 1. mysql_list_processes() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');

$result = mysql_list_processes($link);
while ($row = mysql_fetch_row($result)){
    printf("%s %s %s %s %s\n", $row["Id"], $row["Host"], $row["db"],
       $row["Command"], $row["Time"]);
}
mysql_free_result ($result);
?>
```

The above example would produce the following output:

```
1 localhost test Processlist 0
4 localhost mysql sleep 5
```

See also: **mysql_thread_id()**

mysql_list_tables

(PHP 3, PHP 4)

mysql_list_tables -- List tables in a MySQL database

Description

resource **mysql_list_tables** (string database [, resource link_identifier])

mysql_list_tables() takes a database name and returns a result pointer much like the **mysql_query()** function. You can use the **mysql_tablename()** function to extract the actual table names from the result pointer, or any other result table function such as **mysql_fetch_assoc()**.

The *database* parameter is the name of the database to retrieve the list of tables from. Upon failure, **mysql_list_tables()** returns **FALSE**.

For downward compatibility, the function alias named **mysql_listtables()** can be used. This is deprecated however and is not recommended.

Example 1. mysql_list_tables Example

```
<?php
    $dbname = 'mysql_dbname';

    if (!mysql_connect('mysql_host', 'mysql_user', 'mysql_password')) {
        print 'Could not connect to mysql';
        exit;
    }

    $result = mysql_list_tables($dbname);

    if (!$result) {
        print "DB Error, could not list tables\n";
        print 'MySQL Error: ' . mysql_error();
        exit;
    }

    while ($row = mysql_fetch_row($result)) {
        print "Table: $row[0]\n";
    }

    mysql_free_result($result);
?>
```

See also: **mysql_list_dbs()**, and **mysql_tablename()**.

mysql_num_fields

(PHP 3, PHP 4)

mysql_num_fields -- Get number of fields in result

Description

int **mysql_num_fields** (resource result)

mysql_num_fields() returns the number of fields in a result set.

See also: **mysql_select_db()**, **mysql_query()**, **mysql_fetch_field()**, **mysql_num_rows()**.

For downward compatibility **mysql_numfields()** can also be used. This is deprecated however.

mysql_num_rows

(PHP 3, PHP 4)

mysql_num_rows -- Get number of rows in result

Description

int **mysql_num_rows** (resource result)

mysql_num_rows() returns the number of rows in a result set. This command is only valid for SELECT statements. To retrieve the number of rows affected by a INSERT, UPDATE or DELETE query, use **mysql_affected_rows()**.

Example 1. mysql_num_rows() example

```
<?php

$link = mysql_connect("localhost", "mysql_user", "mysql_password");
mysql_select_db("database", $link);

$result = mysql_query("SELECT * FROM table1", $link);
$num_rows = mysql_num_rows($result);

echo "$num_rows Rows\n";

?>
```

> **Note:** If you use **mysql_unbuffered_query()**, **mysql_num_rows()** will not return the correct value until all the rows in the result set have been retrieved.

See also: **mysql_affected_rows()**, **mysql_connect()**, **mysql_data_seek()**, **mysql_select_db()**, and **mysql_query()**.

For downward compatibility **mysql_numrows()** can also be used. This is deprecated however.

mysql_pconnect

(PHP 3, PHP 4)

mysql_pconnect -- Open a persistent connection to a MySQL server

Description

resource **mysql_pconnect** ([string server [, string username [, string password [, int client_flags]]]])

Returns a positive MySQL persistent link identifier on success, or **FALSE** on error.

mysql_pconnect() establishes a connection to a MySQL server. The following defaults are assumed for missing optional parameters: *server* = 'localhost:3306', *username* = name of the user that owns the server process and *password* = empty password. The *client_flags* parameter can be a combination of the constants MYSQL_CLIENT_SSL, MYSQL_CLIENT_COMPRESS, MYSQL_CLIENT_IGNORE_SPACE or MYSQL_CLIENT_INTERACTIVE.

The *server* parameter can also include a port number. eg. "hostname:port" or a path to a socket eg. ":/path/to/socket" for the localhost.

> **Note:** Support for ":port" was added in 3.0B4.
>
> Support for the ":/path/to/socket" was added in 3.0.10.

mysql_pconnect() acts very much like **mysql_connect()** with two major differences.

First, when connecting, the function would first try to find a (persistent) link that's already open with the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (**mysql_close()** will not close links established by **mysql_pconnect()**).

The optional *client_flags* parameter became available in PHP 4.3.0.

This type of link is therefore called 'persistent'.

> **Note:** Note, that these kind of links only work if you are using a module version of PHP. See the Persistent Database Connections section for more information.

Warning
Using persistent connections can require a bit of tuning of your Apache and MySQL configurations to ensure that you do not exceed the number of connections allowed by MySQL.

mysql_ping

(PHP 4 >= 4.3.0)

mysql_ping -- Ping a server connection or reconnect if there is no connection

Description

bool **mysql_ping** ([resource link_identifier])

mysql_ping() checks whether or not the connection to the server is working. If it has gone down, an automatic reconnection is attempted. This function can be used by scripts that remain idle for a long while, to check whether or not the server has closed the connection and reconnect if necessary. **mysql_ping()** returns **TRUE** if the connection to the server is working, otherwise **FALSE**.

See also: **mysql_thread_id()**, **mysql_list_processes()**.

mysql_query

(PHP 3, PHP 4)

mysql_query -- Send a MySQL query

Description

resource **mysql_query** (string query [, resource link_identifier [, int result_mode]])

mysql_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If *link_identifier* isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if **mysql_connect()** was called with no arguments, and use it.

The optional *result_mode* parameter can be MYSQL_USE_RESULT and MYSQL_STORE_RESULT. It defaults to MYSQL_STORE_RESULT, so the result is buffered. See also **mysql_unbuffered_query()** for the counterpart of this behaviour.

> **Note:** The query string should not end with a semicolon.

Only for SELECT,SHOW,EXPLAIN or DESCRIBE statements **mysql_query()** returns a resource identifier or **FALSE** if the query was not executed correctly. For other type of SQL statements, **mysql_query()** returns **TRUE** on success and **FALSE** on error. A non-**FALSE** return value means that the query was legal and could be executed by the server. It does not indicate anything about the number of rows affected or returned. It is perfectly possible for a query to succeed but affect no rows or return no rows.

The following query is syntactically invalid, so **mysql_query()** fails and returns **FALSE**:

Example 1. mysql_query()

```
<php
$result = mysql_query("SELECT * WHERE 1=1")
    or die("Invalid query: " . mysql_error());
?>
```

The following query is semantically invalid if `my_col` is not a column in the table `my_tbl`, so **mysql_query()** fails and returns **FALSE**:

Example 2. mysql_query()

```
<?php
$result = mysql_query("SELECT my_col FROM my_tbl")
    or die("Invalid query: " . mysql_error());
?>
```

mysql_query() will also fail and return **FALSE** if you don't have permission to access the table(s) referenced by the query.

Assuming the query succeeds, you can call **mysql_num_rows()** to find out how many rows were returned for a SELECT statment or **mysql_affected_rows()** to find out how many rows were affected by a DELETE, INSERT, REPLACE, or UPDATE statement.

Only for SELECT,SHOW,DESCRIBE or EXPLAIN statements, **mysql_query()** returns a new result identifier that you can pass to **mysql_fetch_array()** and other functions dealing with result tables. When you are done with the result set, you can free the resources associated with it by calling **mysql_free_result()**. Although, the memory will automatically be freed at the end of the script's execution.

See also: **mysql_num_rows()**, **mysql_affected_rows()**, **mysql_unbuffered_query()**, **mysql_free_result()**, **mysql_fetch_array()**, **mysql_fetch_row()**, **mysql_fetch_assoc()**, **mysql_result()**, **mysql_select_db()**, and **mysql_connect()**.

mysql_real_escape_string

(PHP 4 >= 4.3.0)

mysql_real_escape_string -- Escapes special characters in a string for use in a SQL statement, taking into account the current charset of the connection.

Description

string **mysql_real_escape_string** (string unescaped_string [, resource link_identifier])

This function will escape special characters in the *unescaped_string*, taking into account the current charset of the connection so that it is safe to place it in a **mysql_query()**.

Note: mysql_real_escape_string() does not escape % and _.

Example 1. mysql_real_escape_string() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$item = "Zak's and Derick's Laptop";
$escaped_item = mysql_real_escape_string($item);
printf ("Escaped string: %s\n", $escaped_item);
?>
```

The above example would produce the following output:

```
Escaped string: Zak\'s and Derick\'s Laptop
```

See also: **mysql_escape_string()**, **mysql_character_set_name()**.

mysql_result

(PHP 3, PHP 4)

mysql_result -- Get result data

Description

mixed **mysql_result** (resource result, int row [, mixed field])

mysql_result() returns the contents of one cell from a MySQL result set. The field argument can be the field's offset, or the field's name, or the field's table dot field name (tablename.fieldname). If the column name has been aliased ('select foo as bar from...'), use the alias instead of the column name.

When working on large result sets, you should consider using one of the functions that fetch an entire row (specified below). As these functions return the contents of multiple cells in one function call, they're MUCH quicker than **mysql_result()**. Also, note that specifying a numeric offset for the field argument is much quicker than specifying a fieldname or tablename.fieldname argument.

Calls to **mysql_result()** should not be mixed with calls to other functions that deal with the result set.

Recommended high-performance alternatives: **mysql_fetch_row()**, **mysql_fetch_array()**, and **mysql_fetch_object()**.

mysql_select_db

(PHP 3, PHP 4)

mysql_select_db -- Select a MySQL database

Description

bool **mysql_select_db** (string database_name [, resource link_identifier])

Returns **TRUE** on success or **FALSE** on failure.

mysql_select_db() sets the current active database on the server that's associated with the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if **mysql_connect()** was called without arguments, and use it.

Every subsequent call to **mysql_query()** will be made on the active database.

See also: **mysql_connect()**, **mysql_pconnect()**, and **mysql_query()**.

For downward compatibility **mysql_selectdb()** can also be used. This is deprecated however.

mysql_stat

(PHP 4 >= 4.3.0)

mysql_stat -- Get current system status

Description

string **mysql_stat** ([resource link_identifier])

mysql_stat() returns the current server status.

> **Note: mysql_stat()** currently only returns status for uptime, threads, queries, open tables, flush tables and queries per second. For a complete list of other status variables you have to use the SHOW STATUS SQL command.

Example 1. mysql_stat() example

```
<?php
$link = mysql_connect('localhost', "mysql_user", "mysql_password");
$status = explode('  ', mysql_stat($link));
print_r($status);
?>
```

The above example would produce the following output:

```
Array
(
    [0] => Uptime: 5380
    [1] => Threads: 2
    [2] => Questions: 1321299
    [3] => Slow queries: 0
    [4] => Opens: 26
    [5] => Flush tables: 1
    [6] => Open tables: 17
    [7] => Queries per second avg: 245.595
)
```

mysql_tablename

(PHP 3, PHP 4)

mysql_tablename -- Get table name of field

Description

string **mysql_tablename** (resource result, int i)

mysql_tablename() takes a result pointer returned by the **mysql_list_tables()** function as well as an integer index and returns the name of a table. The **mysql_num_rows()** function may be used to determine the number of tables in the result pointer.

Example 1. mysql_tablename() Example

```
<?php
    mysql_connect("localhost", "mysql_user", "mysql_password");
    $result = mysql_list_tables("mydb");

    for ($i = 0; $i < mysql_num_rows($result); $i++)
        printf ("Table: %s\n", mysql_tablename($result, $i));

    mysql_free_result($result);
?>
```

See also: **mysql_list_tables()**.

mysql_thread_id

(PHP 4 >= 4.3.0)

mysql_thread_id -- Return the current thread ID

Description

int **mysql_thread_id** ([resource link_identifier])

mysql_thread_id() returns the current thread ID. If the connection is lost and you reconnect with **mysql_ping()**, the thread ID will change. This means you should not get the thread ID and store it for later. You should get it when you need it.

Example 1. mysql_thread_id() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$thread_id = mysql_thread_id($link);
if ($thread_id){
    printf ("current thread id is %d\n", $thread_id);
}
?>
```

The above example would produce the following output:

```
current thread id is 73
```

See also: **mysql_ping()**, **mysql_list_processes()**.

mysql_unbuffered_query

(PHP 4 >= 4.0.6)

mysql_unbuffered_query -- Send an SQL query to MySQL, without fetching and buffering the result rows

Description

resource **mysql_unbuffered_query** (string query [, resource link_identifier [, int result_mode]])

mysql_unbuffered_query() sends a SQL query *query* to MySQL, without fetching and buffering the result rows automatically, as **mysql_query()** does. On the one hand, this saves a considerable amount of memory with SQL queries that produce large result sets. On the other hand, you can start working on the result set immediately after the first row has been retrieved: you don't have to wait until the complete SQL query has been performed. When using multiple DB-connects, you have to specify the optional parameter *link_identifier*.

The optional *result_mode* parameter can be MYSQL_USE_RESULT and MYSQL_STORE_RESULT. It defaults to MYSQL_USE_RESULT, so the result is not buffered. See also **mysql_query()** for the counterpart of this behaviour.

> **Note:** The benefits of **mysql_unbuffered_query()** come at a cost: You cannot use **mysql_num_rows()** on a result set returned from **mysql_unbuffered_query()**. You also have to fetch all result rows from an unbuffered SQL query, before you can send a new SQL query to MySQL.

See also: **mysql_query()**.

LXIV. Mohawk Software session handler functions

Introduction

msession is an interface to a high speed session daemon which can run either locally or remotely. It is designed to provide consistent session management for a PHP web farm. More Information about msession and the session server software itself can be found at http://www.mohawksoft.com/phoenix/.

> **Note:** This extension is not available on Windows platforms.

Requirements

Installation

To enable Msession support configure PHP `--with-msession[=DIR]`, where DIR is the Msession install directory.

Runtime Configuration

Resource Types

Predefined Constants

Table of Contents

msession_connect

(PHP 4 >= 4.2.0)

msession_connect -- Connect to msession server

Description

bool **msession_connect** (string host, string port)

Warning
This function is currently not documented; only the argument list is available.

msession_count

(PHP 4 >= 4.2.0)

msession_count -- Get session count

Description

int **msession_count** (void)

Warning
This function is currently not documented; only the argument list is available.

msession_create

(PHP 4 >= 4.2.0)

msession_create -- Create a session

Description

bool **msession_create** (string session)

Warning
This function is currently not documented; only the argument list is available.

msession_destroy

(PHP 4 >= 4.2.0)

msession_destroy -- Destroy a session

Description

bool **msession_destroy** (string name)

Warning
This function is currently not documented; only the argument list is available.

msession_disconnect

(PHP 4 >= 4.2.0)

msession_disconnect -- Close connection to msession server

Description

void **msession_disconnect** (void)

Warning
This function is currently not documented; only the argument list is available.

msession_find

(PHP 4 >= 4.2.0)

msession_find -- Find value

Description

array **msession_find** (string name, string value)

Warning
This function is currently not documented; only the argument list is available.

msession_get_array

(PHP 4 >= 4.2.0)

msession_get_array -- Get array of ... ?

Description

array **msession_get_array** (string session)

Warning
This function is currently not documented; only the argument list is available.

msession_get

(PHP 4 >= 4.2.0)

msession_get -- Get value from session

Description

string **msession_get** (string session, string name, string value)

Warning
This function is currently not documented; only the argument list is available.

msession_getdata

(no version information, might be only in CVS)

msession_getdata -- Get data ... ?

Description

string **msession_getdata** (string session)

Warning
This function is currently not documented; only the argument list is available.

msession_inc

(PHP 4 >= 4.2.0)

msession_inc -- Increment value in session

Description

string **msession_inc** (string session, string name)

Warning
This function is currently not documented; only the argument list is available.

msession_list

(PHP 4 >= 4.2.0)

msession_list -- List ... ?

Description

array **msession_list** (void)

Warning
This function is currently not documented; only the argument list is available.

msession_listvar

(PHP 4 >= 4.2.0)

msession_listvar -- List sessions with variable

Description

array **msession_listvar** (string name)

Returns an associative array of value, session for all sessions with a variable named *name*.

Used for searching sessions with common attributes.

msession_lock

(PHP 4 >= 4.2.0)

msession_lock -- Lock a session

Description

int **msession_lock** (string name)

Warning
This function is currently not documented; only the argument list is available.

msession_plugin

(PHP 4 >= 4.2.0)

msession_plugin -- Call an escape function within the msession personality plugin

Description

string **msession_plugin** (string session, string val [, string param])

Warning
This function is currently not documented; only the argument list is available.

msession_randstr

(PHP 4 >= 4.2.0)

msession_randstr -- Get random string

Description

string **msession_randstr** (int param)

Warning
This function is currently not documented; only the argument list is available.

msession_set_array

(PHP 4 >= 4.2.0)

msession_set_array -- Set array of ...

Description

bool **msession_set_array** (string session, array tuples)

Warning
This function is currently not documented; only the argument list is available.

msession_set

(PHP 4 >= 4.2.0)

msession_set -- Set value in session

Description

bool **msession_set** (string session, string name, string value)

Warning
This function is currently not documented; only the argument list is available.

msession_setdata

(no version information, might be only in CVS)

msession_setdata -- Set data ... ?

Description

bool **msession_setdata** (string session, string value)

Warning
This function is currently not documented; only the argument list is available.

msession_timeout

(PHP 4 >= 4.2.0)

msession_timeout -- Set/get session timeout

Description

int **msession_timeout** (string session [, int param])

Warning
This function is currently not documented; only the argument list is available.

msession_uniq

(PHP 4 >= 4.2.0)

msession_uniq -- Get uniq id

Description

string **msession_uniq** (int param)

Warning
This function is currently not documented; only the argument list is available.

msession_unlock

(PHP 4 >= 4.2.0)

msession_unlock -- Unlock a session

Description

int **msession_unlock** (string session, int key)

Warning
This function is currently not documented; only the argument list is available.

LXV. muscat functions

Introduction

Warning
This extension is *EXPERIMENTAL*. The behaviour of this extension -- including the names of its functions and anything else documented about this extension -- may change without notice in a future release of PHP. Use this extension at your own risk.

Table of Contents

muscat_close

(4.0.5 - 4.2.3 only)

muscat_close -- Shuts down the muscat session and releases any memory back to PHP.

Description

int **muscat_close** (resource muscat_handle)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

[Not back to the system, note!]

muscat_get

(4.0.5 - 4.2.3 only)

muscat_get -- Gets a line back from the core muscat API.

Description

string **muscat_get** (resource muscat_handle)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

Returns a literal FALSE when there is no more to get (as opposed to ""). Use === FALSE or !== FALSE to check for this.

muscat_give

(4.0.5 - 4.2.3 only)

muscat_give -- Sends string to the core muscat API

Description

int **muscat_give** (resource muscat_handle, string string)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

muscat_setup_net

(4.0.5 - 4.2.3 only)

muscat_setup_net -- Creates a new muscat session and returns the handle.

Description

resource **muscat_setup_net** (string muscat_host, int port)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

muscat_host is the hostname to connect to port is the port number to connect to - actually takes exactly the same args as fsockopen

muscat_setup

(4.0.5 - 4.2.3 only)

muscat_setup -- Creates a new muscat session and returns the handle.

Description

resource **muscat_setup** (int size [, string muscat_dir])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

Size is the ammount of memory in bytes to allocate for muscat muscat_dir is the muscat installation dir e.g. "/usr/local/empower", it defaults to the compile time muscat directory

LXVI. Network Functions

Introduction

Requirements

No external libraries are needed to build this extension.

Installation

There is no installation needed to use these functions; they are part of the PHP core.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. Network Configuration Options

Name	Default	Changeable
define_syslog_variables	"0"	PHP_INI_ALL

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Here is a short explanation of the configuration directives.

`define_syslog_variables` **boolean**

Whether or not to define the various syslog variables (e.g. $LOG_PID, $LOG_CRON, etc.). Turning it off is a good idea

performance-wise. At runtime, you can define these variables by calling **define_syslog_variables()**.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are always available as part of the PHP core.

Table 2. openlog() Options

Constant	Description
LOG_CONS	if there is an error while sending data to the system logger, write directly to the system console
LOG_NDELAY	open the connection to the logger immediately
LOG_ODELAY	(default) delay opening the connection until the first message is logged
LOG_PERROR	print log message also to standard error
LOG_PID	include PID with each message

Table 3. syslog() Priorities (in descending order)

Constant	Description
LOG_EMERG	system is unusable
LOG_ALERT	action must be taken immediately
LOG_CRIT	critical conditions
LOG_ERR	error conditions
LOG_WARNING	warning conditions
LOG_NOTICE	normal, but significant, condition
LOG_INFO	informational message
LOG_DEBUG	debug-level message

Table 4. dns_get_record() Options

Constant	Description
DNS_A	IPv4 Address Resource
DNS_MX	Mail Exchanger Resource
DNS_CNAME	Alias (Canonical Name) Resource
DNS_NS	Authoritative Name Server Resource
DNS_PTR	Pointer Resource
DNS_HINFO	Host Info Resource (See RFC 1010 for the meaning of these values)
DNS_SOA	Start of Authority Resource
DNS_TXT	Text Resource
DNS_ANY	Any Resource Record. On most systems this returns all resource records, however it should not be counted upon for critical uses. Try DNS_ALL instead.
DNS_AAAA	IPv6 Address Resource
DNS_ALL	Itteratively query the name server for each available record type.

Table of Contents

dns_get_record -- Fetch DNS Resource Records associated with a hostname
fsockopen -- Open Internet or Unix domain socket connection
gethostbyaddr -- Get the Internet host name corresponding to a given IP address
gethostbyname -- Get the IP address corresponding to a given Internet host name
gethostbynamel -- Get a list of IP addresses corresponding to a given Internet host name
getmxrr -- Get MX records corresponding to a given Internet host name
getprotobyname -- Get protocol number associated with protocol name
getprotobynumber -- Get protocol name associated with protocol number
getservbyname -- Get port number associated with an Internet service and protocol
getservbyport -- Get Internet service which corresponds to port and protocol
ip2long -- Converts a string containing an (IPv4) Internet Protocol dotted address into a proper address.
long2ip -- Converts an (IPv4) Internet network address into a string in Internet standard dotted format
openlog -- Open connection to system logger
pfsockopen -- Open persistent Internet or Unix domain socket connection
socket_get_status -- Alias of **stream_get_meta_data()**.
socket_set_blocking -- Alias for **stream_set_blocking()**
socket_set_timeout -- Alias for **stream_set_timeout()**
syslog -- Generate a system log message

checkdnsrr

(PHP 3, PHP 4)

checkdnsrr -- Check DNS records corresponding to a given Internet host name or IP address

Description

int **checkdnsrr** (string host [, string type])

Searches DNS for records of type *type* corresponding to *host*. Returns **TRUE** if any records are found; returns **FALSE** if no records were found or if an error occurred.

type may be any one of: A, MX, NS, SOA, PTR, CNAME, AAAA, or ANY. The default is MX.

Host may either be the IP address in dotted-quad notation or the host name.

Note: AAAA type added with PHP 4.3.0

Note: This function is not implemented on Windows platforms. Try the PEAR class Net_DNS.

See also **getmxrr()**, **gethostbyaddr()**, **gethostbyname()**, **gethostbynamel()**, and the named(8) manual page.

closelog

(PHP 3, PHP 4)

closelog -- Close connection to system logger

Description

int **closelog** (void)

closelog() closes the descriptor being used to write to the system logger. The use of **closelog()** is optional.

See also **define_syslog_variables()**, **syslog()** and **openlog()**.

debugger_off

(PHP 3)

debugger_off -- Disable internal PHP debugger (PHP 3)

Description

int **debugger_off** (void)

Disables the internal PHP debugger. This function is only available in PHP 3.

For more information see the appendix on Debugging PHP.

debugger_on

(PHP 3)

debugger_on -- Enable internal PHP debugger (PHP 3)

Description

int **debugger_on** (string address)

Enables the internal PHP debugger, connecting it to *address*. This function is only available in PHP 3.

For more information see the appendix on Debugging PHP.

define_syslog_variables

(PHP 3, PHP 4)

define_syslog_variables -- Initializes all syslog related constants

Description

void **define_syslog_variables** (void)

Initializes all constants used in the syslog functions.

See also **openlog()**, **syslog()** and **closelog()**.

dns_check_record

(PHP 5 CVS only)

dns_check_record -- Synonym for **checkdnsrr()**

Description

int **dns_check_record** (string host [, string type])

Check DNS records corresponding to a given Internet host name or IP address

dns_get_mx

(PHP 5 CVS only)

dns_get_mx -- Synonym for **getmxrr()**

Description

int **getmxrr** (string hostname, array mxhosts [, array &weight])

Get MX records corresponding to a given Internet host name.

dns_get_record

(PHP 5 CVS only)

dns_get_record -- Fetch DNS Resource Records associated with a hostname

Description

array **dns_get_record** (string hostname [, int type [, array &authns, array &addtl]])

> **Note:** This function is not implemented on Windows platforms. Try the PEAR class Net_DNS.

This function returns an array of associative arrays. Each associative array contains *at minimum* the following keys:

Table 1. Basic DNS attributes

Attribute	Meaning
host	The record in the DNS namespace to which the rest of the associated data refers.
class	**dns_get_record()** only returns Internet class records and as such this parameter will always return `IN`.
type	String containing the record type. Additional attributes will also be contained in the resulting array dependant on the value of type. See table below.
ttl	Time To Live remaining for this record. This will *not* equal the record's original ttl, but will rather equal the original ttl minus whatever length of time has passed since the authoritative name server was queried.

`hostname` should be a valid DNS hostname such as "www.example.com". Reverse lookups can be generated using in-addr.arpa notation, but **gethostbyaddr()** is more suitable for the majority of reverse lookups.

By default, **dns_get_record()** will search for any resource records associated with *`hostname`*. To limit the query, specify the optional *`type`* parameter. *`type`* may be any one of the following: **`DNS_A`**, **`DNS_CNAME`**, **`DNS_HINFO`**, **`DNS_MX`**, **`DNS_NS`**, **`DNS_PTR`**, **`DNS_SOA`**, **`DNS_TXT`**, **`DNS_AAAA`**, **`DNS_ALL`** or **`DNS_ANY`**. The default is *DNS_ANY*.

> **Note:** Because of excentricities in the performance of libresolv between platforms, **`DNS_ANY`** will not always return every record, the slower **`DNS_ALL`** will collect all records more reliably.

The optional third and fourth arguments to this function, *`authns`* and *`addtl`* are passed by reference and, if given, will be populated with Resource Records for the *Authoritative Name Servers*, and any *Additional Records* respectively. See the example below.

Table 2. Other keys in associative arrays dependant on 'type'

Type	Extra Columns
A	`ip`: An IPv4 addresses in dotted decimal notation.
MX	`pri`: Priority of mail exchanger. Lower numbers indicate greater priority. `target`: FQDN of the mail exchanger. See also **dns_get_mx()**.
CNAME	`target`: FQDN of location in DNS namespace to which the record is aliased.
NS	`target`: FQDN of the name server which is authoritative for this hostname.
PTR	`target`: Location within the DNS namespace to which this record points.
TXT	`txt`: Arbitrary string data associated with this record.
HINFO	`cpu`: IANA number designating the CPU of the machine referenced by this record. `os`: IANA number designating the Operating System on the machine referenced by this record. See RFC 1010 for the meaning of these values.
SOA	`mname`: FQDN of the machine from which the resource records orignated. `rname`: Email address of the administrative contain for this domain. `serial`: Serial # of this revision of the requested domain. `refresh`: Refresh interval (seconds) secondary name servers should use when updating remote copies of this domain. `retry`: Length of time (seconds) to wait after a failed refresh before making a second attempt. `expire`: Maximum length of time (seconds) a secondary DNS server should retain remote copies of the zone data without a successful refresh before discarding. `minimum-ttl`: Minimum length of time (seconds) a client can continue to use a DNS resolution before it should request a new resolution from the server. Can be overridden by individual resource records.
AAAA	`ipv6`: IPv6 address

> **Note:** Per DNS standards, email addresses are given in user.host format (for example: hostmaster.example.com as opposed to hostmaster@example.com), be sure to check this value and modify if necessary before using it with a functions such as **mail()**.

Example 1. Using dns_get_record()

```
<?php
$result = dns_get_record("php.net");
print_r($result);
?>

/*
Produces ouput similar to the following:
----------------------------------------

Array
(
    [0] => Array
        (
            [host] => php.net
            [type] => MX
            [pri] => 5
            [target] => pair2.php.net
            [class] => IN
            [ttl] => 6765
        )

    [1] => Array
        (
            [host] => php.net
            [type] => A
            [ip] => 64.246.30.37
            [class] => IN
            [ttl] => 8125
        )

)
*/
```

Since it's very common to want the IP address of a mail server once the MX record has been resolved, **dns_get_record()** also returns an array in *addtl* which contains associate records. *authns* is returned as well conatining a list of authoritative name servers.

Example 2. Using dns_get_record() and DNS_ANY

```
<?php
/* Request "ANY" record for php.net,
   and create $authns and $addtl arrays
   containing list of name servers and
   any additional records which go with
   them */
$result = dns_get_record("php.net",DNS_ANY,$authns,$addtl);
print "Result = ";
print_r($result);
print "Auth NS = ";
print_r($authns);
print "Additional = ";
print_r($addtl);
?>

/*
Produces output similar to the following:
----------------------------------------

Result = Array
(
    [0] => Array
        (
            [host] => php.net
            [type] => MX
            [pri] => 5
            [target] => pair2.php.net
            [class] => IN
            [ttl] => 6765
        )

    [1] => Array
        (
            [host] => php.net
            [type] => A
            [ip] => 64.246.30.37
            [class] => IN
            [ttl] => 8125
        )

)
Auth NS = Array
(
    [0] => Array
        (
            [host] => php.net
            [type] => NS
            [target] => remote1.easydns.com
            [class] => IN
            [ttl] => 10722
```

```
        )

    [1] => Array
        (
            [host] => php.net
            [type] => NS
            [target] => remote2.easydns.com
            [class] => IN
            [ttl] => 10722
        )

    [2] => Array
        (
            [host] => php.net
            [type] => NS
            [target] => ns1.easydns.com
            [class] => IN
            [ttl] => 10722
        )

    [3] => Array
        (
            [host] => php.net
            [type] => NS
            [target] => ns2.easydns.com
            [class] => IN
            [ttl] => 10722
        )

)
Additional = Array
(
    [0] => Array
        (
            [host] => pair2.php.net
            [type] => A
            [ip] => 216.92.131.5
            [class] => IN
            [ttl] => 6766
        )

    [1] => Array
        (
            [host] => remote1.easydns.com
            [type] => A
            [ip] => 64.39.29.212
            [class] => IN
            [ttl] => 100384
        )

    [2] => Array
        (
            [host] => remote2.easydns.com
            [type] => A
            [ip] => 212.100.224.80
            [class] => IN
            [ttl] => 81241
        )

    [3] => Array
        (
            [host] => ns1.easydns.com
            [type] => A
            [ip] => 216.220.40.243
            [class] => IN
            [ttl] => 81241
        )

    [4] => Array
        (
            [host] => ns2.easydns.com
            [type] => A
            [ip] => 216.220.40.244
            [class] => IN
            [ttl] => 81241
        )

)
*/
```

See also **dns_get_mx()**, and **dns_check_record()**

fsockopen

(PHP 3, PHP 4)

fsockopen -- Open Internet or Unix domain socket connection

Description

int **fsockopen** (string hostname, int port [, int errno [, string errstr [, float timeout]]])

Initiates a stream connection in the Internet (AF_INET, using TCP or UDP) or Unix (AF_UNIX) domain. For the Internet domain, it will open a TCP socket connection to *hostname* on port *port*. *hostname* may in this case be either a fully qualified domain name or an IP address. For UDP connections, you need to explicitly specify the protocol by prefixing *hostname* with 'udp://'. For the Unix domain, *hostname* will be used as the path to the socket, *port* must be set to 0 in this case. The optional *timeout* can be used to set a timeout in seconds for the connect system call.

> **Note:** If you need to set a timeout for reading/writing data over the socket, use **socket_set_timeout()**, as the *timeout* parameter to **fsockopen()** only applies while connecting the socket.

As of PHP 4.3.0, if you have compiled in OpenSSL support, you may prefix the *hostname* with either 'ssl://' or 'tls://' to use an SSL or TLS client connection over TCP/IP to connect to the remote host.

fsockopen() returns a file pointer which may be used together with the other file functions (such as **fgets()**, **fgetss()**, **fputs()**, **fclose()**, and **feof()**).

If the call fails, it will return **FALSE** and if the optional *errno* and *errstr* arguments are present they will be set to indicate the actual system level error that occurred in the system-level `connect()` call. If the value returned in *errno* is `0` and the function returned **FALSE**, it is an indication that the error occurred before the `connect()` call. This is most likely due to a problem initializing the socket. Note that the *errno* and *errstr* arguments will always be passed by reference.

Depending on the environment, the Unix domain or the optional connect timeout may not be available.

The socket will by default be opened in blocking mode. You can switch it to non-blocking mode by using **socket_set_blocking()**.

Example 1. fsockopen() Example

```
<?php
$fp = fsockopen ("www.example.com", 80, $errno, $errstr, 30);
if (!$fp) {
    echo "$errstr ($errno)<br>\n";
} else {
    fputs ($fp, "GET / HTTP/1.0\r\nHost: www.example.com\r\n\r\n");
    while (!feof($fp)) {
        echo fgets ($fp,128);
    }
    fclose ($fp);
}
?>
```

The example below shows how to retrieve the day and time from the UDP service "daytime" (port 13) in your own machine.

Example 2. Using UDP connection

```
<?php
$fp = fsockopen("udp://127.0.0.1", 13, $errno, $errstr);
if (!$fp) {
    echo "ERROR: $errno - $errstr<br>\n";
} else {
    fwrite($fp,"\n");
    echo fread($fp, 26);
    fclose($fp);
}
?>
```

Warning
UDP sockets will sometimes appear to have opened without an error, even if the remote host is unreachable. The error will only become apparent when you read or write data to/from the socket. The reason for this is because UDP is a "connectionless" protocol, which means that the operating system does not try to establish a link for the socket until it actually needs to send or receive data.

> **Note:** The timeout parameter was introduced in PHP 3.0.9 and UDP support was added in PHP 4.

See also **pfsockopen()**, **socket_set_blocking()**, **socket_set_timeout()**, **fgets()**, **fgetss()**, **fputs()**, **fclose()**, **feof()**, and the Curl extension.

gethostbyaddr

(PHP 3, PHP 4)

gethostbyaddr -- Get the Internet host name corresponding to a given IP address

Description

string **gethostbyaddr** (string ip_address)

Returns the host name of the Internet host specified by *ip_address* or a string containing the unmodified *ip_address* on failure.

See also **gethostbyname()**.

gethostbyname

(PHP 3, PHP 4)

gethostbyname -- Get the IP address corresponding to a given Internet host name

Description

string **gethostbyname** (string hostname)

Returns the IP address of the Internet host specified by *hostname* or a string containing the unmodified *hostname* on failure.

See also **gethostbyaddr()**.

gethostbynamel

(PHP 3, PHP 4)

gethostbynamel -- Get a list of IP addresses corresponding to a given Internet host name

Description

array **gethostbynamel** (string hostname)

Returns a list of IP addresses to which the Internet host specified by *hostname* resolves.

See also **gethostbyname()**, **gethostbyaddr()**, **checkdnsrr()**, **getmxrr()**, and the `named(8)` manual page.

getmxrr

(PHP 3, PHP 4)

getmxrr -- Get MX records corresponding to a given Internet host name

Description

int **getmxrr** (string hostname, array mxhosts [, array weight])

Searches DNS for MX records corresponding to *hostname*. Returns **TRUE** if any records are found; returns **FALSE** if no records were found or if an error occurred.

A list of the MX records found is placed into the array *mxhosts*. If the *weight* array is given, it will be filled with the weight information gathered.

Note: This function is not implemented on Windows platforms. Try the PEAR class Net_DNS.

See also **checkdnsrr()**, **gethostbyname()**, **gethostbynamel()**, **gethostbyaddr()**, and the `named(8)` manual page.

getprotobyname

(PHP 4)

getprotobyname -- Get protocol number associated with protocol name

Description

int **getprotobyname** (string name)

getprotobyname() returns the protocol number associated with the protocol *name* as per `/etc/protocols`.

See also: **getprotobynumber()**.

getprotobynumber

(PHP 4)

getprotobynumber -- Get protocol name associated with protocol number

Description

string **getprotobynumber** (int number)

getprotobynumber() returns the protocol name associated with protocol *number* as per `/etc/protocols`.

See also: **getprotobyname()**.

getservbyname

(PHP 4)

getservbyname -- Get port number associated with an Internet service and protocol

Description

int **getservbyname** (string service, string protocol)

getservbyname() returns the Internet port which corresponds to *service* for the specified *protocol* as per `/etc/services`. *protocol* is either `"tcp"` or `"udp"` (in lowercase).

See also: **getservbyport()**.

getservbyport

(PHP 4)

getservbyport -- Get Internet service which corresponds to port and protocol

Description

string **getservbyport** (int port, string protocol)

getservbyport() returns the Internet service associated with *port* for the specified *protocol* as per `/etc/services`. *protocol* is either `"tcp"` or `"udp"` (in lowercase).

See also: **getservbyname()**.

ip2long

(PHP 4)

ip2long -- Converts a string containing an (IPv4) Internet Protocol dotted address into a proper address.

Description

int **ip2long** (string ip_address)

The function **ip2long()** generates an IPv4 Internet network address from its Internet standard format (dotted string) representation.

Example 1. ip2long() Example

```
<?php
$ip = gethostbyname("www.example.com");
$out = "The following URLs are equivalent:<br>\n";
$out .= "http://www.example.com/, http://".$ip."/, and http://".sprintf("%u",ip2long($ip))."/<br>\n";
echo $out;
?>
```

> **Note:** Because PHP's integer type is signed, and many IP addresses will result in negative integers, you need to use the "%u" formatter of **sprintf()** or **printf()** to get the string representation of the unsigned IP address.

This second example shows how to print a converted address with the **printf()** function :

Example 2. Displaying an IP address

```
<?php
$ip = gethostbyname("www.example.com");
printf("%u\n", ip2long($ip));
echo $out;
?>
```

See also: **long2ip()**

long2ip

(PHP 4)

long2ip -- Converts an (IPv4) Internet network address into a string in Internet standard dotted format

Description

string **long2ip** (int proper_address)

The function **long2ip()** generates an Internet address in dotted format (i.e.: aaa.bbb.ccc.ddd) from the proper address representation.

See also: **ip2long()**

openlog

(PHP 3, PHP 4)

openlog -- Open connection to system logger

Description

int **openlog** (string ident, int option, int facility)

openlog() opens a connection to the system logger for a program. The string *ident* is added to each message. Values for *option* and *facility* are given below. The *option* argument is used to indicate what logging options will be used when generating a log message. The *facility* argument is used to specify what type of program is logging the message. This allows you to specify (in your machine's syslog configuration) how messages coming from different facilities will be handled. The use of **openlog()** is optional. It will automatically be called by **syslog()** if necessary, in which case *ident* will default to FALSE.

Table 1. openlog() Options

Constant	Description
LOG_CONS	if there is an error while sending data to the system logger, write directly to the system console
LOG_NDELAY	open the connection to the logger immediately
LOG_ODELAY	(default) delay opening the connection until the first message is logged

Constant	Description
LOG_PERROR	print log message also to standard error
LOG_PID	include PID with each message

You can use one or more of this options. When using multiple options you need to OR them, i.e. to open the connection immediately, write to the console and include the PID in each message, you will use: `LOG_CONS | LOG_NDELAY | LOG_PID`

Table 2. openlog() Facilities

Constant	Description
LOG_AUTH	security/authorization messages (use LOG_AUTHPRIV instead in systems where that constant is defined)
LOG_AUTHPRIV	security/authorization messages (private)
LOG_CRON	clock daemon (cron and at)
LOG_DAEMON	other system daemons
LOG_KERN	kernel messages
LOG_LOCAL0 ... LOG_LOCAL7	reserved for local use, these are not available in Windows
LOG_LPR	line printer subsystem
LOG_MAIL	mail subsystem
LOG_NEWS	USENET news subsystem
LOG_SYSLOG	messages generated internally by syslogd
LOG_USER	generic user-level messages
LOG_UUCP	UUCP subsystem

See also **define_syslog_variables()**, **syslog()** and **closelog()**.

pfsockopen

(PHP 3>= 3.0.7, PHP 4)

pfsockopen -- Open persistent Internet or Unix domain socket connection

Description

int **pfsockopen** (string hostname, int port [, int errno [, string errstr [, int timeout]]])

This function behaves exactly as **fsockopen()** with the difference that the connection is not closed after the script finishes. It is the persistent version of **fsockopen()**.

socket_get_status

socket_get_status -- Alias of **stream_get_meta_data()**.

Description

This function is an alias of **stream_get_meta_data()**.

socket_set_blocking

socket_set_blocking -- Alias for **stream_set_blocking()**

Description

This function is an alias for **stream_set_blocking()**.

socket_set_timeout

socket_set_timeout -- Alias for **stream_set_timeout()**

Description

This is an alias for **stream_set_timeout()**.

syslog

(PHP 3, PHP 4)

syslog -- Generate a system log message

Description

int **syslog** (int priority, string message)

syslog() generates a log message that will be distributed by the system logger. *priority* is a combination of the facility and the level, values for which are given in the next section. The remaining argument is the message to send, except that the two characters %m will be replaced by the error message string (strerror) corresponding to the present value of errno.

Table 1. syslog() Priorities (in descending order)

Constant	Description
LOG_EMERG	system is unusable
LOG_ALERT	action must be taken immediately
LOG_CRIT	critical conditions
LOG_ERR	error conditions
LOG_WARNING	warning conditions
LOG_NOTICE	normal, but significant, condition
LOG_INFO	informational message
LOG_DEBUG	debug-level message

Example 1. Using syslog()

```
<?php
define_syslog_variables();
// open syslog, include the process ID and also send
// the log to standard error, and use a user defined
// logging mechanism
openlog("myScripLog", LOG_PID | LOG_PERROR, LOG_LOCAL0);

// some code

if (authorized_client()) {
    // do something
} else {
    // unauthorized client!
    // log the attempt
    $access = date("Y/m/d H:i:s");
    syslog(LOG_WARNING,"Unauthorized client: $access $REMOTE_ADDR ($HTTP_USER_AGENT)");
}

closelog();
?>
```

For information on setting up a user defined log handler, see the syslog.conf(5) Unix manual page. More information on the syslog facilities and option can be found in the man pages for syslog(3) on Unix machines.

On Windows NT, the syslog service is emulated using the Event Log.

> **Note:** Use of LOG_LOCAL0 through LOG_LOCAL7 for the *facility* parameter of **openlog()** is not available in Windows.

See also **define_syslog_variables()**, **openlog()** and **closelog()**.

LXVII. Ncurses terminal screen control functions

Introduction

ncurses (new curses) is a free software emulation of curses in System V Rel 4.0 (and above). It uses terminfo format, supports pads, colors, multiple highlights, form characters and function key mapping. Because of the interactive nature of this library, it will be of little use for writing Web applications, but may be useful when writing scripts meant using PHP from the command line.

Warning
This extension is *EXPERIMENTAL*. The behaviour of this extension -- including the names of its functions and anything else documented about this extension -- may change without notice in a future release of PHP. Use this extension at your own risk.

Ncurses is available for the following platforms:

- AIX
- BeOS
- Cygwin
- Digital Unix (aka OSF1)
- FreeBSD
- GNU/Linux
- HPUX
- IRIX
- OS/2
- SCO OpenServer
- Solaris
- SunOS

Requirements

You need the ncurses libraries and headerfiles. Download the latest version from the ftp://ftp.gnu.org/pub/gnu/ncurses/ or from an other GNU-Mirror.

Installation

To get these functions to work, you have to compile the CGI or CLI version of PHP with `--with-ncurses[=DIR]`.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. Ncurses configuration options

Name	Default	Changeable
ncurses.value	"42"	PHP_INI_ALL
ncurses.string	"foobar"	PHP_INI_ALL

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

Error codes

On error ncurses functions return NCURSES_ERR.

Colors

Table 2. ncurses color constants

constant	meaning
NCURSES_COLOR_BLACK	no color (black)
NCURSES_COLOR_WHITE	white
NCURSES_COLOR_RED	red - supported when terminal is in color mode
NCURSES_COLOR_GREEN	green - supported when terminal is in color mod
NCURSES_COLOR_YELLOW	yellow - supported when terminal is in color mod
NCURSES_COLOR_BLUE	blue - supported when terminal is in color mod
NCURSES_COLOR_CYAN	cyan - supported when terminal is in color mod
NCURSES_COLOR_MAGENTA	magenta - supported when terminal is in color mod

Keys

Table 3. ncurses key constants

constant	meaning
NCURSES_KEY_Fo - NCURSES_KEY_F64	function keys F1 - F64
NCURSES_KEY_DOWN	down arrow
NCURSES_KEY_UP	up arrow
NCURSES_KEY_LEFT	left arrow
NCURSES_KEY_RIGHT	right arrow
NCURSES_KEY_HOME	home key (upward+left arrow)
NCURSES_KEY_BACKSPACE	backspace
NCURSES_KEY_DL	delete line
NCURSES_KEY_IL	insert line
NCURSES_KEY_DC	delete character
NCURSES_KEY_IC	insert char or enter insert mode
NCURSES_KEY_EIC	exit insert char mode
NCURSES_KEY_CLEAR	clear screen
NCURSES_KEY_EOS	clear to end of screen
NCURSES_KEY_EOL	clear to end of line
NCURSES_KEY_SF	scroll one line forward
NCURSES_KEY_SR	scroll one line backward
NCURSES_KEY_NPAGE	next page
NCURSES_KEY_PPAGE	previous page
NCURSES_KEY_STAB	set tab
NCURSES_KEY_CTAB	clear tab

constant	meaning
NCURSES_KEY_CATAB	clear all tabs
NCURSES_KEY_SRESET	soft (partial) reset
NCURSES_KEY_RESET	reset or hard reset
NCURSES_KEY_PRINT	print
NCURSES_KEY_LL	lower left
NCURSES_KEY_A1	upper left of keypad
NCURSES_KEY_A3	upper right of keypad
NCURSES_KEY_B2	center of keypad
NCURSES_KEY_C1	lower left of keypad
NCURSES_KEY_C3	lower right of keypad
NCURSES_KEY_BTAB	back tab
NCURSES_KEY_BEG	beginning
NCURSES_KEY_CANCEL	cancel
NCURSES_KEY_CLOSE	close
NCURSES_KEY_COMMAND	cmd (command)
NCURSES_KEY_COPY	copy
NCURSES_KEY_CREATE	create
NCURSES_KEY_END	end
NCURSES_KEY_EXIT	exit
NCURSES_KEY_FIND	find
NCURSES_KEY_HELP	help
NCURSES_KEY_MARK	mark
NCURSES_KEY_MESSAGE	message
NCURSES_KEY_MOVE	move
NCURSES_KEY_NEXT	next
NCURSES_KEY_OPEN	open
NCURSES_KEY_OPTIONS	options
NCURSES_KEY_PREVIOUS	previous
NCURSES_KEY_REDO	redo
NCURSES_KEY_REFERENCE	ref (reference)
NCURSES_KEY_REFRESH	refresh
NCURSES_KEY_REPLACE	replace
NCURSES_KEY_RESTART	restart
NCURSES_KEY_RESUME	resume
NCURSES_KEY_SAVE	save
NCURSES_KEY_SBEG	shiftet beg (beginning)
NCURSES_KEY_SCANCEL	shifted cancel
NCURSES_KEY_SCOMMAND	shifted command
NCURSES_KEY_SCOPY	shifted copy
NCURSES_KEY_SCREATE	shifted create
NCURSES_KEY_SDC	shifted delete char
NCURSES_KEY_SDL	shifted delete line
NCURSES_KEY_SELECT	select
NCURSES_KEY_SEND	shifted end
NCURSES_KEY_SEOL	shifted end of line
NCURSES_KEY_SEXIT	shifted exit
NCURSES_KEY_SFIND	shifted find
NCURSES_KEY_SHELP	shifted help
NCURSES_KEY_SHOME	shifted home
NCURSES_KEY_SIC	shifted input
NCURSES_KEY_SLEFT	shifted left arrow

constant	meaning
NCURSES_KEY_SMESSAGE	shifted message
NCURSES_KEY_SMOVE	shifted move
NCURSES_KEY_SNEXT	shifted next
NCURSES_KEY_SOPTIONS	shifted options
NCURSES_KEY_SPREVIOUS	shifted previous
NCURSES_KEY_SPRINT	shifted print
NCURSES_KEY_SREDO	shifted redo
NCURSES_KEY_SREPLACE	shifted replace
NCURSES_KEY_SRIGHT	shifted right arrow
NCURSES_KEY_SRSUME	shifted resume
NCURSES_KEY_SSAVE	shifted save
NCURSES_KEY_SSUSPEND	shifted suspend
NCURSES_KEY_UNDO	undo
NCURSES_KEY_MOUSE	mouse event has occured
NCURSES_KEY_MAX	maximum key value

Mouse

Table 4. mouse constants

Constant	meaning
NCURSES_BUTTON1_RELEASED - NCURSES_BUTTON4_RELEASED	button (1-4) released
NCURSES_BUTTON1_PRESSED - NCURSES_BUTTON4_PRESSED	button (1-4) pressed
NCURSES_BUTTON1_CLICKED - NCURSES_BUTTON4_CLICKED	button (1-4) clicked
NCURSES_BUTTON1_DOUBLE_CLICKED - NCURSES_BUTTON4_DOUBLE_CLICKED	button (1-4) double clicked
NCURSES_BUTTON1_TRIPLE_CLICKED - NCURSES_BUTTON4_TRIPLE_CLICKED	button (1-4) triple clicked
NCURSES_BUTTON_CTRL	ctrl pressed during click
NCURSES_BUTTON_SHIFT	shift pressed during click
NCURSES_BUTTON_ALT	alt pressed during click
NCURSES_ALL_MOUSE_EVENTS	report all mouse events
NCURSES_REPORT_MOUSE_POSITION	report mouse position

Table of Contents

ncurses_delch -- Delete character at current position, move rest of line left
ncurses_deleteln -- Delete line at current position, move rest of screen up
ncurses_delwin -- Delete a ncurses window
ncurses_doupdate -- Write all prepared refreshes to terminal
ncurses_echo -- Activate keyboard input echo
ncurses_echochar -- Single character output including refresh
ncurses_end -- Stop using ncurses, clean up the screen
ncurses_erase -- Erase terminal screen
ncurses_erasechar -- Returns current erase character
ncurses_filter --
ncurses_flash -- Flash terminal screen (visual bell)
ncurses_flushinp -- Flush keyboard input buffer
ncurses_getch -- Read a character from keyboard
ncurses_getmouse -- Reads mouse event
ncurses_halfdelay -- Put terminal into halfdelay mode
ncurses_has_colors -- Check if terminal has colors
ncurses_has_ic -- Check for insert- and delete-capabilities
ncurses_has_il -- Check for line insert- and delete-capabilities
ncurses_has_key -- Check for presence of a function key on terminal keyboard
ncurses_hline -- Draw a horizontal line at current position using an attributed character and max. n characters long
ncurses_inch -- Get character and attribute at current position
ncurses_init_color -- Set new RGB value for color
ncurses_init_pair -- Allocate a color pair
ncurses_init -- Initialize ncurses
ncurses_insch -- Insert character moving rest of line including character at current position
ncurses_insdelln -- Insert lines before current line scrolling down (negative numbers delete and scroll up)
ncurses_insertln -- Insert a line, move rest of screen down
ncurses_insstr -- Insert string at current position, moving rest of line right
ncurses_instr -- Reads string from terminal screen
ncurses_isendwin -- Ncurses is in endwin mode, normal screen output may be performed
ncurses_keyok -- Enable or disable a keycode
ncurses_killchar -- Returns current line kill character
ncurses_longname -- Returns terminals description
ncurses_mouseinterval -- Set timeout for mouse button clicks
ncurses_mousemask -- Sets mouse options
ncurses_move -- Move output position
ncurses_mvaddch -- Move current position and add character
ncurses_mvaddchnstr -- Move position and add attrributed string with specified length
ncurses_mvaddchstr -- Move position and add attributed string
ncurses_mvaddnstr -- Move position and add string with specified length
ncurses_mvaddstr -- Move position and add string
ncurses_mvcur -- Move cursor immediately
ncurses_mvdelch -- Move position and delete character, shift rest of line left
ncurses_mvgetch -- Move position and get character at new position
ncurses_mvhline -- Set new position and draw a horizontal line using an attributed character and max. n characters long
ncurses_mvinch -- Move position and get attributed character at new position
ncurses_mvvline -- Set new position and draw a vertical line using an attributed character and max. n characters long
ncurses_mvwaddstr -- Add string at new position in window
ncurses_napms -- Sleep
ncurses_newwin -- Create a new window
ncurses_nl -- Translate newline and carriage return / line feed
ncurses_nocbreak -- Switch terminal to cooked mode
ncurses_noecho -- Switch off keyboard input echo
ncurses_nonl -- Do not translate newline and carriage return / line feed
ncurses_noqiflush -- Do not flush on signal characters
ncurses_noraw -- Switch terminal out of raw mode
ncurses_putp --
ncurses_qiflush -- Flush on signal characters
ncurses_raw -- Switch terminal into raw mode
ncurses_refresh -- Refresh screen
ncurses_resetty -- Restores saved terminal state
ncurses_savetty -- Saves terminal state
ncurses_scr_dump -- Dump screen content to file
ncurses_scr_init -- Initialize screen from file dump
ncurses_scr_restore -- Restore screen from file dump
ncurses_scr_set -- Inherit screen from file dump
ncurses_scrl -- Scroll window content up or down without changing current position
ncurses_slk_attr -- Returns current soft label key attribute
ncurses_slk_attroff --
ncurses_slk_attron --

ncurses_addch

(PHP 4 >= 4.1.0)

ncurses_addch -- Add character at current position and advance cursor

Description

int **ncurses_addch** (int ch)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_addchnstr

(PHP 4 >= 4.2.0)

ncurses_addchnstr -- Add attributed string with specified length at current position

Description

int **ncurses_addchnstr** (string s, int n)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_addchstr

(PHP 4 >= 4.2.0)

ncurses_addchstr -- Add attributed string at current position

Description

int **ncurses_addchstr** (string s)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_addnstr

(PHP 4 >= 4.2.0)

ncurses_addnstr -- Add string with specified length at current position

Description

int **ncurses_addnstr** (string s, int n)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_addstr

(PHP 4 >= 4.2.0)

ncurses_addstr -- Output text at current position

Description

int **ncurses_addstr** (string text)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_assume_default_colors

(PHP 4 >= 4.2.0)

ncurses_assume_default_colors -- Define default colors for color o

Description

int **ncurses_assume_default_colors** (int fg, int bg)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_attroff

(PHP 4 >= 4.1.0)

ncurses_attroff -- Turn off the given attributes

Description

int **ncurses_attroff** (int attributes)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_attron

(PHP 4 >= 4.1.0)

ncurses_attron -- Turn on the given attributes

Description

int **ncurses_attron** (int attributes)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_attrset

(PHP 4 >= 4.1.0)

ncurses_attrset -- Set given attributes

Description

int **ncurses_attrset** (int attributes)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_baudrate

(PHP 4 >= 4.1.0)

ncurses_baudrate -- Returns baudrate of terminal

Description

int **ncurses_baudrate** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_beep

(PHP 4 >= 4.1.0)

ncurses_beep -- Let the terminal beep

Description

int **ncurses_beep** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_beep() sends an audlible alert (bell) and if its not possible flashes the screen. Returns **FALSE** on success, otherwise **TRUE**.

See also: **ncurses_flash()**

ncurses_bkgd

(PHP 4 >= 4.1.0)

ncurses_bkgd -- Set background property for terminal screen

Description

int **ncurses_bkgd** (int attrchar)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_bkgdset

(PHP 4 >= 4.1.0)

ncurses_bkgdset -- Control screen background

Description

void **ncurses_bkgdset** (int attrchar)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_border

(PHP 4 >= 4.2.0)

ncurses_border -- Draw a border around the screen using attributed characters

Description

int **ncurses_border** (int left, int right, int top, int bottom, int tl_corner, int tr_corner, int bl_corner, int br_corner)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_can_change_color

(PHP 4 >= 4.1.0)

ncurses_can_change_color -- Check if we can change terminals colors

Description

bool **ncurses_can_change_color** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

The function **ncurses_can_change_color()** returns **TRUE** or **FALSE**, depending on whether the terminal has color capabilities and whether the programmer can change the colors.

ncurses_cbreak

(PHP 4 >= 4.1.0)

ncurses_cbreak -- Switch of input buffering

Description

bool **ncurses_cbreak** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_cbreak() disables line buffering and character processing (interrupt and flow control characters are unaffected), making characters typed by the user immediately available to the program.

ncurses_cbreak() returns **TRUE** or NCURSES_ERR if any error occured.

See also: **ncurses_nocbreak()**

ncurses_clear

(PHP 4 >= 4.1.0)

ncurses_clear -- Clear screen

Description

bool **ncurses_clear** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_clear() clears the screen completely without setting blanks. Returns **FALSE** on success, otherwise **TRUE**.

Note: **ncurses_clear()** clears the screen without setting blanks, which have the current background rendition. To clear screen with blanks, use **ncurses_erase()**.

See also: **ncurses_erase()**

ncurses_clrtobot

(PHP 4 >= 4.1.0)

ncurses_clrtobot -- Clear screen from current position to bottom

Description

bool **ncurses_clrtobot** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_clrtobot() erases all lines from cursor to end of screen and creates blanks. Blanks created by **ncurses_clrtobot()** have the current background rendition. Returns **TRUE** if any error occured, otherwise **FALSE**.

See also: **ncurses_clear()**, **ncurses_clrtoeol()**

ncurses_clrtoeol

(PHP 4 >= 4.1.0)

ncurses_clrtoeol -- Clear screen from current position to end of line

Description

bool **ncurses_clrtoeol** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_clrtoeol() erases the current line from cursor position to the end. Blanks created by **ncurses_clrtoeol()** have the current background rendition. Returns **TRUE** if any error occured, otherwise **FALSE**.

See also: **ncurses_clear()**, **ncurses_clrtobot()**

ncurses_color_set

(PHP 4 >= 4.1.0)

ncurses_color_set -- Set fore- and background color

Description

int **ncurses_color_set** (int pair)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_curs_set

(PHP 4 >= 4.1.0)

ncurses_curs_set -- Set cursor state

Description

int **ncurses_curs_set** (int visibility)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_def_prog_mode

(PHP 4 >= 4.1.0)

ncurses_def_prog_mode -- Saves terminals (program) mode

Description

bool **ncurses_def_prog_mode** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_def_prog_mode() saves the current terminal modes for program (in curses) for use by **ncurses_reset_prog_mode()**. Returns **FALSE** on success, otherwise **TRUE**.

See also: **ncurses_reset_prog_mode()**

ncurses_def_shell_mode

(PHP 4 >= 4.1.0)

ncurses_def_shell_mode -- Saves terminals (shell) mode

Description

bool **ncurses_def_shell_mode** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_def_shell_mode() saves the current terminal modes for shell (not in curses) for use by **ncurses_reset_shell_mode()**. Returns **FALSE** on success, otherwise **TRUE**.

See also: **ncurses_reset_shell_mode()**

ncurses_define_key

(PHP 4 >= 4.2.0)

ncurses_define_key -- Define a keycode

Description

int **ncurses_define_key** (string definition, int keycode)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_delay_output

(PHP 4 >= 4.1.0)

ncurses_delay_output -- Delay output on terminal using padding characters

Description

int **ncurses_delay_output** (int milliseconds)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_delch

(PHP 4 >= 4.1.0)

ncurses_delch -- Delete character at current position, move rest of line left

Description

bool **ncurses_delch** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_delch() deletes the character under the cursor. All characters to the right of the cursor on the same line are moved to the left one position and the last character on the line is filled with a blank. The cursor position does not change. Returns **FALSE** on success, otherwise **TRUE**.

See also: **ncurses_deleteln()**

ncurses_deleteln

(PHP 4 >= 4.1.0)

ncurses_deleteln -- Delete line at current position, move rest of screen up

Description

bool **ncurses_deleteln** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_deleteln() deletes the current line under cursorposition. All lines below the current line are moved up one line. The bottom line of window is cleared. Cursor position does not change. Returns **FALSE** on success, otherwise **TRUE**.

See also: **ncurses_delch()**

ncurses_delwin

(PHP 4 >= 4.1.0)

ncurses_delwin -- Delete a ncurses window

Description

int **ncurses_delwin** (resource window)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_doupdate

(PHP 4 >= 4.1.0)

ncurses_doupdate -- Write all prepared refreshes to terminal

Description

bool **ncurses_doupdate** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_doupdate()() compares the virtual screen to the physical screen and updates the physical screen. This way is more effective than using multiple refresh calls. Returns **FALSE** on success, **TRUE** if any error occured.

ncurses_echo

(PHP 4 >= 4.1.0)

ncurses_echo -- Activate keyboard input echo

Description

bool **ncurses_echo** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_echo() enables echo mode. All characters typed by user are echoed by **ncurses_getch()**. Returns **FALSE** on success, **TRUE** if any error occured.

To disable echo mode use **ncurses_noecho()**.

ncurses_echochar

(PHP 4 >= 4.1.0)

ncurses_echochar -- Single character output including refresh

Description

int **ncurses_echochar** (int character)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_end

(PHP 4 >= 4.1.0)

ncurses_end -- Stop using ncurses, clean up the screen

Description

int **ncurses_end** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_erase

(PHP 4 >= 4.1.0)

ncurses_erase -- Erase terminal screen

Description

bool **ncurses_erase** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_erase() fills the terminal screen with blanks. Created blanks have the current background rendition, set by **ncurses_bkgd()**. Returns **FALSE** on success, **TRUE** if any error occured.

See also: **ncurses_bkgd()**, **ncurses_clear()**

ncurses_erasechar

(PHP 4 >= 4.1.0)

ncurses_erasechar -- Returns current erase character

Description

string **ncurses_erasechar** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_erasechar() returns the current erase char character.

See also: **ncurses_killchar()**

ncurses_filter

(PHP 4 >= 4.1.0)

ncurses_filter --

Description

int **ncurses_filter** (void)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

ncurses_flash

(PHP 4 >= 4.1.0)

ncurses_flash -- Flash terminal screen (visual bell)

Description

bool **ncurses_flash** (void)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_flash() flashes the screen, and if its not possible, sends an audible alert (bell). Returns **FALSE** on success, otherwise **TRUE**.

See also: **ncurses_beep()**

ncurses_flushinp

(PHP 4 >= 4.1.0)

ncurses_flushinp -- Flush keyboard input buffer

Description

bool **ncurses_flushinp** (void)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

The **ncurses_flushinp()** throws away any typeahead that has been typed and has not yet been read by your program. Returns **FALSE** on success, otherwise **TRUE**.

ncurses_getch

(PHP 4 >= 4.1.0)

ncurses_getch -- Read a character from keyboard

Description

int **ncurses_getch** (void)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

ncurses_getmouse

(PHP 4 >= 4.2.0)

ncurses_getmouse -- Reads mouse event

Description

bool **ncurses_getmouse** (array mevent)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_getmouse() reads mouse event out of queue. Function **ncurses_getmouse()** will return ;**FALSE** if a mouse event is actually visible in the given window, otherwise it will return **TRUE**. Event options will be delivered in parameter *mevent*, which has to be an array, passed by reference (see example below). On success an associative array with following keys will be delivered:

- "id" : Id to distinguish multiple devices
- "x" : screen relative x-position in character cells
- "y" : screen relative y-position in character cells
- "z" : currently not supported
- "mmask" : Mouse action

Example 1. ncurses_getmouse() example

```
switch (ncurses_getch){
  case NCURSES_KEY_MOUSE:
    if (!ncurses_getmouse(&$mevent)){
      if ($mevent["mmask"] & NCURSES_MOUSE_BUTTON1_PRESSED){
        $mouse_x = $mevent["x"]; // Save mouse position
        $mouse_y = $mevent["y"];
      }
    }
  break;

  default:
    ....
}
```

See also: **ncurses_ungetmouse()**

ncurses_halfdelay

(PHP 4 >= 4.1.0)

ncurses_halfdelay -- Put terminal into halfdelay mode

Description

int **ncurses_halfdelay** (int tenth)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_has_colors

(PHP 4 >= 4.1.0)

ncurses_has_colors -- Check if terminal has colors

Description

bool **ncurses_has_colors** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_has_colors() returns **TRUE** or **FALSE** depending on whether the terminal has color capacitites.

See also: **ncurses_can_change_color()**

ncurses_has_ic

(PHP 4 >= 4.1.0)

ncurses_has_ic -- Check for insert- and delete-capabilities

Description

bool **ncurses_has_ic** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_has_ic() checks terminals insert- and delete capabilitites. It returns **TRUE** when terminal has insert/delete-capabilities, otherwise **FALSE**.

See also: **ncurses_has_il()**

ncurses_has_il

(PHP 4 >= 4.1.0)

ncurses_has_il -- Check for line insert- and delete-capabilities

Description

bool **ncurses_has_il** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_has_il() checks terminals insert- and delete-line-capabilities. It returns **TRUE** when terminal has insert/delete-line capabilities, otherwise **FALSE**

See also: **ncurses_has_ic()**

ncurses_has_key

(PHP 4 >= 4.1.0)

ncurses_has_key -- Check for presence of a function key on terminal keyboard

Description

int **ncurses_has_key** (int keycode)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_hline

(PHP 4 >= 4.2.0)

ncurses_hline -- Draw a horizontal line at current position using an attributed character and max. n characters long

Description

int **ncurses_hline** (int charattr, int n)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_inch

(PHP 4 >= 4.1.0)

ncurses_inch -- Get character and attribute at current position

Description

string **ncurses_inch** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_inch() returns the character from the current position.

ncurses_init_color

(PHP 4 >= 4.2.0)

ncurses_init_color -- Set new RGB value for color

Description

int **ncurses_init_color** (int color, int r, int g, int b)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

ncurses_init_pair

(PHP 4 >= 4.1.0)

ncurses_init_pair -- Allocate a color pair

Description

int **ncurses_init_pair** (int pair, int fg, int bg)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

ncurses_init

(PHP 4 >= 4.1.0)

ncurses_init -- Initialize ncurses

Description

int **ncurses_init** (void)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

ncurses_insch

(PHP 4 >= 4.1.0)

ncurses_insch -- Insert character moving rest of line including character at current position

Description

int **ncurses_insch** (int character)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

ncurses_insdelln

(PHP 4 >= 4.1.0)

ncurses_insdelln -- Insert lines before current line scrolling down (negative numbers delete and scroll up)

Description

int **ncurses_insdelln** (int count)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_insertln

(PHP 4 >= 4.1.0)

ncurses_insertln -- Insert a line, move rest of screen down

Description

bool **ncurses_insertln** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_insertln() inserts a new line above the current line. The bottom line will be lost.

ncurses_insstr

(PHP 4 >= 4.2.0)

ncurses_insstr -- Insert string at current position, moving rest of line right

Description

int **ncurses_insstr** (string text)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_instr

(PHP 4 >= 4.2.0)

ncurses_instr -- Reads string from terminal screen

Description

int **ncurses_instr** (string buffer)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_instr() returns the number of charaters read from the current character position until end of line. *buffer* contains the characters. Atrributes are stripped from the characters.

ncurses_isendwin

(PHP 4 >= 4.1.0)

ncurses_isendwin -- Ncurses is in endwin mode, normal screen output may be performed

Description

bool **ncurses_isendwin** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_isendwin() returns **TRUE**, if **ncurses_endwin()** has been called without any subsequent calls to **ncurses_wrefresh()**, otherwise **FALSE**.

See also: **ncurses_endwin() ncurses_wrefresh()()**

ncurses_keyok

(PHP 4 >= 4.2.0)

ncurses_keyok -- Enable or disable a keycode

Description

int **ncurses_keyok** (int keycode, bool enable)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_killchar

(PHP 4 >= 4.1.0)

ncurses_killchar -- Returns current line kill character

Description

bool **ncurses_killchar** (void)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_killchar() returns the current line kill character.

See also: **ncurses_erasechar()**

ncurses_longname

(PHP 4 >= 4.2.0)

ncurses_longname -- Returns terminals description

Description

string **ncurses_longname** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_longname() returns a verbose description of the terminal. The descritpion is truncated to 128 characters. On Error **ncurses_longname()** returns NULL.

See also: **ncurses_termname()**

ncurses_mouseinterval

(PHP 4 >= 4.1.0)

ncurses_mouseinterval -- Set timeout for mouse button clicks

Description

int **ncurses_mouseinterval** (int milliseconds)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_mousemask

(PHP 4 >= 4.2.0)

ncurses_mousemask -- Sets mouse options

Description

int **ncurses_mousemask** (int newmask, int oldmask)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Function **ncurses_mousemask()** will set mouse events to be reported. By default no mouse events will be reported. The function **ncurses_mousemask()** will return a mask to indicated which of the in parameter *newmask* specified mouse events can be

reported. On complete failure, it returns o. In parameter *oldmask*, which is passed by reference **ncurses_mousemask()** returns the previous value of mouse event mask. Mouse events are represented bei NCURSES_KEY_MOUSE in the **ncurses_wgetch()** input stream. To read the event data and pop the event of of queue, call **ncurses_getmouse()**.

As a side effect, setting a zero mousemask in *newmask* turns off the mouse pointer. Setting a non zero value turns mouse pointer on.

mouse mask options can be set with the following predefined constants:

- NCURSES_BUTTON1_PRESSED
- NCURSES_BUTTON1_RELEASED
- NCURSES_BUTTON1_CLICKED
- NCURSES_BUTTON1_DOUBLE_CLICKED
- NCURSES_BUTTON1_TRIPLE_CLICKED
- NCURSES_BUTTON2_PRESSED
- NCURSES_BUTTON2_RELEASED
- NCURSES_BUTTON2_CLICKED
- NCURSES_BUTTON2_DOUBLE_CLICKED
- NCURSES_BUTTON2_TRIPLE_CLICKED
- NCURSES_BUTTON3_PRESSED
- NCURSES_BUTTON3_RELEASED
- NCURSES_BUTTON3_CLICKED
- NCURSES_BUTTON3_DOUBLE_CLICKED
- NCURSES_BUTTON3_TRIPLE_CLICKED
- NCURSES_BUTTON4_PRESSED
- NCURSES_BUTTON4_RELEASED
- NCURSES_BUTTON4_CLICKED
- NCURSES_BUTTON4_DOUBLE_CLICKED
- NCURSES_BUTTON4_TRIPLE_CLICKED
- NCURSES_BUTTON_SHIFT>
- NCURSES_BUTTON_CTRL
- NCURSES_BUTTON_ALT
- NCURSES_ALL_MOUSE_EVENTS
- NCURSES_REPORT_MOUSE_POSITION

See also: **ncurses_getmouse()**, **ncurses_ungetmouse()** **ncurese_getch()**

Example 1. ncurses_mousemask() example

```
$newmask = NCURSES_BUTTON1_CLICKED + NCURSES_BUTTON1_RELEASED;
$mask = ncurses_mousemask($newmask, &$oldmask);
if ($mask & $newmask){
  printf ("All specified mouse options will be supported\n");
}
```

ncurses_move

(PHP 4 >= 4.1.0)

ncurses_move -- Move output position

Description

int **ncurses_move** (int y, int x)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_mvaddch

(PHP 4 >= 4.2.0)

ncurses_mvaddch -- Move current position and add character

Description

int **ncurses_mvaddch** (int y, int x, int c)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_mvaddchnstr

(PHP 4 >= 4.2.0)

ncurses_mvaddchnstr -- Move position and add attrributed string with specified length

Description

int **ncurses_mvaddchnstr** (int y, int x, string s, int n)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_mvaddchstr

(PHP 4 >= 4.2.0)

ncurses_mvaddchstr -- Move position and add attributed string

Description

int **ncurses_mvaddchstr** (int y, int x, string s)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

ncurses_mvaddnstr

(PHP 4 >= 4.2.0)

ncurses_mvaddnstr -- Move position and add string with specified length

Description

int **ncurses_mvaddnstr** (int y, int x, string s, int n)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

ncurses_mvaddstr

(PHP 4 >= 4.2.0)

ncurses_mvaddstr -- Move position and add string

Description

int **ncurses_mvaddstr** (int y, int x, string s)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

ncurses_mvcur

(PHP 4 >= 4.2.0)

ncurses_mvcur -- Move cursor immediately

Description

int **ncurses_mvcur** (int old_y, int old_x, int new_y, int new_x)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

ncurses_mvdelch

(PHP 4 >= 4.2.0)

ncurses_mvdelch -- Move position and delete character, shift rest of line left

Description

int **ncurses_mvdelch** (int y, int x)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_mvgetch

(PHP 4 >= 4.2.0)

ncurses_mvgetch -- Move position and get character at new position

Description

int **ncurses_mvgetch** (int y, int x)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_mvhline

(PHP 4 >= 4.2.0)

ncurses_mvhline -- Set new position and draw a horizontal line using an attributed character and max. n characters long

Description

int **ncurses_mvhline** (int y, int x, int attrchar, int n)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_mvinch

(PHP 4 >= 4.2.0)

ncurses_mvinch -- Move position and get attributed character at new position

Description

int **ncurses_mvinch** (int y, int x)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_mvvline

(no version information, might be only in CVS)

ncurses_mvvline -- Set new position and draw a vertical line using an attributed character and max. n characters long

Description

int **ncurses_mvvline** (int y, int x, int attrchar, int n)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_mvwaddstr

(PHP 4 >= 4.2.0)

ncurses_mvwaddstr -- Add string at new position in window

Description

int **ncurses_mvwaddstr** (resource window, int y, int x, string text)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_napms

(PHP 4 >= 4.1.0)

ncurses_napms -- Sleep

Description

int **ncurses_napms** (int milliseconds)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

ncurses_newwin

(PHP 4 >= 4.1.0)

ncurses_newwin -- Create a new window

Description

int **ncurses_newwin** (int rows, int cols, int y, int x)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

ncurses_nl

(PHP 4 >= 4.1.0)

ncurses_nl -- Translate newline and carriage return / line feed

Description

bool **ncurses_nl** (void)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

ncurses_nocbreak

(PHP 4 >= 4.1.0)

ncurses_nocbreak -- Switch terminal to cooked mode

Description

bool **ncurses_nocbreak** (void)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_nocbreak() routine returns terminal to normal (cooked) mode. Initially the terminal may or may not in cbreak mode as the mode is inherited. Therefore a program should call **ncurses_cbreak()** and **ncurses_nocbreak()** explicitly. Returns **TRUE** if any error occured, otherwise **FALSE**.

See also: **ncurses_cbreak()**

ncurses_noecho

(PHP 4 >= 4.1.0)

ncurses_noecho -- Switch off keyboard input echo

Description

bool **ncurses_noecho** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_noecho() prevents echoing of user typed characters. Returns **TRUE** if any error occured, otherwise **FALSE**.

See also: **ncurses_echo()**, **ncurses_getch()**

ncurses_nonl

(PHP 4 >= 4.1.0)

ncurses_nonl -- Do not translate newline and carriage return / line feed

Description

bool **ncurses_nonl** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_noqiflush

(PHP 4 >= 4.1.0)

ncurses_noqiflush -- Do not flush on signal characters

Description

int **ncurses_noqiflush** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_noraw

(PHP 4 >= 4.1.0)

ncurses_noraw -- Switch terminal out of raw mode

Description

bool **ncurses_noraw** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_noraw() switches the terminal out of raw mode. Raw mode is similar to cbreak mode, in that characters typed are immediately passed through to the user program. The differences that are that in raw mode, the interrupt, quit, suspend and flow control characters are all passed through uninterpreted, instead of generating a signal. Returns **TRUE** if any error occured, otherwise **FALSE**.

See also: **ncurses_raw()**, **ncurses_cbreak()**, **ncurses_nocbreak()**

ncurses_putp

(PHP 4 >= 4.2.0)

ncurses_putp --

Description

int **ncurses_putp** (string text)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_qiflush

(PHP 4 >= 4.1.0)

ncurses_qiflush -- Flush on signal characters

Description

int **ncurses_qiflush** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_raw

(PHP 4 >= 4.1.0)

ncurses_raw -- Switch terminal into raw mode

Description

bool **ncurses_raw** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_raw() places the terminal in raw mode. Raw mode is similar to cbreak mode, in that characters typed are immediately passed through to the user program. The differences that are that in raw mode, the interrupt, quit, suspend and flow control characters are all passed through uninterpreted, instead of generating a signal. Returns **TRUE** if any error occured, otherwise **FALSE**.

See also: **ncurses_noraw()**, **ncurses_cbreak()**, **ncurses_nocbreak()**

ncurses_refresh

(PHP 4 >= 4.1.0)

ncurses_refresh -- Refresh screen

Description

int **ncurses_refresh** (int ch)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_resetty

(PHP 4 >= 4.1.0)

ncurses_resetty -- Restores saved terminal state

Description

bool **ncurses_resetty** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Function **ncurses_resetty()** restores the terminal state, which was previously saved by calling **ncurses_savetty()**. This function always returns **FALSE**.

See also: **ncurses_savetty()**

ncurses_savetty

(PHP 4 >= 4.1.0)

ncurses_savetty -- Saves terminal state

Description

bool **ncurses_savetty** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Function **ncurses_savetty()** saves the current terminal state. The saved terminal state can be restored with function **ncurses_resetty()**. **ncurses_savetty()** always returns FALSE.

See also: **ncurses_resetty()**

ncurses_scr_dump

(PHP 4 >= 4.2.0)

ncurses_scr_dump -- Dump screen content to file

Description

int **ncurses_scr_dump** (string filename)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_scr_init

(PHP 4 >= 4.2.0)

ncurses_scr_init -- Initialize screen from file dump

Description

int **ncurses_scr_init** (string filename)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_scr_restore

(PHP 4 >= 4.2.0)

ncurses_scr_restore -- Restore screen from file dump

Description

int **ncurses_scr_restore** (string filename)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

ncurses_scr_set

(PHP 4 >= 4.2.0)

ncurses_scr_set -- Inherit screen from file dump

Description

int **ncurses_scr_set** (string filename)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

ncurses_scrl

(PHP 4 >= 4.1.0)

ncurses_scrl -- Scroll window content up or down without changing current position

Description

int **ncurses_scrl** (int count)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

ncurses_slk_attr

(PHP 4 >= 4.1.0)

ncurses_slk_attr -- Returns current soft label key attribute

Description

bool **ncurses_slk_attr** (void)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_slk_attr() returns the current soft label key attribute. On error returns **TRUE**, otherwise **FALSE**.

ncurses_slk_attroff

(PHP 4 >= 4.1.0)

ncurses_slk_attroff --

Description

int **ncurses_slk_attroff** (int intarg)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_slk_attron

(PHP 4 >= 4.1.0)

ncurses_slk_attron --

Description

int **ncurses_slk_attron** (int intarg)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_slk_attrset

(PHP 4 >= 4.1.0)

ncurses_slk_attrset --

Description

int **ncurses_slk_attrset** (int intarg)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_slk_clear

(PHP 4 >= 4.1.0)

ncurses_slk_clear -- Clears soft labels from screen

Description

bool **ncurses_slk_clear** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

The function **ncurses_slk_clear()** clears soft label keys from screen. Returns **TRUE** on error, otherwise **FALSE**.

ncurses_slk_color

(PHP 4 >= 4.1.0)

ncurses_slk_color -- Sets color for soft label keys

Description

int **ncurses_slk_color** (int intarg)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_slk_init

(PHP 4 >= 4.1.0)

ncurses_slk_init -- Initializes soft label key functions

Description

bool **ncurses_slk_init** (int format)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Funtion **ncurses_slk_init()** must be called before **ncurses_initscr()** or **ncurses_newterm()** is called. If **ncurses_initscr()** eventually uses a line from stdscr to emulate the soft labels, then *format* determines how the labels are arranged of the screen. Setting *format* to 0 indicates a 3-2-3 arrangement of the labels, 1 indicates a 4-4 arrangement and 2 indicates the PC like 4-4-4 mode, but in addition an index line will be created.

ncurses_slk_noutrefresh

(PHP 4 >= 4.1.0)

ncurses_slk_noutrefresh -- Copies soft label keys to virtual screen

Description

bool **ncurses_slk_noutrefresh** (void)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

ncurses_slk_refresh

(PHP 4 >= 4.1.0)

ncurses_slk_refresh -- Copies soft label keys to screen

Description

bool **ncurses_slk_refresh** (void)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_slk_refresh() copies soft label keys from virtual screen to physical screen. Returns **TRUE** on error, otherwise **FALSE**.

ncurses_slk_restore

(PHP 4 >= 4.1.0)

ncurses_slk_restore -- Restores soft label keys

Description

bool **ncurses_slk_restore** (void)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

The function **ncurses_slk_restore()** restores the soft label keys after **ncurses_slk_clear()** has been performed.

ncurses_slk_touch

(PHP 4 >= 4.1.0)

ncurses_slk_touch -- Fources output when ncurses_slk_noutrefresh is performed

Description

bool **ncurses_slk_touch** (void)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

The **ncurses_slk_touch()** function forces all the soft labels to be output the next time a **ncurses_slk_noutrefresh()** is performed.

ncurses_standend

(PHP 4 >= 4.1.0)

ncurses_standend -- Stop using 'standout' attribute

Description

int **ncurses_standend** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_standout

(PHP 4 >= 4.1.0)

ncurses_standout -- Start using 'standout' attribute

Description

int **ncurses_standout** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_start_color

(PHP 4 >= 4.1.0)

ncurses_start_color -- Start using colors

Description

int **ncurses_start_color** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_termattrs

(PHP 4 >= 4.1.0)

ncurses_termattrs -- Returns a logical OR of all attribute flags supported by terminal

Description

bool **ncurses_termattrs** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_termname

(PHP 4 >= 4.2.0)

ncurses_termname -- Returns terminals (short)-name

Description

string **ncurses_termname** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_termname() returns terminals shortname. The shortname is truncated to 14 characters. On error **ncurses_termname()** returns NULL.

See also: **ncurses_longname()**

ncurses_timeout

(PHP 4 >= 4.1.0)

ncurses_timeout -- Set timeout for special key sequences

Description

void **ncurses_timeout** (int millisec)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_typeahead

(PHP 4 >= 4.1.0)

ncurses_typeahead -- Specify different filedescriptor for typeahead checking

Description

int **ncurses_typeahead** (int fd)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_ungetch

(PHP 4 >= 4.1.0)

ncurses_ungetch -- Put a character back into the input stream

Description

int **ncurses_ungetch** (int keycode)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_ungetmouse

(PHP 4 >= 4.2.0)

ncurses_ungetmouse -- Pushes mouse event to queue

Description

bool **ncurses_ungetmouse** (array mevent)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

ncurses_getmouse() pushes a KEY_MOUSE event onto the unput queue and associates with this event the given state sata and screen-relative character cell coordinates, specified in *mevent*. Event options will be specified in associative array *mevent*:

- "id" : Id to distinguish multiple devices
- "x" : screen relative x-position in character cells
- "y" : screen relative y-position in character cells
- "z" : currently not supported
- "mmask" : Mouse action

ncurses_ungetmouse() returns **FALSE** on success, otherwise **TRUE**.

See also: **ncurses_getmouse()**

ncurses_use_default_colors

(PHP 4 >= 4.1.0)

ncurses_use_default_colors -- Assign terminal default colors to color id -1

Description

bool **ncurses_use_default_colors** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_use_env

(PHP 4 >= 4.1.0)

ncurses_use_env -- Control use of environment information about terminal size

Description

void **ncurses_use_env** (bool flag)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_use_extended_names

(PHP 4 >= 4.1.0)

ncurses_use_extended_names -- Control use of extended names in terminfo descriptions

Description

int **ncurses_use_extended_names** (bool flag)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_vidattr

(PHP 4 >= 4.1.0)

ncurses_vidattr --

Description

int **ncurses_vidattr** (int intarg)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

ncurses_vline

(PHP 4 >= 4.2.0)

ncurses_vline -- Draw a vertical line at current position using an attributed character and max. n characters long

Description

int **ncurses_vline** (int charattr, int n)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

ncurses_wrefresh

(PHP 4 >= 4.2.0)

ncurses_wrefresh -- Refresh window on terminal screen

Description

int **ncurses_wrefresh** (resource window)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

LXVIII. Lotus Notes functions

Introduction

Warning
This extension is *EXPERIMENTAL*. The behaviour of this extension -- including the names of its functions and anything else documented about this extension -- may change without notice in a future release of PHP. Use this extension at your own risk.

Table of Contents

notes_version -- Get the version Lotus Notes

notes_body

(PHP 4 >= 4.0.5)

notes_body -- Open the message msg_number in the specified mailbox on the specified server (leave serv

Description

array **notes_body** (string server, string mailbox, int msg_number)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

notes_copy_db

(PHP 4 >= 4.0.5)

notes_copy_db -- Create a note using form form_name

Description

string **notes_copy_db** (string from_database_name, string to_database_name)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

notes_create_db

(PHP 4 >= 4.0.5)

notes_create_db -- Create a Lotus Notes database

Description

bool **notes_create_db** (string database_name)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

notes_create_note

(PHP 4 >= 4.0.5)

notes_create_note -- Create a note using form form_name

Description

string **notes_create_note** (string database_name, string form_name)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

notes_drop_db

(PHP 4 >= 4.0.5)

notes_drop_db -- Drop a Lotus Notes database

Description

bool **notes_drop_db** (string database_name)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

notes_find_note

(PHP 4 >= 4.0.5)

notes_find_note -- Returns a note id found in database_name. Specify the name of the note. Leaving type bla

Description

bool **notes_find_note** (string database_name, string name [, string type])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

notes_header_info

(PHP 4 >= 4.0.5)

notes_header_info -- Open the message msg_number in the specified mailbox on the specified server (leave serv

Description

object **notes_header_info** (string server, string mailbox, int msg_number)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

notes_list_msgs

(PHP 4 >= 4.0.5)

notes_list_msgs -- Returns the notes from a selected database_name

Description

bool **notes_list_msgs** (string db)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

notes_mark_read

(PHP 4 >= 4.0.5)

notes_mark_read -- Mark a note_id as read for the User user_name

Description

string **notes_mark_read** (string database_name, string user_name, string note_id)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

notes_mark_unread

(PHP 4 >= 4.0.5)

notes_mark_unread -- Mark a note_id as unread for the User user_name

Description

string **notes_mark_unread** (string database_name, string user_name, string note_id)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

notes_nav_create

(PHP 4 >= 4.0.5)

notes_nav_create -- Create a navigator name, in database_name

Description

bool **notes_nav_create** (string database_name, string name)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

notes_search

(PHP 4 >= 4.0.5)

notes_search -- Find notes that match keywords in database_name

Description

string **notes_search** (string database_name, string keywords)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

notes_unread

(PHP 4 >= 4.0.5)

notes_unread -- Returns the unread note id's for the current User user_name

Description

string **notes_unread** (string database_name, string user_name)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

notes_version

(PHP 4 >= 4.0.5)

notes_version -- Get the version Lotus Notes

Description

string **notes_version** (string database_name)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

LXIX. Unified ODBC functions

Introduction

In addition to normal ODBC support, the Unified ODBC functions in PHP allow you to access several databases that have borrowed the semantics of the ODBC API to implement their own API. Instead of maintaining multiple database drivers that were all nearly identical, these drivers have been unified into a single set of ODBC functions.

The following databases are supported by the Unified ODBC functions: Adabas D, IBM DB2, iODBC, Solid, and Sybase SQL Anywhere.

> **Note:** There is no ODBC involved when connecting to the above databases. The functions that you use to speak natively to them just happen to share the same names and syntax as the ODBC functions. The exception to this is iODBC. Building PHP with iODBC support enables you to use any ODBC-compliant drivers with your PHP applications. iODBC is maintained by OpenLink Software. More information on iODBC, as well as a HOWTO, is available at www.iodbc.org.

Requirements

To access any of the supported databases you need to have the required libraries installed.

Installation

Please see the Database installation options chapter for more information about configuring PHP with these databases.

The windows version of PHP has built in support for this extension. You do not need to load any additional extension in order to use these functions.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. Unified ODBC Configuration Options

Name	Default	Changeable
odbc.default_db *	NULL	PHP_INI_ALL
odbc.default_user *	NULL	PHP_INI_ALL
odbc.default_pw *	NULL	PHP_INI_ALL
odbc.allow_persistent	"1"	PHP_INI_SYSTEM

Name	Default	Changeable
odbc.check_persistent	"1"	PHP_INI_SYSTEM
odbc.max_persistent	"-1"	PHP_INI_SYSTEM
odbc.max_links	"-1"	PHP_INI_SYSTEM
odbc.defaultlrl	"4096"	PHP_INI_ALL
odbc.defaultbinmode	"1"	PHP_INI_ALL

Note: Entries marked with * are not implemented yet.

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Here is a short explanation of the configuration directives.

odbc.default_db **string**

ODBC data source to use if none is specified in **odbc_connect()** or **odbc_pconnect()**.

odbc.default_user **string**

User name to use if none is specified in **odbc_connect()** or **odbc_pconnect()**.

odbc.default_pw **string**

Password to use if none is specified in **odbc_connect()** or **odbc_pconnect()**.

odbc.allow_persistent **boolean**

Whether to allow persistent ODBC connections.

odbc.check_persistent **boolean**

Check that a connection is still valid before reuse.

odbc.max_persistent **integer**

The maximum number of persistent ODBC connections per process.

odbc.max_links **integer**

The maximum number of ODBC connections per process, including persistent connections.

odbc.defaultlrl **integer**

Handling of LONG fields. Specifies the number of bytes returned to variables.

odbc.defaultbinmode **integer**

Handling of binary data.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

ODBC_TYPE (integer)

ODBC_BINMODE_PASSTHRU (integer)

ODBC_BINMODE_RETURN (integer)

ODBC_BINMODE_CONVERT (integer)

SQL_ODBC_CURSORS (integer)

SQL_CUR_USE_DRIVER (integer)

SQL_CUR_USE_IF_NEEDED (integer)

SQL_CUR_USE_ODBC (integer)

SQL_CONCURRENCY (integer)

SQL_CONCUR_READ_ONLY (integer)

SQL_CONCUR_LOCK (integer)

SQL_CONCUR_ROWVER (integer)

SQL_CONCUR_VALUES (integer)

SQL_CURSOR_TYPE (integer)

SQL_CURSOR_FORWARD_ONLY (integer)

SQL_CURSOR_KEYSET_DRIVEN (integer)

SQL_CURSOR_DYNAMIC (integer)

SQL_CURSOR_STATIC (integer)

SQL_KEYSET_SIZE (integer)

SQL_CHAR (integer)

SQL_VARCHAR (integer)

SQL_LONGVARCHAR (integer)

SQL_DECIMAL (integer)

SQL_NUMERIC (integer)

SQL_BIT (integer)

SQL_TINYINT (integer)

SQL_SMALLINT (integer)

SQL_INTEGER (integer)

SQL_BIGINT (integer)

SQL_REAL (integer)

SQL_FLOAT (integer)

SQL_DOUBLE (integer)

SQL_BINARY (integer)

SQL_VARBINARY (integer)

SQL_LONGVARBINARY (integer)

SQL_DATE (integer)

SQL_TIME (integer)

SQL_TIMESTAMP (integer)

SQL_TYPE_DATE (integer)

SQL_TYPE_TIME (integer)

SQL_TYPE_TIMESTAMP (integer)

SQL_BEST_ROWID (integer)

SQL_ROWVER (integer)

SQL_SCOPE_CURROW (integer)

SQL_SCOPE_TRANSACTION (integer)

SQL_SCOPE_SESSION (integer)

SQL_NO_NULLS (integer)

SQL_NULLABLE (integer)

SQL_INDEX_UNIQUE (integer)

SQL_INDEX_ALL (integer)

SQL_ENSURE (integer)

SQL_QUICK (integer)

Table of Contents

odbc_autocommit

(PHP 3>= 3.0.6, PHP 4)

odbc_autocommit -- Toggle autocommit behaviour

Description

bool **odbc_autocommit** (resource connection_id [, bool OnOff])

Without the *OnOff* parameter, this function returns auto-commit status for *connection_id*. **TRUE** is returned if auto-commit is on, **FALSE** if it is off or an error occurs.

If *OnOff* is **TRUE**, auto-commit is enabled, if it is **FALSE** auto-commit is disabled. Returns **TRUE** on success, **FALSE** on failure.

By default, auto-commit is on for a connection. Disabling auto-commit is equivalent with starting a transaction.

See also odbc_commit() and odbc_rollback().

odbc_binmode

(PHP 3>= 3.0.6, PHP 4)

odbc_binmode -- Handling of binary column data

Description

int **odbc_binmode** (resource result_id, int mode)

(ODBC SQL types affected: BINARY, VARBINARY, LONGVARBINARY)

- ODBC_BINMODE_PASSTHRU: Passthru BINARY data
- ODBC_BINMODE_RETURN: Return as is
- ODBC_BINMODE_CONVERT: Convert to char and return

When binary SQL data is converted to character C data, each byte (8 bits) of source data is represented as two ASCII characters. These characters are the ASCII character representation of the number in its hexadecimal form. For example, a binary 00000001 is converted to "01" and a binary 11111111 is converted to "FF".

Table 1. LONGVARBINARY handling

binmode	longreadlen	result
ODBC_BINMODE_PASSTHRU	0	passthru
ODBC_BINMODE_RETURN	0	passthru
ODBC_BINMODE_CONVERT	0	passthru
ODBC_BINMODE_PASSTHRU	0	passthru
ODBC_BINMODE_PASSTHRU	>0	passthru
ODBC_BINMODE_RETURN	>0	return as is
ODBC_BINMODE_CONVERT	>0	return as char

If odbc_fetch_into() is used, passthru means that an empty string is returned for these columns.

If *result_id* is 0, the settings apply as default for new results.

> **Note:** Default for longreadlen is 4096 and binmode defaults to ODBC_BINMODE_RETURN. Handling of binary long columns is also affected by odbc_longreadlen()

odbc_close_all

(PHP 3>= 3.0.6, PHP 4)

odbc_close_all -- Close all ODBC connections

Description

void **odbc_close_all** (void)

odbc_close_all() will close down all connections to database server(s).

> **Note:** This function will fail if there are open transactions on a connection. This connection will remain open in this case.

odbc_close

(PHP 3>= 3.0.6, PHP 4)

odbc_close -- Close an ODBC connection

Description

void **odbc_close** (resource connection_id)

odbc_close() will close down the connection to the database server associated with the given connection identifier.

> **Note:** This function will fail if there are open transactions on this connection. The connection will remain open in this case.

odbc_columnprivileges

(PHP 4)

odbc_columnprivileges -- Returns a result identifier that can be used to fetch a list of columns and associated privileges

Description

int **odbc_columnprivileges** (resource connection_id [, string qualifier [, string owner [, string table_name [, string column_name]]]])

Lists columns and associated privileges for the given table. Returns an ODBC result identifier or **FALSE** on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- GRANTOR
- GRANTEE
- PRIVILEGE
- IS_GRANTABLE

The result set is ordered by TABLE_QUALIFIER, TABLE_OWNER and TABLE_NAME.

The *column_name* argument accepts search patterns ('%' to match zero or more characters and '_' to match a single character).

odbc_columns

(PHP 4)

odbc_columns -- Lists the column names in specified tables. Returns a result identifier containing the information.

Description

int **odbc_columns** (resource connection_id [, string qualifier [, string schema [, string table_name [, string column_name]]]])

Lists all columns in the requested range. Returns an ODBC result identifier or **FALSE** on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_SCHEM
- TABLE_NAME
- COLUMN_NAME
- DATA_TYPE
- TYPE_NAME
- PRECISION
- LENGTH
- SCALE
- RADIX
- NULLABLE
- REMARKS

The result set is ordered by TABLE_QUALIFIER, TABLE_SCHEM and TABLE_NAME.

The *schema*, *table_name* and *column_name* arguments accept search patterns ('%' to match zero or more characters and '_' to match a single character).

See also odbc_columnprivileges() to retrieve associated privileges.

odbc_commit

(PHP 3>= 3.0.6, PHP 4)

odbc_commit -- Commit an ODBC transaction

Description

bool **odbc_commit** (resource connection_id)

Returns: **TRUE** on success, **FALSE** on failure. All pending transactions on *connection_id* are committed.

odbc_connect

(PHP 3>= 3.0.6, PHP 4)

odbc_connect -- Connect to a datasource

Description

resource **odbc_connect** (string dsn, string user, string password [, int cursor_type])

Returns an ODBC connection id or 0 (**FALSE**) on error.

The connection id returned by this functions is needed by other ODBC functions. You can have multiple connections open at once. The optional fourth parameter sets the type of cursor to be used for this connection. This parameter is not normally needed, but can be useful for working around problems with some ODBC drivers.

With some ODBC drivers, executing a complex stored procedure may fail with an error similar to: "Cannot open a cursor on a stored procedure that has anything other than a single select statement in it". Using SQL_CUR_USE_ODBC may avoid that error. Also, some drivers don't support the optional row_number parameter in **odbc_fetch_row()**. SQL_CUR_USE_ODBC might help in that case, too.

The following constants are defined for cursortype:

- SQL_CUR_USE_IF_NEEDED
- SQL_CUR_USE_ODBC
- SQL_CUR_USE_DRIVER
- SQL_CUR_DEFAULT

For persistent connections see **odbc_pconnect()**.

odbc_cursor

(PHP 3>= 3.0.6, PHP 4)

odbc_cursor -- Get cursorname

Description

string **odbc_cursor** (resource result_id)

odbc_cursor will return a cursorname for the given result_id.

odbc_data_source

(PHP 4 >= 4.3.0)

odbc_data_source -- Returns information about a current connection

Description

resource **odbc_data_source** (resource connection_id, constant fetch_type)

Returns **FALSE** on error, and an array upon success.

This function will return information about the active connection following the information from within the DSN. The *connection_id* is required to be a valid ODBC connection. The *fetch_type* can be one of two constant types: SQL_FETCH_FIRST, SQL_FETCH_NEXT. Use SQL_FETCH_FIRST the first time this function is called, thereafter use the SQL_FETCH_NEXT.

odbc_do

(PHP 3>= 3.0.6, PHP 4)

odbc_do -- Synonym for **odbc_exec()**

Description

resource **odbc_do** (resource conn_id, string query)

odbc_do() will execute a query on the given connection.

odbc_error

(PHP 4 >= 4.0.5)

odbc_error -- Get the last error code

Description

string **odbc_error** ([resource connection_id])

Returns a six-digit ODBC state, or an empty string if there has been no errors. If *connection_id* is specified, the last state of that connection is returned, else the last state of any connection is returned.

See also: **odbc_errormsg()** and **odbc_exec()**.

odbc_errormsg

(PHP 4 >= 4.0.5)

odbc_errormsg -- Get the last error message

Description

string **odbc_errormsg** ([resource connection_id])

Returns a string containing the last ODBC error message, or an empty string if there has been no errors. If *connection_id* is specified, the last state of that connection is returned, else the last state of any connection is returned.

See also: **odbc_error()** and **odbc_exec()**.

odbc_exec

(PHP 3>= 3.0.6, PHP 4)

odbc_exec -- Prepare and execute a SQL statement

Description

resource **odbc_exec** (resource connection_id, string query_string)

Returns **FALSE** on error. Returns an ODBC result identifier if the SQL command was executed successfully.

odbc_exec() will send an SQL statement to the database server specified by *connection_id*. This parameter must be a valid identifier returned by **odbc_connect()** or **odbc_pconnect()**.

See also: **odbc_prepare()** and **odbc_execute()** for multiple execution of SQL statements.

odbc_execute

(PHP 3>= 3.0.6, PHP 4)

odbc_execute -- Execute a prepared statement

Description

bool **odbc_execute** (resource result_id [, array parameters_array])

Executes a statement prepared with **odbc_prepare()**.Returns **TRUE** on success or **FALSE** on failure. The array *parameters_array* only needs to be given if you really have parameters in your statement.

Parameters in *parameter_array* will be substituted for placeholders in the prepared statement in order.

Any parameters in *parameter_array* which start and end with single quotes will be taken as the name of a file to read and send to the database server as the data for the appropriate placeholder.

Note: As of PHP 4.1.1, this file reading functionality has the following restrictions:

- File reading is *not* subject to any safe mode or open-basedir restrictions. This is fixed in PHP 4.2.0.

- Remote files are not supported.
- If you wish to store a string which actually begins and ends with single quotes, you must escape them or add a space or other non-single-quote character to the beginning or end of the parameter, which will prevent the parameter's being taken as a file name. If this is not an option, then you must use another mechanism to store the string, such as executing the query directly with **odbc_exec()**).

odbc_fetch_array

(PHP 4 >= 4.0.2)

odbc_fetch_array -- Fetch a result row as an associative array

Description

array **odbc_fetch_array** (resource result [, int rownumber])

Warning
This function is currently not documented; only the argument list is available.

odbc_fetch_into

(PHP 3>= 3.0.6, PHP 4)

odbc_fetch_into -- Fetch one result row into array

Description

bool **odbc_fetch_into** (resource result_id [, int rownumber, array result_array])

resource **odbc_fetch_into** (resource result_id, array result_array [, int rownumber])

Returns the number of columns in the result; **FALSE** on error. *result_array* must be passed by reference, but it can be of any type since it will be converted to type array. The array will contain the column values starting at array index 0.

As of PHP 4.0.5 the *result_array* does not need to be passed by reference any longer.

As of PHP 4.0.6 the *rownumber* cannot be passed as a constant, but rather as a variable.

As of PHP 4.2.0 the *result_array* and *rownumber* have been swapped. This allows the rownumber to be a constant again. This change will also be the last one to this function.

Example 1. odbc_fetch_into() pre 4.0.6 example

```
$rc = odbc_fetch_into($res_id, $my_array);
```

or

```
$rc = odbc_fetch_into($res_id, $row, $my_array);
$rc = odbc_fetch_into($res_id, 1, $my_array);
```

Example 2. odbc_fetch_into() 4.0.6 example

```
$rc = odbc_fetch_into($res_id, $my_array);
```

or

```
$row = 1;
$rc = odbc_fetch_into($res_id, $row, $my_array);
```

Example 3. odbc_fetch_into() 4.2.0 example

```
$rc = odbc_fetch_into($res_id, $my_array);
```

or

```
$rc = odbc_fetch_into($res_id,$my_array, 2);
```

odbc_fetch_object

(PHP 4 >= 4.0.2)

odbc_fetch_object -- Fetch a result row as an object

Description

object **odbc_fetch_object** (resource result [, int rownumber])

Warning
This function is currently not documented; only the argument list is available.

odbc_fetch_row

(PHP 3>= 3.0.6, PHP 4)

odbc_fetch_row -- Fetch a row

Description

bool **odbc_fetch_row** (resource result_id [, int row_number])

If **odbc_fetch_row()** was succesful (there was a row), **TRUE** is returned. If there are no more rows, **FALSE** is returned.

odbc_fetch_row() fetches a row of the data that was returned by odbc_do() / odbc_exec(). After **odbc_fetch_row()** is called, the fields of that row can be accessed with odbc_result().

If *row_number* is not specified, **odbc_fetch_row()** will try to fetch the next row in the result set. Calls to **odbc_fetch_row()** with and without *row_number* can be mixed.

To step through the result more than once, you can call **odbc_fetch_row()** with *row_number* 1, and then continue doing **odbc_fetch_row()** without *row_number* to review the result. If a driver doesn't support fetching rows by number, the *row_number* parameter is ignored.

odbc_field_len

(PHP 3>= 3.0.6, PHP 4)

odbc_field_len -- Get the length (precision) of a field

Description

int **odbc_field_len** (resource result_id, int field_number)

odbc_field_len() will return the length of the field referecend by number in the given ODBC result identifier. Field numbering starts at 1.

See also: odbc_field_scale() to get the scale of a floating point number.

odbc_field_name

(PHP 3>= 3.0.6, PHP 4)

odbc_field_name -- Get the columnname

Description

string **odbc_field_name** (resource result_id, int field_number)

odbc_field_name() will return the name of the field occupying the given column number in the given ODBC result identifier. Field numbering starts at 1. **FALSE** is returned on error.

odbc_field_num

(PHP 3>= 3.0.6, PHP 4)

odbc_field_num -- Return column number

Description

int **odbc_field_num** (resource result_id, string field_name)

odbc_field_num() will return the number of the column slot that corresponds to the named field in the given ODBC result identifier. Field numbering starts at 1. **FALSE** is returned on error.

odbc_field_precision

(PHP 4)

odbc_field_precision -- Synonym for **odbc_field_len()**

Description

string **odbc_field_precision** (resource result_id, int field_number)

odbc_field_precision() will return the precision of the field referecend by number in the given ODBC result identifier.

See also: **odbc_field_scale()** to get the scale of a floating point number.

odbc_field_scale

(PHP 4)

odbc_field_scale -- Get the scale of a field

Description

string **odbc_field_scale** (resource result_id, int field_number)

odbc_field_precision() will return the scale of the field referecend by number in the given ODBC result identifier.

odbc_field_type

(PHP 3>= 3.0.6, PHP 4)

odbc_field_type -- Datatype of a field

Description

string **odbc_field_type** (resource result_id, int field_number)

odbc_field_type() will return the SQL type of the field referecend by number in the given ODBC result identifier. Field

numbering starts at 1.

odbc_foreignkeys

(PHP 4)

odbc_foreignkeys -- Returns a list of foreign keys in the specified table or a list of foreign keys in other tables that refer to the primary key in the specified table

Description

resource **odbc_foreignkeys** (resource connection_id, string pk_qualifier, string pk_owner, string pk_table, string fk_qualifier, string fk_owner, string fk_table)

odbc_foreignkeys() retrieves information about foreign keys. Returns an ODBC result identifier or **FALSE** on failure.

The result set has the following columns:

- PKTABLE_QUALIFIER
- PKTABLE_OWNER
- PKTABLE_NAME
- PKCOLUMN_NAME
- FKTABLE_QUALIFIER
- FKTABLE_OWNER
- FKTABLE_NAME
- FKCOLUMN_NAME
- KEY_SEQ
- UPDATE_RULE
- DELETE_RULE
- FK_NAME
- PK_NAME

If *pk_table* contains a table name, **odbc_foreignkeys()** returns a result set containing the primary key of the specified table and all of the foreign keys that refer to it.

If *fk_table* contains a table name, **odbc_foreignkeys()** returns a result set containing all of the foreign keys in the specified table and the primary keys (in other tables) to which they refer.

If both *pk_table* and *fk_table* contain table names, **odbc_foreignkeys()** returns the foreign keys in the table specified in *fk_table* that refer to the primary key of the table specified in *pk_table*. This should be one key at most.

odbc_free_result

(PHP 3>= 3.0.6, PHP 4)

odbc_free_result -- Free resources associated with a result

Description

bool **odbc_free_result** (resource result_id)

Always returns **TRUE**.

odbc_free_result() only needs to be called if you are worried about using too much memory while your script is running. All result memory will automatically be freed when the script is finished. But, if you are sure you are not going to need the result data anymore in a script, you may call **odbc_free_result()**, and the memory associated with *result_id* will be freed.

Note: If auto-commit is disabled (see odbc_autocommit()) and you call **odbc_free_result()** before commiting, all pending transactions are rolled back.

odbc_gettypeinfo

(PHP 4)

odbc_gettypeinfo -- Returns a result identifier containing information about data types supported by the data source.

Description

int **odbc_gettypeinfo** (resource connection_id [, int data_type])

Retrieves information about data types supported by the data source. Returns an ODBC result identifier or **FALSE** on failure. The optional argument *data_type* can be used to restrict the information to a single data type.

The result set has the following columns:

- TYPE_NAME
- DATA_TYPE
- PRECISION
- LITERAL_PREFIX
- LITERAL_SUFFIX
- CREATE_PARAMS
- NULLABLE
- CASE_SENSITIVE
- SEARCHABLE
- UNSIGNED_ATTRIBUTE
- MONEY
- AUTO_INCREMENT
- LOCAL_TYPE_NAME
- MINIMUM_SCALE
- MAXIMUM_SCALE

The result set is ordered by DATA_TYPE and TYPE_NAME.

odbc_longreadlen

(PHP 3>= 3.0.6, PHP 4)

odbc_longreadlen -- Handling of LONG columns

Description

int **odbc_longreadlen** (resource result_id, int length)

(ODBC SQL types affected: LONG, LONGVARBINARY) The number of bytes returned to PHP is controled by the parameter length. If it is set to 0, Long column data is passed thru to the client.

Note: Handling of LONGVARBINARY columns is also affected by odbc_binmode().

odbc_next_result

(PHP 4 >= 4.0.5)

odbc_next_result -- Checks if multiple results are avaiable

Description

bool **odbc_next_result** (resource result_id)

Warning
This function is currently not documented; only the argument list is available.

odbc_num_fields

(PHP 3>= 3.0.6, PHP 4)

odbc_num_fields -- Number of columns in a result

Description

int **odbc_num_fields** (resource result_id)

odbc_num_fields() will return the number of fields (columns) in an ODBC result. This function will return -1 on error. The argument is a valid result identifier returned by odbc_exec().

odbc_num_rows

(PHP 3>= 3.0.6, PHP 4)

odbc_num_rows -- Number of rows in a result

Description

int **odbc_num_rows** (resource result_id)

odbc_num_rows() will return the number of rows in an ODBC result. This function will return -1 on error. For INSERT, UPDATE and DELETE statements **odbc_num_rows()** returns the number of rows affected. For a SELECT clause this can be the number of rows available.

Note: Using **odbc_num_rows()** to determine the number of rows available after a SELECT will return -1 with many drivers.

odbc_pconnect

(PHP 3>= 3.0.6, PHP 4)

odbc_pconnect -- Open a persistent database connection

Description

int **odbc_pconnect** (string dsn, string user, string password [, int cursor_type])

Returns an ODBC connection id or 0 (FALSE) on error. This function is much like odbc_connect(), except that the connection is not really closed when the script has finished. Future requests for a connection with the same *dsn*, *user*, *password* combination (via odbc_connect() and **odbc_pconnect()**) can reuse the persistent connection.

> **Note:** Persistent connections have no effect if PHP is used as a CGI program.

For information about the optional cursor_type parameter see the odbc_connect() function. For more information on persistent connections, refer to the PHP FAQ.

odbc_prepare

(PHP 3>= 3.0.6, PHP 4)

odbc_prepare -- Prepares a statement for execution

Description

resource **odbc_prepare** (resource connection_id, string query_string)

Returns **FALSE** on error.

Returns an ODBC result identifier if the SQL command was prepared successfully. The result identifier can be used later to execute the statement with **odbc_execute()**.

odbc_primarykeys

(PHP 4)

odbc_primarykeys -- Returns a result identifier that can be used to fetch the column names that comprise the primary key for a table

Description

resource **odbc_primarykeys** (resource connection_id, string qualifier, string owner, string table)

Returns the column names that comprise the primary key for a table. Returns an ODBC result identifier or **FALSE** on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- COLUMN_NAME
- KEY_SEQ
- PK_NAME

odbc_procedurecolumns

(PHP 4)

odbc_procedurecolumns -- Retrieve information about parameters to procedures

Description

resource **odbc_procedurecolumns** (resource connection_id [, string qualifier [, string owner [, string proc [, string column]]]])

Returns the list of input and output parameters, as well as the columns that make up the result set for the specified procedures. Returns an ODBC result identifier or **FALSE** on failure.

The result set has the following columns:

- PROCEDURE_QUALIFIER
- PROCEDURE_OWNER
- PROCEDURE_NAME
- COLUMN_NAME
- COLUMN_TYPE
- DATA_TYPE

- TYPE_NAME
- PRECISION
- LENGTH
- SCALE
- RADIX
- NULLABLE
- REMARKS

The result set is ordered by PROCEDURE_QUALIFIER, PROCEDURE_OWNER, PROCEDURE_NAME and COLUMN_TYPE.

The *owner*, *proc* and *column* arguments accept search patterns ('%' to match zero or more characters and '_' to match a single character).

odbc_procedures

(PHP 4)

odbc_procedures -- Get the list of procedures stored in a specific data source. Returns a result identifier containing the information.

Description

resource **odbc_procedures** (resource connection_id [, string qualifier [, string owner [, string name]]])

Lists all procedures in the requested range. Returns an ODBC result identifier or **FALSE** on failure.

The result set has the following columns:

- PROCEDURE_QUALIFIER
- PROCEDURE_OWNER
- PROCEDURE_NAME
- NUM_INPUT_PARAMS
- NUM_OUTPUT_PARAMS
- NUM_RESULT_SETS
- REMARKS
- PROCEDURE_TYPE

The *owner* and *name* arguments accept search patterns ('%' to match zero or more characters and '_' to match a single character).

odbc_result_all

(PHP 3>= 3.0.6, PHP 4)

odbc_result_all -- Print result as HTML table

Description

int **odbc_result_all** (resource result_id [, string format])

Returns the number of rows in the result or **FALSE** on error.

odbc_result_all() will print all rows from a result identifier produced by **odbc_exec()**. The result is printed in HTML table format. With the optional string argument *format*, additional overall table formatting can be done.

odbc_result

(PHP 3>= 3.0.6, PHP 4)

odbc_result -- Get result data

Description

string **odbc_result** (resource result_id, mixed field)

Returns the contents of the field.

field can either be an integer containing the column number of the field you want; or it can be a string containing the name of the field. For example:

```
$item_3 = odbc_result ($Query_ID, 3);
$item_val = odbc_result ($Query_ID, "val");
```

The first call to **odbc_result()** returns the value of the third field in the current record of the query result. The second function call to **odbc_result()** returns the value of the field whose field name is "val" in the current record of the query result. An error occurs if a column number parameter for a field is less than one or exceeds the number of columns (or fields) in the current record. Similarly, an error occurs if a field with a name that is not one of the fieldnames of the table(s) that is(are) being queried.

Field indices start from 1. Regarding the way binary or long column data is returned refer to **odbc_binmode()** and **odbc_longreadlen()**.

odbc_rollback

(PHP 3>= 3.0.6, PHP 4)

odbc_rollback -- Rollback a transaction

Description

int **odbc_rollback** (resource connection_id)

Rolls back all pending statements on *connection_id*. Returns **TRUE** on success, **FALSE** on failure.

odbc_setoption

(PHP 3>= 3.0.6, PHP 4)

odbc_setoption -- Adjust ODBC settings. Returns **FALSE** if an error occurs, otherwise **TRUE**.

Description

int **odbc_setoption** (resource id, int function, int option, int param)

This function allows fiddling with the ODBC options for a particular connection or query result. It was written to help find work arounds to problems in quirky ODBC drivers. You should probably only use this function if you are an ODBC programmer and understand the effects the various options will have. You will certainly need a good ODBC reference to explain all the different options and values that can be used. Different driver versions support different options.

Because the effects may vary depending on the ODBC driver, use of this function in scripts to be made publicly available is strongly discouraged. Also, some ODBC options are not available to this function because they must be set before the connection is established or the query is prepared. However, if on a particular job it can make PHP work so your boss doesn't tell you to use a commercial product, that's all that really matters.

id is a connection id or result id on which to change the settings.For SQLSetConnectOption(), this is a connection id. For SQLSetStmtOption(), this is a result id.

Function is the ODBC function to use. The value should be 1 for SQLSetConnectOption() and 2 for SQLSetStmtOption().

Parameter *option* is the option to set.

Parameter *param* is the value for the given *option*.

Example 1. ODBC Setoption Examples

```
// 1. Option 102 of SQLSetConnectOption() is SQL_AUTOCOMMIT.
//    Value 1 of SQL_AUTOCOMMIT is SQL_AUTOCOMMIT_ON.
//    This example has the same effect as
//    odbc_autocommit($conn, true);

odbc_setoption ($conn, 1, 102, 1);

// 2. Option 0 of SQLSetStmtOption() is SQL_QUERY_TIMEOUT.
//    This example sets the query to timeout after 30 seconds.

$result = odbc_prepare ($conn, $sql);
odbc_setoption ($result, 2, 0, 30);
odbc_execute ($result);
```

odbc_specialcolumns

(PHP 4)

odbc_specialcolumns -- Returns either the optimal set of columns that uniquely identifies a row in the table or columns that are automatically updated when any value in the row is updated by a transaction

Description

resource **odbc_specialcolumns** (resource connection_id, int type, string qualifier, string owner, string table, int scope, int nullable)

When the type argument is SQL_BEST_ROWID, **odbc_specialcolumns()** returns the column or columns that uniquely identify each row in the table.

When the type argument is SQL_ROWVER, **odbc_specialcolumns()** returns the optimal column or set of columns that, by retrieving values from the column or columns, allows any row in the specified table to be uniquely identified.

Returns an ODBC result identifier or **FALSE** on failure.

The result set has the following columns:

- SCOPE
- COLUMN_NAME
- DATA_TYPE
- TYPE_NAME
- PRECISION
- LENGTH
- SCALE
- PSEUDO_COLUMN

The result set is ordered by SCOPE.

odbc_statistics

(PHP 4)

odbc_statistics -- Retrieve statistics about a table

Description

resource **odbc_statistics** (resource connection_id, string qualifier, string owner, string table_name, int unique, int accuracy)

Get statistics about a table and it's indexes. Returns an ODBC result identifier or **FALSE** on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- NON_UNIQUE
- INDEX_QUALIFIER
- INDEX_NAME
- TYPE
- SEQ_IN_INDEX
- COLUMN_NAME
- COLLATION
- CARDINALITY
- PAGES
- FILTER_CONDITION

The result set is ordered by NON_UNIQUE, TYPE, INDEX_QUALIFIER, INDEX_NAME and SEQ_IN_INDEX.

odbc_tableprivileges

(PHP 4)

odbc_tableprivileges -- Lists tables and the privileges associated with each table

Description

int **odbc_tableprivileges** (resource connection_id [, string qualifier [, string owner [, string name]]])

Lists tables in the requested range and the privileges associated with each table. Returns an ODBC result identifier or **FALSE** on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- GRANTOR
- GRANTEE
- PRIVILEGE
- IS_GRANTABLE

The result set is ordered by TABLE_QUALIFIER, TABLE_OWNER and TABLE_NAME.

The *owner* and *name* arguments accept search patterns ('%' to match zero or more characters and '_' to match a single character).

odbc_tables

(PHP 3>= 3.0.17, PHP 4)

odbc_tables -- Get the list of table names stored in a specific data source. Returns a result identifier containing the information.

Description

int **odbc_tables** (resource connection_id [, string qualifier [, string owner [, string name [, string types]]]])

Lists all tables in the requested range. Returns an ODBC result identifier or **FALSE** on failure.

The result set has the following columns:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- TABLE_TYPE
- REMARKS

The result set is ordered by TABLE_TYPE, TABLE_QUALIFIER, TABLE_OWNER and TABLE_NAME.

The *owner* and *name* arguments accept search patterns ('%' to match zero or more characters and '_' to match a single character).

To support enumeration of qualifiers, owners, and table types, the following special semantics for the *qualifier*, *owner*, *name*, and *table_type* are available:

- If *qualifier* is a single percent character (%) and *owner* and *name* are empty strings, then the result set contains a list of valid qualifiers for the data source. (All columns except the TABLE_QUALIFIER column contain NULLs.)
- If *owner* is a single percent character (%) and *qualifier* and *name* are empty strings, then the result set contains a list of valid owners for the data source. (All columns except the TABLE_OWNER column contain NULLs.)
- If *table_type* is a single percent character (%) and *qualifier*, *owner* and *name* are empty strings, then the result set contains a list of valid table types for the data source. (All columns except the TABLE_TYPE column contain NULLs.)

If *table_type* is not an empty string, it must contain a list of comma-separated values for the types of interest; each value may be enclosed in single quotes (') or unquoted. For example, "'TABLE','VIEW'" or "TABLE, VIEW". If the data source does not support a specified table type, **odbc_tables()** does not return any results for that type.

See also **odbc_tableprivileges()** to retrieve associated privileges.

LXX. Object Aggregation/Composition Functions

Warning
This extension is *EXPERIMENTAL*. The behaviour of this extension -- including the names of its functions and anything else documented about this extension -- may change without notice in a future release of PHP. Use this extension at your own risk.

Introduction

In Object Oriented Programming, it is common to see the composition of simple classes (and/or instances) into a more complex one. This is a flexible strategy for building complicated objects and object hierachies and can function as a dynamic alternative to multiple inheritance. There are two ways to perform class (and/or object) composition depending on the relationship between the composed elements: *Association* and *Aggregation*.

An *Association* is a composition of independently constructed and externally visible parts. When we associate classes or objects, each one keeps a reference to the ones it is associated with. When we associate classes statically, one class will contain a reference to an instance of the other class. For example:

Example 1. Class association

```
class DateTime {

   function DateTime() {
       // empty constructor
   }

   function now() {
       return date("Y-m-d H:i:s");
   }
}
```

```
class Report {
   var $_dt = new DateTime();
   // more properties ...

   function Report() {
       // initialization code ...
   }

   function generateReport() {
       $dateTime = $_dt->now();
       // more code ...
   }

   // more methods ...
}

$rep = new Report();
```

We can also associate instances at runtime by passing a reference in a constructor (or any other method), which allow us to dynamically change the association relationship between objects. We will modify the example above to illustrate this point:

Example 2. Object association

```
class DateTime {
   // same as previous example
}

class DateTimePlus {
   var $_format;

   function DateTimePlus($format="Y-m-d H:i:s") {
       $this->_format = $format
   }

   function now() {
       return date($this->_format);
   }
}

class Report {
   var $_dt;    // we'll keep the reference to DateTime here
   // more properties ...

   function Report() {
       // do some initialization
   }

   function setDateTime(&$dt) {
       $this->_dt =& $dt;
   }

   function generateReport() {
       $dateTime = $_dt->now();
       // more code ...
   }

   // more methods ...
}

$rep = new Report();
$dt = new DateTime();
$dtp = new DateTimePlus("l, F j, Y (h:i:s a, T)");

// generate report with simple date for web display
$rep->setDateTime(&$dt);
echo $rep->generateReport();

// later on in the code ...

// generate report with fancy date
$rep->setDateTime(&$dtp);
$output = $rep->generateReport();
// save $output in database
// ... etc ...
```

Aggregation, on the other hand, implies encapsulation (hidding) of the parts of the composition. We can aggregate classes by using a (static) inner class (PHP does not yet support inner classes), in this case the aggregated class definition is not accessible, except through the class that contains it. The aggregation of instances (object aggregation) involves the dynamic creation of subobjects inside an object, in the process, expanding the properties and methods of that object.

Object aggregation is a natural way of representing a whole-part relationship, (for example, molecules are aggregates of atoms), or can be used to obtain an effect equivalent to multiple inheritance, without having to permanently bind a subclass to two or more parent classes and their interfaces. In fact object aggregation can be more flexible, in which we can select what methods or properties to "inherit" in the aggregated object.

Examples

We define 3 classes, each implementing a different storage method:

Example 3. storage_classes.inc

```
<?php
class FileStorage {
    var $data;

    function FileStorage($data) {
        $this->data = $data;
    }
    function write($name) {
        $fp = fopen(name, "w");
        fwrite($fp, $this->data);
        fclose($data);
    }
}

class WDDXStorage {
    var $data;
    var $version = "1.0";
    var $_id; // "private" variable

    function WDDXStorage($data) {
        $this->data = $data;
        $this->_id = $this->_genID();
    }

    function store() {
        if ($this->_id) {
            $pid = wddx_packet_start($this->_id);
            wddx_add_vars($pid, "this->data");
            $packet = wddx_packet_end($pid);
        } else {
            $packet = wddx_serialize_value($this->data);
        }
        $dbh = dba_open("varstore", "w", "gdbm");
        dba_insert(md5(uniqid("",true)), $packet, $dbh);
        dba_close($dbh);
    }

    // a private method
    function _genID() {
        return md5(uniqid(rand(),true));
    }
}

class DBStorage {
    var $data;
    var $dbtype = "mysql";

    function DBStorage($data) {
        $this->data = $data;
    }

    function save() {
        $dbh = mysql_connect();
        mysql_select_db("storage", $dbh);
        $serdata = serialize($this->data);
        mysql_query("insert into vars ('$serdata',now())", $dbh);
        mysql_close($dbh);
    }
}

?>
```

We then instantiate a couple of objects from the defined classes, and perform some aggregations and deaggregations, printing some object information along the way:

Example 4. test_aggregation.php

```
<?php
include "storageclasses.inc";

// some utilty functions

function p_arr($arr) {
    foreach($arr as $k=>$v)
        $out[] = "\t$k => $v";
    return implode("\n", $out);
}

function object_info($obj) {
    $out[] = "Class: ".get_class($obj);
    foreach(get_object_vars($obj) as $var=>$val)
        if (is_array($val))
```

```
            $out[] = "property: $var (array)\n".p_arr($val);
        else
            $out[] = "property: $var = $val";
    foreach(get_class_methods($obj) as $method)
        $out[] = "method: $method";
    return implode("\n", $out);
}


$data = array(M_PI, "kludge != cruft");

// we create some basic objects
$fs = new FileStorage($data);
$ws = new WDDXStorage($data);

// print information on the objects
echo "\$fs object\n";
echo object_info($fs)."\n";
echo "\n\$ws object\n";
echo object_info($ws)."\n";

// do some aggregation

echo "\nLet's aggregate \$fs to the WDDXStorage class\n";
aggregate($fs, "WDDXStorage");
echo "\$fs object\n";
echo object_info($fs)."\n";

echo "\nNow let us aggregate it to the DBStorage class\n";
aggregate($fs, "DBStorage");
echo "\$fs object\n";
echo object_info($fs)."\n";

echo "\nAnd finally deaggregate WDDXStorage\n";
deaggregate($fs, "WDDXStorage");
echo "\$fs object\n";
echo object_info($fs)."\n";

?>
```

We will now consider the output to understand some of the side-effects and limitation of object aggregation in PHP. First, the newly created `$fs` and `$ws` objects give the expected output (according to their respective class declaration). Note that for the purposes of object aggregation, *private elements of a class/object begin with an underscore character ("_")*, even though there is not real distinction between public and private class/object elements in PHP.

```
$fs object
Class: filestorage
property: data (array)
    0 => 3.1415926535898
    1 => kludge != cruft
method: filestorage
method: write

$ws object
Class: wddxstorage
property: data (array)
    0 => 3.1415926535898
    1 => kludge != cruft
property: version = 1.0
property: _id = ID::9bb2b640764d4370eb04808af8b076a5
method: wddxstorage
method: store
method: _genid
```

We then aggregate `$fs` with the **WDDXStorage** class, and print out the object information. We can see now that even though nominally the `$fs` object is still of **FileStorage**, it now has the property `$version`, and the method **store()**, both defined in **WDDXStorage**. One important thing to note is that it has not aggregated the private elements defined in the class, which are present in the `$ws` object. Also absent is the constructor from **WDDXStorage**, which will not be logical to aggegate.

```
Let's aggregate $fs to the WDDXStorage class
$fs object
Class: filestorage
property: data (array)
    0 => 3.1415926535898
    1 => kludge != cruft
property: version = 1.0
method: filestorage
method: write
method: store
```

The proccess of aggregation is cummulative, so when we aggregate `$fs` with the class **DBStorage**, generating an object that can use the storage methods of all the defined classes.

```
Now let us aggregate it to the DBStorage class
$fs object
Class: filestorage
property: data (array)
    0 => 3.1415926535898
    1 => kludge != cruft
```

```
property: version = 1.0
property: dbtype = mysql
method: filestorage
method: write
method: store
method: save
```

Finally, the same way we aggregated properties and methods dynamically, we can also deaggregate them from the object. So, if we deaggregate the class **WDDXStorage** from `$fs`, we will obtain:

```
And deaggregate the WDDXStorage methods and properties
$fs object
Class: filestorage
property: data (array)
    0 => 3.1415926535898
    1 => kludge != cruft
property: dbtype = mysql
method: filestorage
method: write
method: save
```

One point that we have not mentioned above, is that the process of aggregation will not override existing properties or methods in the objects. For example, the class **FileStorage** defines a `$data` property, and the class **WDDXStorage** also defines a similar property which will not override the one in the object acquired during instantiation from the class **FileStorage**.

Table of Contents

aggregate_info

(PHP 5 CVS only)

aggregate_info -- returns an associative array of the methods and properties from each class that has been aggregated to the object.

Description

array **aggregate_info** (object object)

Will return the aggretaion information for a particular object as an associative array of arrays of methods and properties. The key for the main array is the name of the aggregated class.

For example the code below

Example 1. Using aggregate_info()

```
<?php

class Slicer {
    var $vegetable;

    function Slicer($vegetable) {
        $this->vegetable = $vegetable;
    }

    function slice_it($num_cuts) {
        echo "Doing some simple slicing\n";
        for ($i=0; $i < $num_cuts; $i++) {
            // do some slicing
        }
    }
}

class Dicer {
    var $vegetable;
    var $rotation_angle = 90;   // degrees
```

```
    function Dicer($vegetable) {
        $this->vegetable = $vegetable;
    }

    function dice_it($num_cuts) {
        echo "Cutting in one direction\n";
        for ($i=0; $i < $num_cuts; $i++) {
            // do some cutting
        }
        $this->rotate($this->rotation_angle);
        echo "Cutting in a second direction\n";
        for ($i=0; $i < $num_cuts; $i++) {
            // do some more cutting
        }
    }

    function rotate($deg) {
        echo "Now rotating {$this->vegetable} {$deg} degrees\n";
    }

    function _secret_super_dicing($num_cuts) {
        // so secret we cannot show you ;-)
    }
}

$obj = new Slicer('onion');
aggregate($obj, 'Dicer');
print_r(aggregate_info($obj));
?>
```

Will produce the output

```
Array
(
    [dicer] => Array
        (
            [methods] => Array
                (
                    [0] => dice_it
                    [1] => rotate
                )

            [properties] => Array
                (
                    [0] => rotation_angle
                )

        )

)
```

As you can see, all properties and methods of the **Dicer** class have been aggregated into our new object, with the exception of the class constructor and the method `_secret_super_dicing`

See also **aggregate()**, **aggregate_methods()**, **aggregate_methods_by_list()**, **aggregate_methods_by_regexp()**, **aggregate_properties()**, **aggregate_properties_by_list()**, **aggregate_properties_by_regexp()**, **deaggregate()**

aggregate_methods_by_list

(PHP 4 >= 4.2.0)

aggregate_methods_by_list -- selective dynamic class methods aggregation to an object

Description

void **aggregate_methods_by_list** (object object, string class_name, array methods_list [, boolean exclude])

Aggregates methods from a class to an existing object using a list of method names. The optional paramater *exclude* is used to decide whether the list contains the names of methods to include in the aggregation (i.e. *exclude* is **FALSE**, which is the default value), or to exclude from the aggregation (*exclude* is **TRUE**).

The class constructor or methods whose names start with an underscore character (_), which are considered private to the aggregated class, are always excluded.

See also **aggregate()**, **aggregate_info()**, **aggregate_methods()**, **aggregate_methods_by_regexp()**, **aggregate_properties()**, **aggregate_properties_by_list()**, **aggregate_properties_by_regexp()**, **deaggregate()**

aggregate_methods_by_regexp

(PHP 4 >= 4.2.0)

aggregate_methods_by_regexp -- selective class methods aggregation to an object using a regular expression

Description

void **aggregate_methods_by_regexp** (object object, string class_name, string regexp [, boolean exclude])

Aggregates methods from a class to an existing object using a regular expresion to match method names. The optional paramater *exclude* is used to decide whether the regular expression will select the names of methods to include in the aggregation (i.e. *exclude* is **FALSE**, which is the default value), or to exclude from the aggregation (*exclude* is **TRUE**).

The class constructor or methods whose names start with an underscore character (_), which are considered private to the aggregated class, are always excluded.

See also **aggregate()**, **aggregate_info()**, **aggregate_methods()**, **aggregate_methods_by_list()**, **aggregate_properties()**, **aggregate_properties_by_list()**, **aggregate_properties_by_regexp()**, **deaggregate()**

aggregate_methods

(PHP 4 >= 4.2.0)

aggregate_methods -- dynamic class and object aggregation of methods

Description

void **aggregate_methods** (object object, string class_name)

Aggregates all methods defined in a class to an existing object, except for the class constructor, or methods whose names start with an underscore character (_) which are considered private to the aggregated class.

See also **aggregate()**, **aggregate_info()**, **aggregate_methods_by_list()**, **aggregate_methods_by_regexp()**, **aggregate_properties()**, **aggregate_properties_by_list()**, **aggregate_properties_by_regexp()**, **deaggregate()**

aggregate_properties_by_list

(PHP 4 >= 4.2.0)

aggregate_properties_by_list -- selective dynamic class properties aggregation to an object

Description

void **aggregate_properties_by_list** (object object, string class_name, array properties_list [, boolean exclude])

Aggregates properties from a class to an existing object using a list of property names. The optional paramater *exclude* is used to decide whether the list contains the names of class properties to include in the aggregation (i.e. *exclude* is **FALSE**, which is the default value), or to exclude from the aggregation (*exclude* is **TRUE**).

The properties whose names start with an underscore character (_), which are considered private to the aggregated class, are always excluded.

See also **aggregate()**, **aggregate_methods()**, **aggregate_methods_by_list()**, **aggregate_methods_by_regexp()**, **aggregate_properties()**, **aggregate_properties_by_regexp()**, **aggregate_info()**, **deaggregate()**

aggregate_properties_by_regexp

(PHP 4 >= 4.2.0)

aggregate_properties_by_regexp -- selective class properties aggregation to an object using a regular expression

Description

void **aggregate_properties_by_regexp** (object object, string class_name, string regexp [, boolean exclude])

Aggregates properties from a class to an existing object using a regular expresion to match their names. The optional paramater *exclude* is used to decide whether the regular expression will select the names of class properties to include in the aggregation (i.e. *exclude* is **FALSE**, which is the default value), or to exclude from the aggregation (*exclude* is **TRUE**).

The properties whose names start with an underscore character (_), which are considered private to the aggregated class, are always excluded.

See also **aggregate()**, **aggregate_methods()**, **aggregate_methods_by_list()**, **aggregate_methods_by_regexp()**, **aggregate_properties()**, **aggregate_properties_by_list()**, **aggregate_info()**, **deaggregate()**

aggregate_properties

(PHP 4 >= 4.2.0)

aggregate_properties -- dynamic aggregation of class properties to an object

Description

void **aggregate_properties** (object object, string class_name)

Aggregates all properties defined in a class to an existing object, except for properties whose names start with an underscore character (_) which are considered private to the aggregated class.

See also **aggregate()**, **aggregate_methods()**, **aggregate_methods_by_list()**, **aggregate_methods_by_regexp()**, **aggregate_properties_by_list()**, **aggregate_properties_by_regexp()**, **aggregate_info()**, **deaggregate()**

aggregate

(PHP 4 >= 4.2.0)

aggregate -- dynamic class and object aggregation of methods and properties

Description

void **aggregate** (object object, string class_name)

Aggregates methods and properties defined in a class to an existing object. Methods and properties with names starting with an underscore character (_) are considered private to the aggregated class and are not used, constructors are also excluded from the aggregation procedure.

See also **aggregate_info()**, **aggregate_methods()**, **aggregate_methods_by_list()**, **aggregate_methods_by_regexp()**, **aggregate_properties()**, **aggregate_properties_by_list()**, **aggregate_properties_by_regexp()**, **deaggregate()**

aggregation_info

(PHP 4 >= 4.2.0)

aggregation_info -- Synonym for **aggregate_info()**

Description

array **aggregation_info** (object object)

Will return the aggretaion information for a particular object as an associative array of arrays of methods and properties. The key for the main array is the name of the aggregated class.

deaggregate

(PHP 4 >= 4.2.0)

deaggregate -- removes the aggregated methods and properties from an object

Description

void **object_aggregation** (object object [, string class_name])

Removes the methods and properties from classes that were aggregated to an object. If the optional *class_name* parameters is passed, only those methods and properties defined in that class are removed, otherwise all aggregated methods and properties are eliminated.

See also **aggregate()**, **aggregate_methods()**, **aggregate_methods_by_list()**, **aggregate_methods_by_regexp()**, **aggregate_properties()**, **aggregate_properties_by_list()**, **aggregate_properties_by_regexp()**, **aggregate_info()**

LXXI. Oracle 8 functions

Introduction

These functions allow you to access Oracle8 and Oracle7 databases. It uses the Oracle8 Call-Interface (OCI8)

This extension is more flexible than the standard Oracle extension. It supports binding of global and local PHP variables to Oracle placeholders, has full LOB, FILE and ROWID support and allows you to use user-supplied define variables.

Requirements

You will need the Oracle8 client libraries to use this extension.

Before using this extension, make sure that you have set up your Oracle environment variables properly for the Oracle user, as well as your web daemon user. The variables you might need to set are as follows:

- ORACLE_HOME
- ORACLE_SID
- LD_PRELOAD
- LD_LIBRARY_PATH
- NLS_LANG
- ORA_NLS33

After setting up the environment variables for your webserver user, be sure to also add the webserver user (nobody, www) to the oracle group.

If your webserver doesn't start or crashes at startup: Check that Apache is linked with the pthread library:

```
# ldd /www/apache/bin/httpd
    libpthread.so.0 => /lib/libpthread.so.0 (0x4001c000)
    libm.so.6 => /lib/libm.so.6 (0x4002f000)
    libcrypt.so.1 => /lib/libcrypt.so.1 (0x4004c000)
    libdl.so.2 => /lib/libdl.so.2 (0x4007a000)
    libc.so.6 => /lib/libc.so.6 (0x4007e000)
    /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

If the libpthread is not listed you have to reinstall Apache:

```
# cd /usr/src/apache_1.3.xx
# make clean
# LIBS=-lpthread ./config.status
# make
# make install
```

Please note that on some systems like UnixWare it is libthread instead of libpthread. PHP and Apache have to be configured with EXTRA_LIBS=-lthread.

Installation

You have to compile PHP with the option `--with-oci8[=DIR]`, where DIR defaults to your environmment variable ORACLE_HOME.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

`OCI_DEFAULT` (integer)

`OCI_DESCRIBE_ONLY` (integer)

`OCI_COMMIT_ON_SUCCESS` (integer)

`OCI_EXACT_FETCH` (integer)

`SQLT_BFILEE` (integer)

`SQLT_CFILEE` (integer)

`SQLT_CLOB` (integer)

`SQLT_BLOB` (integer)

`SQLT_RDD` (integer)

`OCI_B_SQLT_NTY` (integer)

`OCI_SYSDATE` (integer)

`OCI_B_BFILE` (integer)

`OCI_B_CFILEE` (integer)

`OCI_B_CLOB` (integer)

`OCI_B_BLOB` (integer)

`OCI_B_ROWID` (integer)

`OCI_B_CURSOR` (integer)

`OCI_B_BIN` (integer)

`OCI_FETCHSTATEMENT_BY_COLUMN` (integer)

`OCI_FETCHSTATEMENT_BY_ROW` (integer)

`OCI_ASSOC` (integer)

`OCI_NUM` (integer)

`OCI_BOTH` (integer)

`OCI_RETURN_NULLS` (integer)

OCI_RETURN_LOBS (integer)

OCI_DTYPE_FILE (integer)

OCI_DTYPE_LOB (integer)

OCI_DTYPE_ROWID (integer)

OCI_D_FILE (integer)

OCI_D_LOB (integer)

OCI_D_ROWID (integer)

Examples

Example 1. OCI Hints

```
<?php
// by sergo@bacup.ru

// Use option: OCI_DEFAULT for execute command to delay execution
OCIExecute($stmt, OCI_DEFAULT);

// for retrieve data use (after fetch):

$result = OCIResult($stmt, $n);
if (is_object ($result)) $result = $result->load();

// For INSERT or UPDATE statement use:

$sql = "insert into table (field1, field2) values (field1 = 'value',
 field2 = empty_clob()) returning field2 into :field2";
OCIParse($conn, $sql);
$clob = OCINewDescriptor($conn, OCI_D_LOB);
OCIBindByName ($stmt, ":field2", &$clob, -1, OCI_B_CLOB);
OCIExecute($stmt, OCI_DEFAULT);
$clob->save ("some text");
OCICommit($conn);

?>
```

You can easily access stored procedures in the same way as you would from the commands line.

Example 2. Using Stored Procedures

```
<?php
// by webmaster@remoterealty.com
$sth = OCIParse ( $dbh, "begin sp_newaddress( :address_id, '$firstname',
 '$lastname', '$company', '$address1', '$address2', '$city', '$state',
 '$postalcode', '$country', :error_code );end;" );

// This calls stored procedure sp_newaddress, with :address_id being an
// in/out variable and :error_code being an out variable.
// Then you do the binding:

   OCIBindByName ( $sth, ":address_id", $addr_id, 10 );
   OCIBindByName ( $sth, ":error_code", $errorcode, 10 );
   OCIExecute ( $sth );

?>
```

Table of Contents

OCIBindByName

(PHP 3>= 3.0.4, PHP 4)

OCIBindByName -- Bind a PHP variable to an Oracle Placeholder

Description

bool **OCIBindByName** (int stmt, string ph_name, mixed & variable, int length [, int type])

OCIBindByName() binds the PHP variable *variable* to the Oracle placeholder *ph_name*. Whether it will be used for input or output will be determined run-time, and the necessary storage space will be allocated. The *length* parameter sets the maximum length for the bind. If you set *length* to -1 **OCIBindByName()** will use the current length of *variable* to set the maximum length.

If you need to bind an abstract Datatype (LOB/ROWID/BFILE) you need to allocate it first using **OCINewDescriptor()** function. The *length* is not used for abstract Datatypes and should be set to -1. The *type* variable tells oracle, what kind of descriptor we want to use. Possible values are: OCI_B_FILE (Binary-File), OCI_B_CFILE (Character-File), OCI_B_CLOB (Character-LOB), OCI_B_BLOB (Binary-LOB) and OCI_B_ROWID (ROWID).

Example 1. OCIDefineByName

```
<?php
/* OCIBindByPos example thies@thieso.net (980221)
  inserts 3 records into emp, and uses the ROWID for updating the
  records just after the insert.
*/

$conn = OCILogon("scott","tiger");

$stmt = OCIParse($conn,"insert into emp (empno, ename) ".
                                        "values (:empno,:ename) ".
                                        "returning ROWID into :rid");

$data = array(1111 => "Larry", 2222 => "Bill", 3333 => "Jim");

$rowid = OCINewDescriptor($conn,OCI_D_ROWID);
```

```
OCIBindByName($stmt,":empno",&$empno,32);
OCIBindByName($stmt,":ename",&$ename,32);
OCIBindByName($stmt,":rid",&$rowid,-1,OCI_B_ROWID);

$update = OCIParse($conn,"update emp set sal = :sal where ROWID = :rid");
OCIBindByName($update,":rid",&$rowid,-1,OCI_B_ROWID);
OCIBindByName($update,":sal",&$sal,32);

$sal = 10000;

while (list($empno,$ename) = each($data)) {
        OCIExecute($stmt);
        OCIExecute($update);
}

$rowid->free();

OCIFreeStatement($update);
OCIFreeStatement($stmt);

$stmt = OCIParse($conn,"select * from emp where empno in (1111,2222,3333)");
OCIExecute($stmt);
while (OCIFetchInto($stmt,&$arr,OCI_ASSOC)) {
        var_dump($arr);
}
OCIFreeStatement($stmt);

/* delete our "junk" from the emp table.... */
$stmt = OCIParse($conn,"delete from emp where empno in (1111,2222,3333)");
OCIExecute($stmt);
OCIFreeStatement($stmt);

OCILogoff($conn);
?>
```

Warning
It is a bad idea to use magic quotes and **OciBindByName()** simultaneously as no quoting is needed on quoted variables and any quotes magically applied will be written into your database as **OciBindByName()** is not able to distinguish magically added quotings from those added by intention.

OCICancel

(PHP 3>= 3.0.8, PHP 4)

OCICancel -- Cancel reading from cursor

Description

bool **OCICancel** (int stmt)

If you do not want read more data from a cursor, then call **OCICancel()**.

OCICollAppend

(PHP 4 >= 4.0.6)

OCICollAppend -- Coming soon

Description

bool **OCICollAppend** (object collection, object object)

Warning
This function is currently not documented; only the argument list is available.

OCICollAssign

(PHP 4 >= 4.0.6)

OCICollAssign -- Coming soon

Description

bool **OCICollAssign** (object collection, object object)

Warning
This function is currently not documented; only the argument list is available.

OCICollAssignElem

(PHP 4 >= 4.0.6)

OCICollAssignElem -- Coming soon

Description

bool **OCICollAssignElem** (object collection, string ndx, string val)

Warning
This function is currently not documented; only the argument list is available.

OCICollGetElem

(PHP 4 >= 4.0.6)

OCICollGetElem -- Coming soon

Description

string **OCICollGetElem** (object collection, string ndx)

Warning
This function is currently not documented; only the argument list is available.

OCICollMax

(PHP 4 >= 4.0.6)

OCICollMax -- Coming soon

Description

int **OCICollMax** (object collection)

Warning
This function is currently not documented; only the argument list is available.

OCICollSize

(PHP 4 >= 4.0.6)

OCICollSize -- Coming soon

Description

int **OCICollSize** (object collection)

Warning
This function is currently not documented; only the argument list is available.

OCICollTrim

(PHP 4 >= 4.0.6)

OCICollTrim -- Coming soon

Description

bool **OCICollTrim** (object collection, int num)

Warning
This function is currently not documented; only the argument list is available.

OCIColumnIsNULL

(PHP 3>= 3.0.4, PHP 4)

OCIColumnIsNULL -- Test whether a result column is **NULL**

Description

bool **OCIColumnIsNULL** (int stmt, mixed column)

OCIColumnIsNULL() returns **TRUE** if the returned column *column* in the result from the statement *stmt* is **NULL**. You can either use the column-number (1-Based) or the column-name for the *col* parameter.

OCIColumnName

(PHP 3>= 3.0.4, PHP 4)

OCIColumnName -- Returns the name of a column

Description

string **OCIColumnName** (int stmt, int col)

OCIColumnName() returns the name of the column corresponding to the column number (1-based) that is passed in.

Example 1. OCIColumnName

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn,"select * from emp");
    OCIExecute($stmt);
    print "<TABLE BORDER=\"1\">";
    print "<TR>";
    print "<TH>Name</TH>";
    print "<TH>Type</TH>";
    print "<TH>Length</TH>";
    print "</TR>";
    $ncols = OCINumCols($stmt);
    for ( $i = 1; $i <= $ncols; $i++ ) {
        $column_name  = OCIColumnName($stmt,$i);
        $column_type  = OCIColumnType($stmt,$i);
        $column_size  = OCIColumnSize($stmt,$i);
        print "<TR>";
        print "<TD>$column_name</TD>";
        print "<TD>$column_type</TD>";
        print "<TD>$column_size</TD>";
        print "</TR>";
    }
    print "</TABLE>\n";
```

```
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

See also **OCINumCols()**, **OCIColumnType()**, and **OCIColumnSize()**.

OCIColumnPrecision

(PHP 4)

OCIColumnPrecision -- Coming soon

Description

int **OCIColumnPrecision** (int stmt, int col)

Warning
This function is currently not documented; only the argument list is available.

OCIColumnScale

(PHP 4)

OCIColumnScale -- Coming soon

Description

int **OCIColumnScale** (int stmt, int col)

Warning
This function is currently not documented; only the argument list is available.

OCIColumnSize

(PHP 3>= 3.0.4, PHP 4)

OCIColumnSize -- Return result column size

Description

int **OCIColumnSize** (int stmt, mixed column)

OCIColumnSize() returns the size of the column as given by Oracle. You can either use the column-number (1-Based) or the column-name for the *col* parameter.

Example 1. OCIColumnSize

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn,"select * from emp");
    OCIExecute($stmt);
    print "<TABLE BORDER=\"1\">";
    print "<TR>";
    print "<TH>Name</TH>";
    print "<TH>Type</TH>";
    print "<TH>Length</TH>";
    print "</TR>";
    $ncols = OCINumCols($stmt);
    for ( $i = 1; $i <= $ncols; $i++ ) {
        $column_name  = OCIColumnName($stmt,$i);
        $column_type  = OCIColumnType($stmt,$i);
        $column_size  = OCIColumnSize($stmt,$i);
```

```
        print "<TR>";
        print "<TD>$column_name</TD>";
        print "<TD>$column_type</TD>";
        print "<TD>$column_size</TD>";
        print "</TR>";
    }
    print "</TABLE>";
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

See also **OCINumCols()**, **OCIColumnName()**, and **OCIColumnSize()**.

OCIColumnType

(PHP 3>= 3.0.4, PHP 4)

OCIColumnType -- Returns the data type of a column

Description

mixed **OCIColumnType** (int stmt, int col)

OCIColumnType() returns the data type of the column corresponding to the column number (1-based) that is passed in.

Example 1. OCIColumnType

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn,"select * from emp");
    OCIExecute($stmt);
    print "<TABLE BORDER=\"1\">";
    print "<TR>";
    print "<TH>Name</TH>";
    print "<TH>Type</TH>";
    print "<TH>Length</TH>";
    print "</TR>";
    $ncols = OCINumCols($stmt);
    for ( $i = 1; $i <= $ncols; $i++ ) {
        $column_name  = OCIColumnName($stmt,$i);
        $column_type  = OCIColumnType($stmt,$i);
        $column_size  = OCIColumnSize($stmt,$i);
        print "<TR>";
        print "<TD>$column_name</TD>";
        print "<TD>$column_type</TD>";
        print "<TD>$column_size</TD>";
        print "</TR>";
    }
    print "</TABLE>\n";
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

See also **OCINumCols()**, **OCIColumnName()**, and **OCIColumnSize()**.

OCIColumnTypeRaw

(PHP 4)

OCIColumnTypeRaw -- Coming soon

Description

mixed **OCIColumnTypeRaw** (int stmt, int col)

Warning
This function is currently not documented; only the argument list is available.

OCICommit

(PHP 3>= 3.0.7, PHP 4)

OCICommit -- Commits outstanding transactions

Description

bool **OCICommit** (int connection)

OCICommit() commits all outstanding statements for Oracle connection *connection*. Returns **TRUE** on success or **FALSE** on failure.

This example demonstrates how OCICommit is used.

Example 1. OCICommit

```
<?php
    // Login to Oracle server
    $conn = OCILogon('scott', 'tiger');

    // Parse SQL
    $stmt = OCIParse($conn, "INSERT INTO employees (name, surname) VALUES ('Maxim', 'Maletsky')");

    // Execute statement
    OCIExecute($stmt);

    // Commit transaction
    $committed = OCICommit($conn);

    // Test whether commit was successful. If error occurred, return error message
    if(!$committed) {
        $error = OCIError($conn);
        echo 'Commit failed. Oracle reports: ' . $error['message'];
    }

    // Close connection
    OCILogoff($conn);
?>
```

See also OCIRollback().

OCIDefineByName

(PHP 3>= 3.0.7, PHP 4)

OCIDefineByName -- Use a PHP variable for the define-step during a SELECT

Description

bool **OCIDefineByName** (int stmt, string Column-Name, mixed variable [, int type])

OCIDefineByName() binds PHP variables for fetches of SQL-Columns. Be careful that Oracle uses ALL-UPPERCASE column-names, whereby in your select you can also write lowercase. **OCIDefineByName()** expects the *Column-Name* to be in uppercase. If you define a variable that doesn't exists in you select statement, no error will be given!

If you need to define an abstract datatype (LOB/ROWID/BFILE) you need to allocate it first using OCINewDescriptor() function. See also the OCIBindByName() function.

Example 1. OCIDefineByName

```
<?php
/* OCIDefineByName example - thies@thieso.net (980219) */

$conn = OCILogon("scott","tiger");

$stmt = OCIParse($conn,"select empno, ename from emp");

/* the define MUST be done BEFORE ociexecute! */

OCIDefineByName($stmt,"EMPNO",$empno);
OCIDefineByName($stmt,"ENAME",$ename);

OCIExecute($stmt);
```

```
while (OCIFetch($stmt)) {
    echo "empno:".$empno."\n";
    echo "ename:".$ename."\n";
}

OCIFreeStatement($stmt);
OCILogoff($conn);
?>
```

OCIError

(PHP 3>= 3.0.7, PHP 4)

OCIError -- Return the last error of stmt|conn|global

Description

array **OCIError** ([int stmt|conn|global])

OCIError() returns the last error found. If the optional *stmt|conn|global* is not provided, the last error encountered is returned. If no error is found, **OCIError()** returns **FALSE**. **OCIError()** returns the error as an associative array. In this array, *code* consists the oracle error code and *message* the oracle errorstring.

OCIExecute

(PHP 3>= 3.0.4, PHP 4)

OCIExecute -- Execute a statement

Description

int **OCIExecute** (int statement [, int mode])

OCIExecute() executes a previously parsed statement. (see **OCIParse()**). The optional *mode* allows you to specify the execution-mode (default is OCI_COMMIT_ON_SUCCESS). If you don't want statements to be committed automatically specify OCI_DEFAULT as your mode.

Returns **TRUE** on success or **FALSE** on failure.

OCIFetch

(PHP 3>= 3.0.4, PHP 4)

OCIFetch -- Fetches the next row into result-buffer

Description

bool **OCIFetch** (int statement)

OCIFetch() fetches the next row (for SELECT statements) into the internal result-buffer.

OCIFetchInto

(PHP 3>= 3.0.4, PHP 4)

OCIFetchInto -- Fetches the next row into result-array

Description

int **OCIFetchInto** (int stmt, array &result [, int mode])

OCIFetchInto() fetches the next row (for SELECT statements) into the *result* array. **OCIFetchInto()** will overwrite the previous

content of *result*. By default *result* will contain a zero-based array of all columns that are not **NULL**.

The *mode* parameter allows you to change the default behaviour. You can specify more than one flag by simply adding them up (eg OCI_ASSOC+OCI_RETURN_NULLS). The known flags are:

OCI_ASSOC Return an associative array.
OCI_NUM Return an numbered array starting with zero. (DEFAULT)
OCI_RETURN_NULLS Return empty columns.
OCI_RETURN_LOBS Return the value of a LOB instead of the descriptor.

Example 1. A simple OCIFetchInto() example

```
<?php
$conn = ocilogon("username","password");

$query = "SELECT apples FROM oranges";

$statement = OCIParse ($conn, $query);
OCIExecute ($statement);

while (OCIFetchInto ($statement, $row, OCI_ASSOC)) {
    print $row['apples'];
}
?>
```

See also **OCIFetch()** and **OCIExecute()**.

OCIFetchStatement

(PHP 3>= 3.0.8, PHP 4)

OCIFetchStatement -- Fetch all rows of result data into an array

Description

int **OCIFetchStatement** (int stmt, array & variable)

OCIFetchStatement() fetches all the rows from a result into a user-defined array. **OCIFetchStatement()** returns the number of rows fetched.

Example 1. OCIFetchStatement

```
<?php
/* OCIFetchStatement example mbritton@verinet.com (990624) */

$conn = OCILogon("scott","tiger");

$stmt = OCIParse($conn,"select * from emp");

OCIExecute($stmt);

$nrows = OCIFetchStatement($stmt,$results);
if ( $nrows > 0 ) {
   print "<TABLE BORDER=\"1\">\n";
   print "<TR>\n";
   while ( list( $key, $val ) = each( $results ) ) {
      print "<TH>$key</TH>\n";
   }
   print "</TR>\n";

   for ( $i = 0; $i < $nrows; $i++ ) {
      reset($results);
      print "<TR>\n";
      while ( $column = each($results) ) {
         $data = $column['value'];
         print "<TD>$data[$i]</TD>\n";
      }
      print "</TR>\n";
   }
   print "</TABLE>\n";
} else {
   echo "No data found<BR>\n";
}
print "$nrows Records Selected<BR>\n";

OCIFreeStatement($stmt);
OCILogoff($conn);
?>
```

OCIFreeCollection

(PHP 4 >= 4.1.0)

OCIFreeCollection -- Coming soon

Description

bool **OCIFreeCollection** (object lob)

Warning
This function is currently not documented; only the argument list is available.

OCIFreeCursor

(PHP 3>= 3.0.8, PHP 4)

OCIFreeCursor -- Free all resources associated with a cursor

Description

int **OCIFreeCursor** (int stmt)

OCIFreeCursor() returns **TRUE** if successful, or **FALSE** if unsuccessful.

OCIFreeDesc

(PHP 4)

OCIFreeDesc -- Deletes a large object descriptor

Description

bool **OCIFreeDesc** (object lob)

OCIFreeDesc() returns **TRUE** if successful, or **FALSE** if unsuccessful.

OCIFreeStatement

(PHP 3>= 3.0.5, PHP 4)

OCIFreeStatement -- Free all resources associated with a statement

Description

bool **OCIFreeStatement** (int stmt)

OCIFreeStatement() returns **TRUE** if successful, or **FALSE** if unsuccessful.

OCIInternalDebug

(PHP 3>= 3.0.4, PHP 4)

OCIInternalDebug -- Enables or disables internal debug output

Description

void **OCIInternalDebug** (int onoff)

OCIInternalDebug() enables internal debug output. Set *onoff* to 0 to turn debug output off, 1 to turn it on.

OCILoadLob

(PHP 4)

OCILoadLob -- Coming soon

Description

string **OCILoadLob** (object lob)

Warning
This function is currently not documented; only the argument list is available.

OCILogOff

(PHP 3>= 3.0.4, PHP 4)

OCILogOff -- Disconnects from Oracle

Description

bool **OCILogOff** (int connection)

OCILogOff() closes an Oracle connection.

Using **OCILogOff()** isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution. See also freeing resources.

OCILogon

(PHP 3>= 3.0.4, PHP 4)

OCILogon -- Establishes a connection to Oracle

Description

int **OCILogon** (string username, string password [, string db])

OCILogon() returns an connection identifier needed for most other OCI calls. The optional third parameter can either contain the name of the local Oracle instance or the name of the entry in tnsnames.ora to which you want to connect. If the optional third parameter is not specified, PHP uses the environment variables ORACLE_SID (Oracle instance) or TWO_TASK (tnsnames.ora) to determine which database to connect to.

Connections are shared at the page level when using **OCILogon()**. This means that commits and rollbacks apply to all open transactions in the page, even if you have created multiple connections.

This example demonstrates how the connections are shared.

Example 1. OCILogon

```
<?php
print "<HTML><PRE>";
$db = "";

$c1 = ocilogon("scott","tiger",$db);
$c2 = ocilogon("scott","tiger",$db);

function create_table($conn)
{ $stmt = ociparse($conn,"create table scott.hallo (test varchar2(64))");
  ociexecute($stmt);
```

```
  echo $conn." created table\n\n";
}

function drop_table($conn)
{ $stmt = ociparse($conn,"drop table scott.hallo");
  ociexecute($stmt);
  echo $conn." dropped table\n\n";
}

function insert_data($conn)
{ $stmt = ociparse($conn,"insert into scott.hallo
           values('$conn' || ' ' || to_char(sysdate,'DD-MON-YY HH24:MI:SS'))");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn." inserted hallo\n\n";
}

function delete_data($conn)
{ $stmt = ociparse($conn,"delete from scott.hallo");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn." deleted hallo\n\n";
}

function commit($conn)
{ ocicommit($conn);
  echo $conn." committed\n\n";
}

function rollback($conn)
{ ocirollback($conn);
  echo $conn." rollback\n\n";
}

function select_data($conn)
{ $stmt = ociparse($conn,"select * from scott.hallo");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn."----selecting\n\n";
  while (ocifetch($stmt))
    echo $conn." <".ociresult($stmt,"TEST").">\n\n";
  echo $conn."----done\n\n";
}

create_table($c1);
insert_data($c1);   // Insert a row using c1
insert_data($c2);   // Insert a row using c2

select_data($c1);   // Results of both inserts are returned
select_data($c2);

rollback($c1);      // Rollback using c1

select_data($c1);   // Both inserts have been rolled back
select_data($c2);

insert_data($c2);   // Insert a row using c2
commit($c2);        // commit using c2

select_data($c1);   // result of c2 insert is returned

delete_data($c1);   // delete all rows in table using c1
select_data($c1);   // no rows returned
select_data($c2);   // no rows returned
commit($c1);        // commit using c1

select_data($c1);   // no rows returned
select_data($c2);   // no rows returned

drop_table($c1);
print "</PRE></HTML>";
?>
```

See also **OCIPLogon()** and **OCINLogon()**.

OCINewCollection

(PHP 4 >= 4.0.6)

OCINewCollection -- Coming soon

Description

bool **OCINewCollection** (int conn, string tdo [, string shema])

Warning
This function is currently not documented; only the argument list is available.

OCINewCursor

(PHP 3>= 3.0.8, PHP 4)

OCINewCursor -- Return a new cursor (Statement-Handle)

Description

int **OCINewCursor** (int conn)

OCINewCursor() allocates a new statement handle on the specified connection.

Example 1. Using a REF CURSOR from a stored procedure

```
<?php
// suppose your stored procedure info.output returns a ref cursor in :data

$conn = OCILogon("scott","tiger");
$curs = OCINewCursor($conn);
$stmt = OCIParse($conn,"begin info.output(:data); end;");

ocibindbyname($stmt,"data",&$curs,-1,OCI_B_CURSOR);
ociexecute($stmt);
ociexecute($curs);

while (OCIFetchInto($curs,&$data)) {
    var_dump($data);
}

OCIFreeStatement($stmt);
OCIFreeCursor($curs);
OCILogoff($conn);
?>
```

Example 2. Using a REF CURSOR in a select statement

```
<?php
print "<HTML><BODY>";
$conn = OCILogon("scott","tiger");
$count_cursor = "CURSOR(select count(empno) num_emps from emp " .
                "where emp.deptno = dept.deptno) as EMPCNT from dept";
$stmt = OCIParse($conn,"select deptno,dname,$count_cursor");

ociexecute($stmt);
print "<TABLE BORDER=\"1\">";
print "<TR>";
print "<TH>DEPT NAME</TH>";
print "<TH>DEPT #</TH>";
print "<TH># EMPLOYEES</TH>";
print "</TR>";

while (OCIFetchInto($stmt,&$data,OCI_ASSOC)) {
    print "<TR>";
    $dname  = $data["DNAME"];
    $deptno = $data["DEPTNO"];
    print "<TD>$dname</TD>";
    print "<TD>$deptno</TD>";
    ociexecute($data["EMPCNT"]);
    while (OCIFetchInto($data["EMPCNT"],&$subdata,OCI_ASSOC)) {
        $num_emps = $subdata["NUM_EMPS"];
        print  "<TD>$num_emps</TD>";
    }
    print "</TR>";
}
print "</TABLE>";
print "</BODY></HTML>";
OCIFreeStatement($stmt);
OCILogoff($conn);
?>
```

OCINewDescriptor

(PHP 3>= 3.0.7, PHP 4)

OCINewDescriptor -- Initialize a new empty LOB or FILE descriptor

Description

string **OCINewDescriptor** (int connection [, int type])

OCINewDescriptor() allocates storage to hold descriptors or LOB locators. Valid values for *type* are OCI_D_FILE, OCI_D_LOB, OCI_D_ROWID. For LOB descriptors, the methods load, save, and savefile are associated with the descriptor, for BFILE only the load method exists. See the second example usage hints.

Example 1. OCINewDescriptor

```
<?php
    /* This script is designed to be called from a HTML form.
     * It expects $user, $password, $table, $where, and $commitsize
     * to be passed in from the form.  The script then deletes
     * the selected rows using the ROWID and commits after each
     * set of $commitsize rows. (Use with care, there is no rollback)
     */
    $conn = OCILogon($user, $password);
    $stmt = OCIParse($conn,"select rowid from $table $where");
    $rowid = OCINewDescriptor($conn,OCI_D_ROWID);
    OCIDefineByName($stmt,"ROWID",&$rowid);
    OCIExecute($stmt);
    while ( OCIFetch($stmt) ) {
       $nrows = OCIRowCount($stmt);
       $delete = OCIParse($conn,"delete from $table where ROWID = :rid");
       OCIBindByName($delete,":rid",&$rowid,-1,OCI_B_ROWID);
       OCIExecute($delete);
       print "$nrows\n";
       if ( ($nrows % $commitsize) == 0 ) {
           OCICommit($conn);
       }
    }
    $nrows = OCIRowCount($stmt);
    print "$nrows deleted...\n";
    OCIFreeStatement($stmt);
    OCILogoff($conn);
?>

<?php
    /* This script demonstrates file upload to LOB columns
     * The formfield used for this example looks like this
     * <form action="upload.php" method="post" enctype="multipart/form-data">
     * <input type="file" name="lob_upload">
     * ...
     */
  if(!isset($lob_upload) || $lob_upload == 'none'){
?>
<form action="upload.php" method="post" enctype="multipart/form-data">
Upload file: <input type="file" name="lob_upload"><br>
<input type="submit" value="Upload"> - <input type="reset">
</form>
<?php
  } else {

     // $lob_upload contains the temporary filename of the uploaded file

     // see also the features section on file upload,
     // if you would like to use secure uploads

     $conn = OCILogon($user, $password);
     $lob = OCINewDescriptor($conn, OCI_D_LOB);
     $stmt = OCIParse($conn,"insert into $table (id, the_blob)
              values(my_seq.NEXTVAL, EMPTY_BLOB()) returning the_blob into :the_blob");
     OCIBindByName($stmt, ':the_blob', &$lob, -1, OCI_B_BLOB);
     OCIExecute($stmt, OCI_DEFAULT);
     if($lob->savefile($lob_upload)){
        OCICommit($conn);
        echo "Blob successfully uploaded\n";
     }else{
        echo "Couldn't upload Blob\n";
     }
     OCIFreeDesc($lob);
     OCIFreeStatement($stmt);
     OCILogoff($conn);
  }
?>
```

Example 2. OCINewDescriptor

```
<?php
    /* Calling PL/SQL stored procedures which contain clobs as input
     * parameters (PHP 4 >= 4.0.6).
     * Example PL/SQL stored procedure signature is:
     *
     * PROCEDURE save_data
     *   Argument Name                  Type                    In/Out Default?
     *   ------------------------------ ----------------------- ------ --------
     *   KEY                            NUMBER(38)              IN
     *   DATA                           CLOB                    IN
     *
     */
```

```
    $conn = OCILogon($user, $password);
    $stmt = OCIParse($conn, "begin save_data(:key, :data); end;");
    $clob = OCINewDescriptor($conn, OCI_D_LOB);
        OCIBindByName($stmt, ':key', $key);
        OCIBindByName($stmt, ':data', $clob, -1, OCI_B_CLOB);
        $clob->WriteTemporary($data);
        OCIExecute($stmt, OCI_DEFAULT);
        OCICommit($conn);
        $clob->close();
        $clob->free();
        OCIFreeStatement($stmt);
?>
```

OCINLogon

(PHP 3>= 3.0.8, PHP 4)

OCINLogon -- Establishes a new connection to Oracle

Description

int **OCINLogon** (string username, string password [, string db])

OCINLogon() creates a new connection to an Oracle 8 database and logs on. The optional third parameter can either contain the name of the local Oracle instance or the name of the entry in tnsnames.ora to which you want to connect. If the optional third parameter is not specified, PHP uses the environment variables ORACLE_SID (Oracle instance) or TWO_TASK (tnsnames.ora) to determine which database to connect to.

OCINLogon() forces a new connection. This should be used if you need to isolate a set of transactions. By default, connections are shared at the page level if using **OCILogon()** or at the web server process level if using **OCIPLogon()**. If you have multiple connections open using **OCINLogon()**, all commits and rollbacks apply to the specified connection only.

This example demonstrates how the connections are separated.

Example 1. OCINLogon

```
<?php
print "<HTML><PRE>";
$db = "";

$c1 = ocilogon("scott","tiger",$db);
$c2 = ocinlogon("scott","tiger",$db);

function create_table($conn)
{ $stmt = ociparse($conn,"create table scott.hallo (test
varchar2(64))");
  ociexecute($stmt);
  echo $conn." created table\n\n";
}

function drop_table($conn)
{ $stmt = ociparse($conn,"drop table scott.hallo");
  ociexecute($stmt);
  echo $conn." dropped table\n\n";
}

function insert_data($conn)
{ $stmt = ociparse($conn,"insert into scott.hallo
          values('$conn' || ' ' || to_char(sysdate,'DD-MON-YY HH24:MI:SS'))");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn." inserted hallo\n\n";
}

function delete_data($conn)
{ $stmt = ociparse($conn,"delete from scott.hallo");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn." deleted hallo\n\n";
}

function commit($conn)
{ ocicommit($conn);
  echo $conn." committed\n\n";
}

function rollback($conn)
{ ocirollback($conn);
  echo $conn." rollback\n\n";
}

function select_data($conn)
{ $stmt = ociparse($conn,"select * from scott.hallo");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn."----selecting\n\n";
  while (ocifetch($stmt))
```

```
    echo $conn." <".ociresult($stmt,"TEST").">\n\n";
  echo $conn."----done\n\n";
}

create_table($c1);
insert_data($c1);

select_data($c1);
select_data($c2);

rollback($c1);

select_data($c1);
select_data($c2);

insert_data($c2);
commit($c2);

select_data($c1);

delete_data($c1);
select_data($c1);
select_data($c2);
commit($c1);

select_data($c1);
select_data($c2);

drop_table($c1);
print "</PRE></HTML>";
?>
```

See also **OCILogon()** and **OCIPLogon()**.

OCINumCols

(PHP 3>= 3.0.4, PHP 4)

OCINumCols -- Return the number of result columns in a statement

Description

int **OCINumCols** (int stmt)

OCINumCols() returns the number of columns in a statement.

Example 1. OCINumCols

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn,"select * from emp");
    OCIExecute($stmt);
    while ( OCIFetch($stmt) ) {
        print "\n";
        $ncols = OCINumCols($stmt);
        for ( $i = 1; $i <= $ncols; $i++ ) {
            $column_name  = OCIColumnName($stmt,$i);
            $column_value = OCIResult($stmt,$i);
            print $column_name . ': ' . $column_value . "\n";
        }
        print "\n";
    }
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

OCIParse

(PHP 3>= 3.0.4, PHP 4)

OCIParse -- Parse a query and return a statement

Description

int **OCIParse** (int conn, string query)

OCIParse() parses the *query* using *conn*. It returns the statement identity if the *query* is valid, **FALSE** if not. The *query* can be any valid SQL statement or PL/SQL block.

OCIPLogon

(PHP 3>= 3.0.8, PHP 4)

OCIPLogon -- Connect to an Oracle database using a persistent connection

Description

int **OCIPLogon** (string username, string password [, string db])

OCIPLogon() creates a persistent connection to an Oracle 8 database and logs on. The optional third parameter can either contain the name of the local Oracle instance or the name of the entry in tnsnames.ora to which you want to connect. If the optional third parameter is not specified, PHP uses the environment variables ORACLE_SID (Oracle instance) or TWO_TASK (tnsnames.ora) to determine which database to connect to.

See also **OCILogon()** and **OCINLogon()**.

OCIResult

(PHP 3>= 3.0.4, PHP 4)

OCIResult -- Returns column value for fetched row

Description

mixed **OCIResult** (int statement, mixed column)

OCIResult() returns the data for column *column* in the current row (see **OCIFetch()**).**OCIResult()** will return everything as strings except for abstract types (ROWIDs, LOBs and FILEs).

OCIRollback

(PHP 3>= 3.0.7, PHP 4)

OCIRollback -- Rolls back outstanding transactions

Description

bool **OCIRollback** (int connection)

OCIRollback() rolls back all outstanding statements for Oracle connection *connection*. **OCIRollback()** returns **TRUE** on success and **FALSE** otherwise.

See also **OCICommit()**.

OCIRowCount

(PHP 3>= 3.0.7, PHP 4)

OCIRowCount -- Gets the number of affected rows

Description

int **OCIRowCount** (int statement)

OCIRowCount() returns the number of rows affected for eg update-statements. This function will not tell you the number of rows that a select will return!

Example 1. OCIRowCount

```
<?php
    print "<HTML><PRE>";
    $conn = OCILogon("scott","tiger");
    $stmt = OCIParse($conn,"create table emp2 as select * from emp");
    OCIExecute($stmt);
    print OCIRowCount($stmt) . " rows inserted.<BR>";
    OCIFreeStatement($stmt);
    $stmt = OCIParse($conn,"delete from emp2");
    OCIExecute($stmt);
    print OCIRowCount($stmt) . " rows deleted.<BR>";
    OCICommit($conn);
    OCIFreeStatement($stmt);
    $stmt = OCIParse($conn,"drop table emp2");
    OCIExecute($stmt);
    OCIFreeStatement($stmt);
    OCILogOff($conn);
    print "</PRE></HTML>";
?>
```

OCISaveLob

(PHP 4)

OCISaveLob -- Coming soon

Description

bool **OCISaveLob** (object lob)

Warning
This function is currently not documented; only the argument list is available.

OCISaveLobFile

(PHP 4)

OCISaveLobFile -- Coming soon

Description

bool **OCISaveLobFile** (object lob)

Warning
This function is currently not documented; only the argument list is available.

OCIServerVersion

(PHP 3>= 3.0.4, PHP 4)

OCIServerVersion -- Return a string containing server version information

Description

string **OCIServerVersion** (int conn)

Example 1. OCIServerVersion

```
<?php
   $conn = OCILogon("scott","tiger");
```

```
    print "Server Version: " . OCIServerVersion($conn);
    OCILogOff($conn);
 ?>
```

OCISetPrefetch

(PHP 3>= 3.0.12, PHP 4)

OCISetPrefetch -- Sets number of rows to be prefetched

Description

int **OCISetPrefetch** (int stmt, int rows)

Sets the number of top level rows to be prefetched. The default value is 1 row.

OCIStatementType

(PHP 3>= 3.0.5, PHP 4)

OCIStatementType -- Return the type of an OCI statement

Description

string **OCIStatementType** (int stmt)

OCIStatementType() returns one of the following values:

1. "SELECT"
2. "UPDATE"
3. "DELETE"
4. "INSERT"
5. "CREATE"
6. "DROP"
7. "ALTER"
8. "BEGIN"
9. "DECLARE"
10. "UNKNOWN"

Example 1. OCIStatementType() examples

```
 <?php
     print "<HTML><PRE>";
     $conn = OCILogon("scott","tiger");
     $sql  = "delete from emp where deptno = 10";

     $stmt = OCIParse($conn,$sql);
     if ( OCIStatementType($stmt) == "DELETE" ) {
          die "You are not allowed to delete from this table<BR>";
     }

     OCILogoff($conn);
     print "</PRE></HTML>";
 ?>
```

OCIWriteLobToFile

(PHP 4)

OCIWriteLobToFile -- Coming soon

Description

bool **OCIWriteLobToFile** (object lob [, string filename [, int start [, int lenght]]])

Warning
This function is currently not documented; only the argument list is available.

LXXII. OpenSSL functions

Introduction

This module uses the functions of OpenSSL for generation and verification of signatures and for sealing (encrypting) and opening (decrypting) data. OpenSSL offers many features that this module currently doesn't support. Some of these may be added in the future.

Requirements

In order to use the OpenSSL functions you need to install the OpenSSL package. PHP-4.0.4pl1 requires OpenSSL >= 0.9.6, but PHP-4.0.5 and greater will also work with OpenSSL >= 0.9.5.

Installation

To use PHP's OpenSSL support you must also compile PHP `--with-openssl[=DIR]`.

> **Note to Win32 Users:** In order to enable this module on a Windows environment, you must copy *libeay32.dll* from the DLL folder of the PHP/Win32 binary package to the SYSTEM32 folder of your windows machine. (Ex: C:\WINNT\SYSTEM32 or C:\WINDOWS\SYSTEM32)
>
> Additionally, if you are planning to use the key generation and certificate signing functions, you will need to install a valid `openssl.cnf` on your system. As of PHP 4.3.0, we include a sample configuration file in the `openssl` of our win32 binary distribution. If you are using PHP 4.2.0 or later and are missing the file, you can obtain it from the OpenSSL home page or by downloading the PHP 4.3.0 release and using the configuration file from there.
>
> **Note to Win32 Users:** PHP will search for the `openssl.cnf` using the following logic:
>
> - the `OPENSSL_CONF` environmental variable, if set, will be used as the path (including filename) of the configuration file.
> - the `SSLEAY_CONF` environmental variable, if set, will be used as the path (including filename) of the configuration file.
> - The file `openssl.cnf` will be assumed to be found in the default certificate area, as configured at the time that the openssl DLL was compiled. This is usually means that the default filename is `c:\usr\local\ssl\openssl.cnf`.
>
> In your installation, you need to decide whether to install the configuration file at `c:\usr\local\ssl\openssl.cnf` or whether to install it someplace else and use environmental variables (possibly on a per-virtual-host basis) to locate the configuration file. Note that it is possible to override the default path from the script using the *`configargs`* of the functions that require a configuration file.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

Key/Certificate parameters

Quite a few of the openssl functions require a key or a certificate parameter. PHP 4.0.5 and earlier have to use a key or certificate **resource** returned by one of the openssl_get_xxx functions. Later versions may use one of the following methods:

- Certificates
  1. An X.509 resource returned from **openssl_x509_read()**
  2. A string having the format `file://path/to/cert.pem`; the named file must contain a PEM encoded certificate
  3. A string containing the content of a certificate, PEM encoded
- Public/Private Keys
  1. A key resource returned from **openssl_get_publickey()** or **openssl_get_privatekey()**
  2. For public keys only: an X.509 resource
  3. A string having the format `file://path/to/file.pem` - the named file must contain a PEM encoded certificate/private key (it may contain both)
  4. A string containing the content of a certificate/key, PEM encoded
  5. For private keys, you may also use the syntax *array($key, $passphrase)* where $key represents a key specified using the file:// or textual content notation above, and $passphrase represents a string containing the passphrase for that private key

---

Certificate Verification

When calling a function that will verify a signature/certificate, the *cainfo* parameter is an array containing file and directory names that specify the locations of trusted CA files. If a directory is specified, then it must be a correctly formed hashed directory as the **openssl** command would use.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

`X509_PURPOSE_SSL_CLIENT` (integer)

`X509_PURPOSE_SSL_SERVER` (integer)

`X509_PURPOSE_NS_SSL_SERVER` (integer)

`X509_PURPOSE_SMIME_SIGN` (integer)

`X509_PURPOSE_SMIME_ENCRYPT` (integer)

`X509_PURPOSE_CRL_SIGN` (integer)

`X509_PURPOSE_ANY` (integer)

`OPENSSL_PKCS1_PADDING` (integer)

`OPENSSL_SSLV23_PADDING` (integer)

`OPENSSL_NO_PADDING` (integer)

`OPENSSL_PKCS1_OAEP_PADDING` (integer)

OPENSSL_KEYTYPE_RSA (integer)

OPENSSL_KEYTYPE_DSA (integer)

OPENSSL_KEYTYPE_DH (integer)

PKCS7 Flags/Constants

The S/MIME functions make use of flags which are specified using a bitfield which can include one or more of the following values:

Table 1. PKCS7 CONSTANTS

Constant	Description
PKCS7_TEXT	adds text/plain content type headers to encrypted/signed message. If decrypting or verifying, it strips those headers from the output - if the decrypted or verified message is not of MIME type text/plain then an error will occur.
PKCS7_BINARY	normally the input message is converted to "canonical" format which is effectively using CR and LF as end of line: as required by the S/MIME specification. When this options is present, no translation occurs. This is useful when handling binary data which may not be in MIME format.
PKCS7_NOINTERN	when verifying a message, certificates (if any) included in the message are normally searched for the signing certificate. With this option only the certificates specified in the *extracerts* parameter of **openssl_pkcs7_verify()** are used. The supplied certificates can still be used as untrusted CAs however.
PKCS7_NOVERIFY	do not verify the signers certificate of a signed message.
PKCS7_NOCHAIN	do not chain verification of signers certificates: that is don't use the certificates in the signed message as untrusted CAs.
PKCS7_NOCERTS	when signing a message the signer's certificate is normally included - with this option it is excluded. This will reduce the size of the signed message but the verifier must have a copy of the signers certificate available locally (passed using the *extracerts* to **openssl_pkcs7_verify()** for example.
PKCS7_NOATTR	normally when a message is signed, a set of attributes are included which include the signing time and the supported symmetric algorithms. With this option they are not included.
PKCS7_DETACHED	When signing a message, use cleartext signing with the MIME type multipart/signed. This is the default if the *flags* parameter to **openssl_pkcs7_sign()** if you do not specify any flags. If you turn this option off, the message will be signed using opaque signing, which is more resistant to translation by mail relays but cannot be read by mail agents that do not support S/MIME.
PKCS7_NOSIGS	Don't try and verify the signatures on a message

Note: These constants were added in 4.0.6.

Table of Contents

openssl_x509_check_private_key -- Checks if a private key corresponds to a certificate
openssl_x509_checkpurpose -- Verifies if a certificate can be used for a particular purpose
openssl_x509_export_to_file -- Exports a certificate to file
openssl_x509_export -- Exports a certificate as a string
openssl_x509_free -- Free certificate resource
openssl_x509_parse -- Parse an X509 certificate and return the information as an array
openssl_x509_read -- Parse an X.509 certificate and return a resource identifier for it

openssl_csr_export_to_file

(PHP 4 >= 4.2.0)

openssl_csr_export_to_file -- Exports a CSR to a file

Description

bool **openssl_csr_export_to_file** (resource csr, string outfilename [, bool notext])

openssl_csr_export_to_file() takes the Certificate Signing Request represented by *csr* and saves it as ascii-armoured text into the file named by *outfilename*. Returns **TRUE** on success or **FALSE** on failure. The optional *notext* affects the verbosity of the output; if it is **FALSE** then additional human-readable information is included in the output. The default value of *notext* is **TRUE**

See also **openssl_csr_export()**, **openssl_csr_new()** and **openssl_csr_sign()**.

openssl_csr_export

(PHP 4 >= 4.2.0)

openssl_csr_export -- Exports a CSR as a string

Description

bool **openssl_csr_export** (resource csr, string &out [, bool notext])

openssl_csr_export() takes the Certificate Signing Request represented by *csr* and stores it as ascii-armoured text into *out*, which is passed by reference. Returns **TRUE** on success or **FALSE** on failure. The optional *notext* affects the verbosity of the output; if it is **FALSE** then additional human-readable information is included in the output. The default value of *notext* is **TRUE**

See also **openssl_csr_export_to_file()**, **openssl_csr_new()** and **openssl_csr_sign()**.

openssl_csr_new

(PHP 4 >= 4.2.0)

openssl_csr_new -- Generates a CSR

Description

bool **openssl_csr_new** (array dn, resource privkey [, array configargs [, array extraattribs]])

openssl_csr_new() generates a new CSR (Certificate Signing Request) based on the information provided by *dn*, which represents the Distinguished Name to be used in the certificate.

privkey should be set to a private key that was previously generated by **openssl_pkey_new()** (or otherwise obtained from the other openssl_pkey family of functions). The corresponding public portion of the key will be used to sign the CSR.

extraattribs is used to specify additional configuration options for the CSR. Both *dn* and *extraattribs* are associative arrays whose keys are converted to OIDs and applied to the relevant part of the request.

> **Note:** You need to have a valid `openssl.cnf` installed for this function to operate correctly. See the notes under the installation section for more information.

By default, the information in your system `openssl.conf` is used to initialize the request; you can specify a configuration file

section by setting the `config_section_section` key of *`configargs`*. You can also specify and alternative openssl configuration file by setting the `config` key to the path of the file you want to use. The following keys, if present in *`configargs`* behave as their equivalents in the `openssl.conf`, as listed in the table below.

Table 1. Configuration overrides

***configargs* key**	**type**	**openssl.conf equivalent**	**description**
digest_alg	string	default_md	Selects which digest method to use
x509_extensions	string	x509_extensions	Selects which extensions should be used when creating an x509 certificate
req_extensions	string	req_extensions	Selects which extensions should be used when creating a CSR
private_key_bits	integer	default_bits	Specifies how many bits should be used to generate a private key
private_key_type	integer	none	Specifies the type of private key to create. This can be one of **OPENSSL_KEYTYPE_DSA**, **OPENSSL_KEYTYPE_DH** or **OPENSSL_KEYTYPE_RSA**. The default value is **OPENSSL_KEYTYPE_RSA** which is currently the only supported key type.
encrypt_key	booean	encrypt_key	Should an exported key (with passphrase) be encrypted?

Returns **TRUE** on success or **FALSE** on failure.

Example 1. openssl_csr_new() example - creating a self-signed-certificate

```
// Fill in data for the distinguished name to be used in the cert
// You must change the values of these keys to match your name and
// company, or more precisely, the name and company of the person/site
// that you are generating the certificate for.
// For SSL certificates, the commonName is usually the domain name of
// that will be using the certificate, but for S/MIME certificates,
// the commonName will be the name of the individual who will use the
// certificate.
$dn = array(
    "countryName" => "UK",
    "stateOrProvinceName" => "Somerset",
    "localityName" => "Glastonbury",
    "organizationName" => "The Brain Room Limited",
    "organizationalUnitName" => "PHP Documentation Team",
    "commonName" => "Wez Furlong",
    "emailAddress" => "wez@php.net"
);

// Generate a new private (and public) key pair
$privkey = openssl_pkey_new();

// Generate a certificate signing request
$csr = openssl_csr_new($dn, $privkey);

// You will usually want to create a self-signed certificate at this
// point until your CA fulfills your request.
// This creates a self-signed cert that is valid for 365 days
$sscert = openssl_csr_sign($csr, null, $privkey, 365);

// Now you will want to preserve your private key, CSR and self-signed
// cert so that they can be installed into your web server, mail server
// or mail client (depending on the intended use of the certificate).
// This example shows how to get those things into variables, but you
// can also store them directly into files.
// Typically, you will send the CSR on to your CA who will then issue
// you with the "real" certificate.
openssl_csr_export($csr, $csrout) and debug_zval_dump($csrout);
openssl_x509_export($sscert, $certout) and debug_zval_dump($certout);
openssl_pkey_export($privkey, $pkeyout, "mypassword") and debug_zval_dump($pkeyout);

// Show any errors that occurred here
while (($e = openssl_error_string()) !== false) {
    echo $e . "\n";
}
```

openssl_csr_sign

(PHP 4 >= 4.2.0)

openssl_csr_sign -- Sign a CSR with another certificate (or itself) and generate a certificate

Description

resource **openssl_csr_sign** (mixed csr, mixed cacert, mixed priv_key, long days)

openssl_csr_sign() generates an x509 certificate resource from the *csr* previously generated by **openssl_csr_new()**, but it can also be the path to a PEM encoded CSR when specified as `file://path/to/csr` or an exported string generated by **openssl_csr_export()**. The generated certificate will be signed by *cacert*. If *cacert* is **NULL**, the generated certificate will be a self-signed certificate. *priv_key* is the private key that corresponds to *cacert*. *days* specifies the length of time for which the generated certificate will be valid, in days.

Returns an x509 certificate resource on success, **FALSE** on failure.

> **Note:** You need to have a valid `openssl.cnf` installed for this function to operate correctly. See the notes under the installation section for more information.

Example 1. openssl_csr_sign() example - signing a CSR (how to implement your own CA)

```
// Let's assume that this script is set to receive a CSR that has
// been pasted into a textarea from another page
$csrdata = $_POST["CSR"];

// We will sign the request using our own "certificate authority"
// certificate.  You can use any certificate to sign another, but
// the process is worthless unless the signing certificate is trusted
// by the software/users that will deal with the newly signed certificate

// We need our CA cert and it's private key
$cacert = "file://path/to/ca.crt";
$privkey = array("file://path/to/ca.key", "your_ca_key_passphrase");

$userscert = openssl_csr_sign($csrdata, $cacert, $privkey, 365);

// Now display the generated certificate so that the user can
// copy and paste it into their local configuration (such as a file
// to hold the certificate for their SSL server)
openssl_x509_export($usercert, $certout) and echo $certout;

// Show any errors that occurred here
while (($e = openssl_error_string()) !== false) {
    echo $e . "\n";
}
```

openssl_error_string

(PHP 4 >= 4.0.6)

openssl_error_string -- Return openSSL error message

Description

mixed **openssl_error_string** (void)

Returns an error message string, or **FALSE** if there are no more error messages to return.

openssl_error_string() returns the last error from the openSSL library. Error messages are stacked, so this function should be called multiple times to collect all of the information.

Example 1. openssl_error_string() example

```
// lets assume you just called an openssl function that failed
while($msg = openssl_error_string())
    echo $msg . "<br />\n";
```

> **Note:** This function was added in 4.0.6.

openssl_free_key

(PHP 4 >= 4.0.4)

openssl_free_key -- Free key resource

Description

void **openssl_free_key** (resource key_identifier)

openssl_free_key() frees the key associated with the specified *key_identifier* from memory.

openssl_get_privatekey

(PHP 4 >= 4.0.4)

openssl_get_privatekey -- Get a private key

Description

resource **openssl_get_privatekey** (mixed key [, string passphrase])

This is an alias for **openssl_pkey_get_private()**.

openssl_get_publickey

(PHP 4 >= 4.0.4)

openssl_get_publickey -- Extract public key from certificate and prepare it for use

Description

resource **openssl_get_publickey** (mixed certificate)

This is an alias for **openssl_pkey_get_public()**.

openssl_open

(PHP 4 >= 4.0.4)

openssl_open -- Open sealed data

Description

bool **openssl_open** (string sealed_data, string open_data, string env_key, mixed priv_key_id)

Returns **TRUE** on success or **FALSE** on failure. If successful the opened data is returned in *open_data*.

openssl_open() opens (decrypts) *sealed_data* using the private key associated with the key identifier *priv_key_id* and the envelope key *env_key*, and fills *open_data* with the decrypted data. The envelope key is generated when the data are sealed and can only be used by one specific private key. See **openssl_seal()** for more information.

Example 1. openssl_open() example

```
// $sealed and $env_key are assumed to contain the sealed data
// and our envelope key, both given to us by the sealer.

// fetch private key from file and ready it
$fp = fopen("/src/openssl-0.9.6/demos/sign/key.pem", "r");
$priv_key = fread($fp, 8192);
fclose($fp);
$pkeyid = openssl_get_privatekey($priv_key);

// decrypt the data and store it in $open
if (openssl_open($sealed, $open, $env_key, $pkeyid))
    echo "here is the opened data: ", $open;
else
    echo "failed to open data";

// free the private key from memory
openssl_free_key($pkeyid);
```

See also **openssl_seal()**.

openssl_pkcs7_decrypt

(PHP 4 >= 4.0.6)

openssl_pkcs7_decrypt -- Decrypts an S/MIME encrypted message

Description

bool **openssl_pkcs7_decrypt** (string infilename, string outfilename, mixed recipcert [, mixed recipkey])

Decrypts the S/MIME encrypted message contained in the file specified by *infilename* using the certificate and it's associated private key specified by *recipcert* and *recipkey*.

The decrypted message is output to the file specified by *outfilename*

Example 1. openssl_pkcs7_decrypt() example

```
// $cert and $key are assumed to contain your personal certificate and private
// key pair, and that you are the recipient of an S/MIME message
$infilename = "encrypted.msg";  // this file holds your encrypted message
$outfilename = "decrypted.msg"; // make sure you can write to this file

if (openssl_pkcs7_decrypt($infilename, $outfilename, $cert, $key))
    echo "decrypted!";
else
    echo "failed to decrypt!";
```

Note: This function was added in 4.0.6.

openssl_pkcs7_encrypt

(PHP 4 >= 4.0.6)

openssl_pkcs7_encrypt -- Encrypt an S/MIME message

Description

bool **openssl_pkcs7_encrypt** (string infile, string outfile, mixed recipcerts, array headers [, long flags])

openssl_pkcs7_encrypt() takes the contents of the file named *infile* and encrypts them using an RC2 40-bit cipher so that they can only be read by the intended recipients specified by *recipcerts*, which is either a lone X.509 certificate, or an array of X.509 certificates. *headers* is an array of headers that will be prepended to the data after it has been encrypted. *flags* can be used to specify options that affect the encoding process - see PKCS7 constants. *headers* can be either an associative array keyed by header name, or an indexed array, where each element contains a single header line.

Example 1. openssl_pkcs7_encrypt() example

```
// the message you want to encrypt and send to your secret agent
// in the field, known as nighthawk.  You have his certificate
// in the file nighthawk.pem
$data = <<<EOD
Nighthawk,

Top secret, for your eyes only!

The enemy is closing in! Meet me at the cafe at 8.30am
to collect your forged passport!

HQ
EOD;

// load key
$key = file_get_contents("nighthawk.pem");

// save message to file
$fp = fopen("msg.txt", "w");
fwrite($fp, $data);
fclose($fp);

// encrypt it
if (openssl_pkcs7_encrypt("msg.txt", "enc.txt", $key,
    array("To" => "nighthawk@example.com", // keyed syntax
          "From: HQ <hq@example.com>", // indexed syntax
          "Subject" => "Eyes only")))
{
    // message encrypted - send it!
    exec(ini_get("sendmail_path") . " < enc.txt");
```

```
}
```

openssl_pkcs7_sign

(PHP 4 >= 4.0.6)

openssl_pkcs7_sign -- sign an S/MIME message

Description

bool **openssl_pkcs7_sign** (string infilename, string outfilename, mixed signcert, mixed privkey, array headers [, long flags [, string extracertsfilename]])

openssl_pkcs7_sign() takes the contents of the file named *infilename* and signs them using the certificate and it's matching private key specified by *signcert* and *privkey* parameters.

headers is an array of headers that will be prepended to the data after it has been signed (see **openssl_pkcs7_encrypt()** for more information about the format of this parameter.

flags can be used to alter the output - see PKCS7 constants - if not specified, it defaults to PKCS7_DETACHED.

extracerts specifies the name of a file containing a bunch of extra certificates to include in the signature which can for example be used to help the recipient to verify the certificate that you used.

Example 1. openssl_pkcs7_sign() example

```
// the message you want to sign so that recipient can be sure it was you that
// sent it
$data = <<<EOD

You have my authorization to spend $10,000 on dinner expenses.

The CEO
EOD;
// save message to file
$fp = fopen("msg.txt", "w");
fwrite($fp, $data);
fclose($fp);
// encrypt it
if (openssl_pkcs7_sign("msg.txt", "signed.txt", "mycert.pem",
    array("mycert.pem", "mypassphrase"),
    array("To" => "joes@sales.com", // keyed syntax
          "From: HQ <ceo@sales.com>", // indexed syntax
          "Subject" => "Eyes only"))
{
    // message signed - send it!
    exec(ini_get("sendmail_path") . " < signed.txt");
}
```

Note: This function was added in 4.0.6.

openssl_pkcs7_verify

(PHP 4 >= 4.0.6)

openssl_pkcs7_verify -- Verifies the signature of an S/MIME signed message

Description

bool **openssl_pkcs7_verify** (string filename, int flags [, string outfilename [, array cainfo [, string extracerts]]])

openssl_pkcs7_verify() reads the S/MIME message contained in the filename specified by *filename* and examines the digital signature. It returns **TRUE** if the signature is verified, **FALSE** if it is not correct (the message has been tampered with, or the signing certificate is invalid), or -1 on error.

flags can be used to affect how the signature is verified - see PKCS7 constants for more information.

If the *outfilename* is specified, it should be a string holding the name of a file into which the certificates of the persons that signed the messages will be stored in PEM format.

If the *cainfo* is specified, it should hold information about the trusted CA certificates to use in the verification process - see

certificate verification for more information about this parameter.

If the *extracerts* is specified, it is the filename of a file containing a bunch of certificates to use as untrusted CAs.

> **Note:** This function was added in 4.0.6.

openssl_pkey_export_to_file

(PHP 4 >= 4.2.0)

openssl_pkey_export_to_file -- Gets an exportable representation of a key into a file

Description

bool **openssl_pkey_export_to_file** (mixed key, string outfilename [, string passphrase [, array configargs]])

openssl_pkey_export_to_file() saves an ascii-armoured (PEM encoded) rendition of *key* into the file named by *outfilename*. The key can be optionally protected by a *passphrase*. *configargs* can be used to fine-tune the export process by specifying and/or overriding options for the openssl configuration file. See **openssl_csr_new()** for more information about *configargs*. Returns **TRUE** on success or **FALSE** on failure.

> **Note:** You need to have a valid `openssl.cnf` installed for this function to operate correctly. See the notes under the installation section for more information.

openssl_pkey_export

(PHP 4 >= 4.2.0)

openssl_pkey_export -- Gets an exportable representation of a key into a string

Description

bool **openssl_pkey_export** (mixed key, string &out [, string passphrase [, array configargs]])

openssl_pkey_export() exports *key* as a PEM encoded string and stores it into *out* (which is passed by reference). The key is optionally protected by *passphrase*. *configargs* can be used to fine-tune the export process by specifying and/or overriding options for the openssl configuration file. See **openssl_csr_new()** for more information about *configargs*. Returns **TRUE** on success or **FALSE** on failure.

> **Note:** You need to have a valid `openssl.cnf` installed for this function to operate correctly. See the notes under the installation section for more information.

openssl_pkey_get_private

(PHP 4 >= 4.2.0)

openssl_pkey_get_private -- Get a private key

Description

resource **openssl_get_privatekey** (mixed key [, string passphrase])

Returns a positive key resource identifier on success, or **FALSE** on error.

openssl_get_privatekey() parses *key* and prepares it for use by other functions. *key* can be one of the following:

1. a string having the format `file://path/to/file.pem`. The named file must contain a PEM encoded certificate/private key (it may contain both).
2. A PEM formatted private key.

The optional parameter *passphrase* must be used if the specified key is encrypted (protected by a passphrase).

openssl_pkey_get_public

(PHP 4 >= 4.2.0)

openssl_pkey_get_public -- Extract public key from certificate and prepare it for use

Description

resource **openssl_pkey_get_public** (mixed certificate)

Returns a positive key resource identifier on success, or **FALSE** on error.

openssl_get_publickey() extracts the public key from *certificate* and prepares it for use by other functions. *certificate* can be one of the following:

1. an X.509 certificate resource
2. a string having the format `file://path/to/file.pem`. The named file must contain a PEM encoded certificate/private key (it may contain both).
3. A PEM formatted private key.

openssl_pkey_new

(PHP 4 >= 4.2.0)

openssl_pkey_new -- Generates a new private key

Description

resource **openssl_pkey_new** ([array configargs])

openssl_pkey_new() generates a new private and public key pair. The public component of the key can be obtained using **openssl_pkey_get_public()**. You can finetune the key generation (such as specifying the number of bits) using *configargs*. See **openssl_csr_new()** for more information about *configargs*.

> **Note:** You need to have a valid `openssl.cnf` installed for this function to operate correctly. See the notes under the installation section for more information.

openssl_private_decrypt

(PHP 4 >= 4.0.6)

openssl_private_decrypt -- Decrypts data with private key

Description

bool **openssl_private_decrypt** (string data, string &decrypted, mixed key [, int padding])

openssl_private_decrypt() decrypts *data* that was previous encrypted via **openssl_private_encrypt()** and stores the result into *decrypted*. *key* must be the private key corresponding that was used to encrypt the data. *padding* defaults to OPENSSL_PKCS1_PADDING, but can also be one of OPENSSL_SSLV23_PADDING, OPENSSL_PKCS1_OAEP_PADDING OPENSSL_NO_PADDING.

openssl_private_encrypt

(PHP 4 >= 4.0.6)

openssl_private_encrypt -- Encrypts data with private key

Description

bool **openssl_private_encrypt** (string data, string crypted, mixed key [, int padding])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

openssl_public_decrypt

(PHP 4 >= 4.0.6)

openssl_public_decrypt -- Decrypts data with public key

Description

bool **openssl_public_decrypt** (string data, string crypted, resource key [, int padding])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

openssl_public_encrypt

(PHP 4 >= 4.0.6)

openssl_public_encrypt -- Encrypts data with public key

Description

bool **openssl_public_encrypt** (string data, string crypted, mixed key [, int padding])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

openssl_seal

(PHP 4 >= 4.0.4)

openssl_seal -- Seal (encrypt) data

Description

int **openssl_seal** (string data, string sealed_data, array env_keys, array pub_key_ids)

Returns the length of the sealed data on success, or **FALSE** on error. If successful the sealed data is returned in *sealed_data*, and the envelope keys in *env_keys*.

openssl_seal() seals (encrypts) *data* by using RC4 with a randomly generated secret key. The key is encrypted with each of the

public keys associated with the identifiers in *pub_key_ids* and each encrypted key is returned in *env_keys*. This means that one can send sealed data to multiple recipients (provided one has obtained their public keys). Each recipient must receive both the sealed data and the envelope key that was encrypted with the recipient's public key.

Example 1. openssl_seal() example

```
// $data is assumed to contain the data to be sealed

// fetch public keys for our recipients, and ready them
$fp = fopen("/src/openssl-0.9.6/demos/maurice/cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
$pk1 = openssl_get_publickey($cert);
// Repeat for second recipient
$fp = fopen("/src/openssl-0.9.6/demos/sign/cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
$pk2 = openssl_get_publickey($cert);

// seal message, only owners of $pk1 and $pk2 can decrypt $sealed with keys
// $ekeys[0] and $ekeys[1] respectively.
openssl_seal($data, $sealed, $ekeys, array($pk1,$pk2));

// free the keys from memory
openssl_free_key($pk1);
openssl_free_key($pk2);
```

See also **openssl_open()**.

openssl_sign

(PHP 4 >= 4.0.4)

openssl_sign -- Generate signature

Description

bool **openssl_sign** (string data, string signature, mixed priv_key_id)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Returns **TRUE** on success or **FALSE** on failure. If successful the signature is returned in *signature*.

openssl_sign() computes a signature for the specified *data* by using SHA1 for hashing followed by encryption using the private key associated with *priv_key_id*. Note that the data itself is not encrypted.

Example 1. openssl_sign() example

```
// $data is assumed to contain the data to be signed

// fetch private key from file and ready it
$fp = fopen("/src/openssl-0.9.6/demos/sign/key.pem", "r");
$priv_key = fread($fp, 8192);
fclose($fp);
$pkeyid = openssl_get_privatekey($priv_key);

// compute signature
openssl_sign($data, $signature, $pkeyid);

// free the key from memory
openssl_free_key($pkeyid);
```

See also **openssl_verify()**.

openssl_verify

(PHP 4 >= 4.0.4)

openssl_verify -- Verify signature

Description

int **openssl_verify** (string data, string signature, mixed pub_key_id)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Returns 1 if the signature is correct, 0 if it is incorrect, and -1 on error.

openssl_verify() verifies that the *signature* is correct for the specified *data* using the public key associated with *pub_key_id*. This must be the public key corresponding to the private key used for signing.

Example 1. openssl_verify() example

```
// $data and $signature are assumed to contain the data and the signature

// fetch public key from certificate and ready it
$fp = fopen("/src/openssl-0.9.6/demos/sign/cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
$pubkeyid = openssl_get_publickey($cert);

// state whether signature is okay or not
$ok = openssl_verify($data, $signature, $pubkeyid);
if ($ok == 1)
    echo "good";
elseif ($ok == 0)
    echo "bad";
else
    echo "ugly, error checking signature";

// free the key from memory
openssl_free_key($pubkeyid);
```

See also **openssl_sign()**.

openssl_x509_check_private_key

(PHP 4 >= 4.2.0)

openssl_x509_check_private_key -- Checks if a private key corresponds to a certificate

Description

bool **openssl_x509_check_private_key** (mixed cert, mixed key)

openssl_x509_check_private_key() returns **TRUE** if *key* is the private key that corresponds to *cert*, or **FALSE** otherwise.

openssl_x509_checkpurpose

(PHP 4 >= 4.0.6)

openssl_x509_checkpurpose -- Verifies if a certificate can be used for a particular purpose

Description

bool **openssl_x509_checkpurpose** (mixed x509cert, int purpose, array cainfo [, string untrustedfile])

Returns **TRUE** if the certificate can be used for the intended purpose, **FALSE** if it cannot, or -1 on error.

openssl_x509_checkpurpose() examines the certificate specified by *x509cert* to see if it can be used for the purpose specified by *purpose*.

cainfo should be an array of trusted CA files/dirs as described in Certificate Verification.

untrustedfile, if specified, is the name of a PEM encoded file holding certificates that can be used to help verify the certificate, although no trust in placed in the certificates that come from that file.

Table 1. openssl_x509_checkpurpose() purposes

Constant	Description
X509_PURPOSE_SSL_CLIENT	Can the certificate be used for the client side of an SSL connection?
X509_PURPOSE_SSL_SERVER	Can the certificate be used for the server side of an SSL connection?
X509_PURPOSE_NS_SSL_SERVER	Can the cert be used for Netscape SSL server?
X509_PURPOSE_SMIME_SIGN	Can the cert be used to sign S/MIME email?
X509_PURPOSE_SMIME_ENCRYPT	Can the cert be used to encrypt S/MIME email?
X509_PURPOSE_CRL_SIGN	Can the cert be used to sign a certificate revocation list (CRL)?
X509_PURPOSE_ANY	Can the cert be used for Any/All purposes?

These options are not bitfields - you may specify one only!

Note: This function was added in 4.0.6.

openssl_x509_export_to_file

(PHP 4 >= 4.2.0)

openssl_x509_export_to_file -- Exports a certificate to file

Description

bool **openssl_x509_export_to_file** (mixed x509, string outfilename [, bool notext])

openssl_x509_export_to_file() stores *x509* into a file named by *outfilename* in a PEM encoded format. The optional parameter *notext* default to **TRUE**. If set to **FALSE**, additional human readable text will also be stored into the output file. Returns **TRUE** on success or **FALSE** on failure.

openssl_x509_export

(PHP 4 >= 4.2.0)

openssl_x509_export -- Exports a certificate as a string

Description

bool **openssl_x509_export** (mixed x509, string &output [, bool notext])

openssl_x509_export() stores *x509* into a file named by *outfilename* in a PEM encoded format. The optional parameter *notext* default to **TRUE**. If set to **FALSE**, additional human readable text will also be stored into *output*. Returns **TRUE** on success or **FALSE** on failure.

openssl_x509_free

(PHP 4 >= 4.0.6)

openssl_x509_free -- Free certificate resource

Description

void **openssl_x509_free** (resource x509cert)

openssl_x509_free() frees the certificate associated with the specified *x509cert* resource from memory.

Note: This function was added in 4.0.6.

openssl_x509_parse

(PHP 4 >= 4.0.6)

openssl_x509_parse -- Parse an X509 certificate and return the information as an array

Description

array **openssl_x509_parse** (mixed x509cert [, bool shortnames])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

openssl_x509_parse() returns information about the supplied *x509cert*, including fields such as subject name, issuer name, purposes, valid from and valid to dates etc. *shortnames* controls how the data is indexed in the array - if *shortnames* is **TRUE** (the default) then fields will be indexed with the short name form, otherwise, the long name form will be used - e.g.: CN is the shortname form of commonName.

The structure of the returned data is (deliberately) not yet documented, as it is still subject to change.

> **Note:** This function was added in 4.0.6.

openssl_x509_read

(PHP 4 >= 4.0.6)

openssl_x509_read -- Parse an X.509 certificate and return a resource identifier for it

Description

resource **openssl_x509_read** (mixed x509certdata)

openssl_x509_read() parses the certificate supplied by *x509certdata* and returns a resource identifier for it.

> **Note:** This function was added in 4.0.6.

LXXIII. Oracle functions

Introduction

This extension adds support for Oracle database server access. See also the OCI8 extension.

Installation

You have to compile PHP with the option `--with-oracle[=DIR]`, where DIR defaults to your environmment variable ORACLE_HOME.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

ORA_BIND_INOUT (integer)

ORA_BIND_IN (integer)

ORA_BIND_OUT (integer)

ORA_FETCHINTO_ASSOC (integer)

ORA_FETCHINTO_NULLS (integer)

Table of Contents

Ora_Bind

(PHP 3, PHP 4)

Ora_Bind -- bind a PHP variable to an Oracle parameter

Description

int **ora_bind** (int cursor, string PHP variable name, string SQL parameter name, int length [, int type])

Returns **TRUE** if the bind succeeds, otherwise **FALSE**. Details about the error can be retrieved using the **ora_error()** and **ora_errorcode()** functions.

This function binds the named PHP variable with a SQL parameter. The SQL parameter must be in the form ":name". With the optional type parameter, you can define whether the SQL parameter is an in/out (0, default), in (1) or out (2) parameter. As of PHP 3.0.1, you can use the constants ORA_BIND_INOUT, ORA_BIND_IN and ORA_BIND_OUT instead of the numbers.

ora_bind must be called after **ora_parse()** and before **ora_exec()**. Input values can be given by assignment to the bound PHP variables, after calling **ora_exec()** the bound PHP variables contain the output values if available.

```
<?php
ora_parse($curs, "declare tmp INTEGER; begin tmp := :in; :out := tmp; :x := 7.77; end;");
ora_bind($curs, "result", ":x", $len, 2);
ora_bind($curs, "input", ":in", 5, 1);
ora_bind($curs, "output", ":out", 5, 2);
$input = 765;
ora_exec($curs);
echo "Result: $result<BR>Out: $output<BR>In: $input";
?>
```

Ora_Close

(PHP 3, PHP 4)

Ora_Close -- close an Oracle cursor

Description

int **ora_close** (int cursor)

Returns **TRUE** if the close succeeds, otherwise **FALSE**. Details about the error can be retrieved using the ora_error() and ora_errorcode() functions.

This function closes a data cursor opened with ora_open().

Ora_ColumnName

(PHP 3, PHP 4)

Ora_ColumnName -- get name of Oracle result column

Description

string **Ora_ColumnName** (int cursor, int column)

Returns the name of the field/column *column* on the cursor *cursor*. The returned name is in all uppercase letters. Column o is the first column.

Ora_ColumnSize

(PHP 3, PHP 4)

Ora_ColumnSize -- get size of Oracle result column

Description

int **Ora_ColumnSize** (int cursor, int column)

Returns the size of the Oracle column *column* on the cursor *cursor*. Column o is the first column.

Ora_ColumnType

(PHP 3, PHP 4)

Ora_ColumnType -- get type of Oracle result column

Description

string **Ora_ColumnType** (int cursor, int column)

Returns the Oracle data type name of the field/column *column* on the cursor *cursor*. Column o is the first column. The returned type will be one of the following:

```
"VARCHAR2"
"VARCHAR"
"CHAR"
"NUMBER"
"LONG"
"LONG RAW"
"ROWID"
"DATE"
"CURSOR"
```

Ora_Commit

(PHP 3, PHP 4)

Ora_Commit -- commit an Oracle transaction

Description

int **ora_commit** (int conn)

Returns **TRUE** on success or **FALSE** on failure. Details about the error can be retrieved using the **ora_error()** and **ora_errorcode()** functions.

This function commits an Oracle transaction. A transaction is defined as all the changes on a given connection since the last commit/rollback, autocommit was turned off or when the connection was established.

Ora_CommitOff

(PHP 3, PHP 4)

Ora_CommitOff -- disable automatic commit

Description

int **ora_commitoff** (int conn)

Returns **TRUE** on success or **FALSE** on failure. Details about the error can be retrieved using the **ora_error()** and **ora_errorcode()** functions.

This function turns off automatic commit after each **ora_exec()**.

Ora_CommitOn

(PHP 3, PHP 4)

Ora_CommitOn -- enable automatic commit

Description

int **ora_commiton** (int conn)

This function turns on automatic commit after each **ora_exec()** on the given connection.

Returns **TRUE** on success or **FALSE** on failure. Details about the error can be retrieved using the **ora_error()** and **ora_errorcode()** functions.

Ora_Do

(PHP 3, PHP 4)

Ora_Do -- Parse, Exec, Fetch

Description

int **ora_do** (int conn, string query)

This function is quick combination of **ora_parse()**, **ora_exec()** and **ora_fetch()**. It will parse and execute a statement, then fetch the first result row.

Returns **TRUE** on success or **FALSE** on failure. Details about the error can be retrieved using the **ora_error()** and **ora_errorcode()** functions.

See also **ora_parse()**,**ora_exec()**, and **ora_fetch()**.

Ora_Error

(PHP 3, PHP 4)

Ora_Error -- get Oracle error message

Description

string **Ora_Error** (int cursor_or_connection)

Returns an error message of the form *XXX-NNNNN* where *XXX* is where the error comes from and *NNNNN* identifies the error message.

> **Note:** Support for connection ids was added in 3.0.4.

On UNIX versions of Oracle, you can find details about an error message like this: `$ oerr ora 00001 00001, 00000, "unique constraint (%s.%s) violated" // *Cause: An update or insert statement attempted to insert a duplicate key // For Trusted ORACLE configured in DBMS MAC mode, you may see // this message if a duplicate entry exists at a different level. // *Action: Either remove the unique restriction or do not insert the key`

Ora_ErrorCode

(PHP 3, PHP 4)

Ora_ErrorCode -- get Oracle error code

Description

int **Ora_ErrorCode** (int cursor_or_connection)

Returns the numeric error code of the last executed statement on the specified cursor or connection.

> **Note:** Support for connection ids was added in 3.0.4.

Ora_Exec

(PHP 3, PHP 4)

Ora_Exec -- execute parsed statement on an Oracle cursor

Description

int **ora_exec** (int cursor)

Returns **TRUE** on success or **FALSE** on failure. Details about the error can be retrieved using the ora_error() and ora_errorcode() functions.

See also ora_parse(), ora_fetch(), and ora_do().

Ora_Fetch_Into

(PHP 3, PHP 4)

Ora_Fetch_Into -- Fetch a row into the specified result array

Description

int **ora_fetch_into** (int cursor, array result [, int flags])

Fetches a row of data into an array. The *flags* has two flag values: if the **ORA_FETCHINTO_NULLS** flag is set, columns with **NULL** values are set in the array; and if the **ORA_FETCHINTO_ASSOC** flag is set, an associative array is created.

Returns the number of columns fetched.

Example 1. ora_fetch_into()

```
<?php
$results = array();
ora_fetch_into($cursor, $results);
```

```
echo $results[0];
echo $results[1];
$results = array();
ora_fetch_into($cursor, $results, ORA_FETCHINTO_NULLS|ORA_FETCHINTO_ASSOC);
echo $results['MyColumn'];
?>
```

See also ora_parse(),ora_exec(), ora_fetch(), and ora_do().

Ora_Fetch

(PHP 3, PHP 4)

Ora_Fetch -- fetch a row of data from a cursor

Description

int **ora_fetch** (int cursor)

Returns **TRUE** (a row was fetched) or **FALSE** (no more rows, or an error occured). If an error occured, details can be retrieved using the ora_error() and ora_errorcode() functions. If there was no error, ora_errorcode() will return 0.

Retrieves a row of data from the specified cursor.

See also ora_parse(),ora_exec(), and ora_do().

Ora_GetColumn

(PHP 3, PHP 4)

Ora_GetColumn -- get data from a fetched column

Description

mixed **ora_getcolumn** (int cursor, mixed column)

Returns the column data. If an error occurs, **FALSE** is returned and ora_errorcode() will return a non-zero value. Note, however, that a test for **FALSE** on the results from this function may be **TRUE** in cases where there is not error as well (**NULL** result, empty string, the number 0, the string "0").

Fetches the data for a column or function result.

Ora_Logoff

(PHP 3, PHP 4)

Ora_Logoff -- close an Oracle connection

Description

int **ora_logoff** (int connection)

Returns **TRUE** on success or **FALSE** on failure. Details about the error can be retrieved using the ora_error() and ora_errorcode() functions.

Logs out the user and disconnects from the server.

See also ora_logon().

Ora_Logon

(PHP 3, PHP 4)

Ora_Logon -- open an Oracle connection

Description

int **ora_logon** (string user, string password)

Establishes a connection between PHP and an Oracle database with the given username and password.

Connections can be made using SQL*Net by supplying the TNS name to *user* like this:

```
$conn = Ora_Logon("user@TNSNAME", "pass");
```

If you have character data with non-ASCII characters, you should make sure that NLS_LANG is set in your environment. For server modules, you should set it in the server's environment before starting the server.

Returns a connection index on success, or FALSE on failure. Details about the error can be retrieved using the ora_error() and ora_errorcode() functions.

Ora_Numcols

(PHP 3, PHP 4)

Ora_Numcols -- Returns the number of columns

Description

int **ora_numcols** (int cursor_ind)

ora_numcols() returns the number of columns in a result. Only returns meaningful values after an parse/exec/fetch sequence.

See also ora_parse(),ora_exec(), ora_fetch(), and ora_do().

Ora_Numrows

(PHP 3, PHP 4)

Ora_Numrows -- Returns the number of rows

Description

int **ora_numrows** (int cursor_ind)

ora_numrows() returns the number of rows in a result.

Ora_Open

(PHP 3, PHP 4)

Ora_Open -- open an Oracle cursor

Description

int **ora_open** (int connection)

Opens an Oracle cursor associated with connection.

Returns a cursor index or FALSE on failure. Details about the error can be retrieved using the ora_error() and ora_errorcode() functions.

Ora_Parse

(PHP 3, PHP 4)

Ora_Parse -- parse an SQL statement

Description

int **ora_parse** (int cursor_ind, string sql_statement, int defer)

This function parses an SQL statement or a PL/SQL block and associates it with the given cursor.

Returns **TRUE** on success or **FALSE** on failure.

See also **ora_exec()**, **ora_fetch()**, and **ora_do()**.

Ora_pLogon

(PHP 3, PHP 4)

Ora_pLogon -- Open a persistent Oracle connection

Description

int **ora_plogon** (string user, string password)

Establishes a persistent connection between PHP and an Oracle database with the given username and password.

See also **ora_logon()**.

Ora_Rollback

(PHP 3, PHP 4)

Ora_Rollback -- roll back transaction

Description

int **ora_rollback** (int connection)

This function undoes an Oracle transaction. (See **ora_commit()** for the definition of a transaction.)

Returns **TRUE** on success or **FALSE** on failure. Details about the error can be retrieved using the **ora_error()** and **ora_errorcode()** functions.

LXXIV. Ovrimos SQL functions

Introduction

Ovrimos SQL Server, is a client/server, transactional RDBMS combined with Web capabilities and fast transactions.

> **Note:** This extension is not available on Windows platforms.

Requirements

Ovrimos SQL Server is available at http://www.ovrimos.com/. You'll need to install the sqlcli library available in the Ovrimos SQL Server distribution.

Installation

To enable Ovrimos support in PHP just compile PHP with the `--with-ovrimos[=DIR]` parameter to your configure line. DIR is the Ovrimos' libsqlcli install directory.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

Examples

Example 1. Connect to Ovrimos SQL Server and select from a system table

```
<?php
$conn = ovrimos_connect ("server.domain.com", "8001", "admin", "password");
if ($conn != 0) {
    echo ("Connection ok!");
    $res = ovrimos_exec ($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        ovrimos_result_all ($res);
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

This will just connect to an Ovrimos SQL server.

Table of Contents

ovrimos_close

(PHP 4 >= 4.0.3)

ovrimos_close -- Closes the connection to ovrimos

Description

void **ovrimos_close** (int connection)

ovrimos_close() is used to close the specified connection.

ovrimos_close() closes a connection to Ovrimos. This has the effect of rolling back uncommitted transactions.

ovrimos_commit

(PHP 4 >= 4.0.3)

ovrimos_commit -- Commits the transaction

Description

int **ovrimos_commit** (int connection_id)

ovrimos_commit() is used to commit the transaction.

ovrimos_commit() commits the transaction.

ovrimos_connect

(PHP 4 >= 4.0.3)

ovrimos_connect -- Connect to the specified database

Description

int **ovrimos_connect** (string host, string db, string user, string password)

ovrimos_connect() is used to connect to the specified database.

ovrimos_connect() returns a connection id (greater than o) or o for failure. The meaning of 'host' and 'port' are those used everywhere in Ovrimos APIs. 'Host' is a host name or IP address and 'db' is either a database name, or a string containing the port number.

Example 1. ovrimos_connect() Example

```
<?php
$conn = ovrimos_connect ("server.domain.com", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res=ovrimos_exec ($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        ovrimos_result_all ($res);
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

The above example will connect to the database and print out the specified table.

ovrimos_cursor

(PHP 4 >= 4.0.3)

ovrimos_cursor -- Returns the name of the cursor

Description

int **ovrimos_cursor** (int result_id)

ovrimos_cursor() is used to get the name of the cursor.

ovrimos_cursor() returns the name of the cursor. Useful when wishing to perform positioned updates or deletes.

ovrimos_exec

(PHP 4 >= 4.0.3)

ovrimos_exec -- Executes an SQL statement

Description

int **ovrimos_exec** (int connection_id, string query)

ovrimos_exec() is used to execute an SQL statement.

ovrimos_exec() executes an SQL statement (query or update) and returns a result_id or **FALSE**. Evidently, the SQL statement should not contain parameters.

ovrimos_execute

(PHP 4 >= 4.0.3)

ovrimos_execute -- Executes a prepared SQL statement

Description

bool **ovrimos_execute** (int result_id [, array parameters_array])

ovrimos_execute() is used to execute an SQL statement.

ovrimos_execute() executes a prepared statement. Returns **TRUE** or **FALSE**. If the prepared statement contained parameters (question marks in the statement), the correct number of parameters should be passed in an array. Notice that I don't follow the PHP convention of placing just the name of the optional parameter inside square brackets. I couldn't bring myself on liking it.

ovrimos_fetch_into

(PHP 4 >= 4.0.3)

ovrimos_fetch_into -- Fetches a row from the result set

Description

bool **ovrimos_fetch_into** (int result_id, array result_array [, string how [, int rownumber]])

ovrimos_fetch_into() is used to fetch a row from the result set.

ovrimos_fetch_into() fetches a row from the result set into 'result_array', which should be passed by reference. Which row is fetched is determined by the two last parameters. 'how' is one of 'Next' (default), 'Prev', 'First', 'Last', 'Absolute', corresponding to forward direction from current position, backward direction from current position, forward direction from the start, backward direction from the end and absolute position from the start (essentially equivalent to 'first' but needs 'rownumber'). Case is not significant. 'Rownumber' is optional except for absolute positioning. Returns **TRUE** or **FALSE**.

Example 1. A fetch into example

```
<?php
$conn=ovrimos_connect ("neptune", "8001", "admin", "password");
if ($conn!=0) {
    echo "Connection ok!";
    $res=ovrimos_exec ($conn,"select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        if (ovrimos_fetch_into ($res, &$row)) {
            list ($table_id, $table_name) = $row;
            echo "table_id=".$table_id.", table_name=".$table_name."\n";
            if (ovrimos_fetch_into ($res, &$row)) {
```

```
                list ($table_id, $table_name) = $row;
                echo "table_id=".$table_id.", table_name=".$table_name."\n";
            } else {
                echo "Next: error\n";
            }
        } else {
            echo "First: error\n";
        }
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

This example will fetch a row.

ovrimos_fetch_row

(PHP 4 >= 4.0.3)

ovrimos_fetch_row -- Fetches a row from the result set

Description

bool **ovrimos_fetch_row** (int result_id [, int how [, int row_number]])

ovrimos_fetch_row() is used to fetch a row from the result set.

ovrimos_fetch_row() fetches a row from the result set. Column values should be retrieved with other calls. Returns **TRUE** or **FALSE**.

Example 1. A fetch row example

```
<?php
$conn = ovrimos_connect ("remote.host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res=ovrimos_exec ($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        if (ovrimos_fetch_row ($res, "First")) {
            $table_id = ovrimos_result ($res, 1);
            $table_name = ovrimos_result ($res, 2);
            echo "table_id=".$table_id.", table_name=".$table_name."\n";
            if (ovrimos_fetch_row ($res, "Next")) {
                $table_id = ovrimos_result ($res, "table_id");
                $table_name = ovrimos_result ($res, "table_name");
                echo "table_id=".$table_id.", table_name=".$table_name."\n";
            } else {
                echo "Next: error\n";
            }
        } else {
            echo "First: error\n";
        }
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

This will fetch a row and print the result.

ovrimos_field_len

(PHP 4 >= 4.0.3)

ovrimos_field_len -- Returns the length of the output column

Description

int **ovrimos_field_len** (int result_id, int field_number)

ovrimos_field_len() is used to get the length of the output column with number *field_number*, in result *result_id*.

ovrimos_field_len() returns the length of the output column at the (1-based) index specified.

ovrimos_field_name

(PHP 4 >= 4.0.3)

ovrimos_field_name -- Returns the output column name

Description

int **ovrimos_field_name** (int result_id, int field_number)

ovrimos_field_name() is used to get the output column name.

ovrimos_field_name() returns the output column name at the (1-based) index specified.

ovrimos_field_num

(PHP 4 >= 4.0.3)

ovrimos_field_num -- Returns the (1-based) index of the output column

Description

int **ovrimos_field_num** (int result_id, string field_name)

ovrimos_field_num() is used to get the (1-based) index of the output column.

ovrimos_field_num() returns the (1-based) index of the output column specified by name, or **FALSE**.

ovrimos_field_type

(PHP 4 >= 4.0.3)

ovrimos_field_type -- Returns the (numeric) type of the output column

Description

int **ovrimos_field_type** (int result_id, int field_number)

ovrimos_field_type() is used to get the (numeric) type of the output column.

ovrimos_field_type() returns the (numeric) type of the output column at the (1-based) index specified.

ovrimos_free_result

(PHP 4 >= 4.0.3)

ovrimos_free_result -- Frees the specified result_id

Description

bool **ovrimos_free_result** (int result_id)

ovrimos_free_result() is used to free the result_id.

ovrimos_free_result() frees the specified result_id *result_id*. Returns **TRUE**.

ovrimos_longreadlen

(PHP 4 >= 4.0.3)

ovrimos_longreadlen -- Specifies how many bytes are to be retrieved from long datatypes

Description

int **ovrimos_longreadlen** (int result_id, int length)

ovrimos_longreadlen() is used to specify how many bytes are to be retrieved from long datatypes.

ovrimos_longreadlen() specifies how many bytes are to be retrieved from long datatypes (long varchar and long varbinary). Default is zero. It currently sets this parameter the specified result set. Returns **TRUE**.

ovrimos_num_fields

(PHP 4 >= 4.0.3)

ovrimos_num_fields -- Returns the number of columns

Description

int **ovrimos_num_fields** (int result_id)

ovrimos_num_fields() is used to get the number of columns.

ovrimos_num_fields() returns the number of columns in a result_id resulting from a query.

ovrimos_num_rows

(PHP 4 >= 4.0.3)

ovrimos_num_rows -- Returns the number of rows affected by update operations

Description

int **ovrimos_num_rows** (int result_id)

ovrimos_num_rows() is used to get the number of rows affected by update operations.

ovrimos_num_rows() returns the number of rows affected by update operations.

ovrimos_prepare

(PHP 4 >= 4.0.3)

ovrimos_prepare -- Prepares an SQL statement

Description

int **ovrimos_prepare** (int connection_id, string query)

ovrimos_prepare() is used to prepare an SQL statement.

ovrimos_prepare() prepares an SQL statement and returns a result_id (or **FALSE** on failure).

Example 1. Connect to Ovrimos SQL Server and prepare a statement

```
<?php
$conn=ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn!=0) {
    echo "Connection ok!";
    $res=ovrimos_prepare ($conn, "select table_id, table_name
                                  from sys.tables where table_id=1");
    if ($res != 0) {
        echo "Prepare ok!";
```

```
        if (ovrimos_execute ($res)) {
            echo "Execute ok!\n";
            ovrimos_result_all ($res);
        } else {
            echo "Execute not ok!";
        }
        ovrimos_free_result ($res);
    } else {
        echo "Prepare not ok!\n";
    }
    ovrimos_close($conn);
}
?>
```

This will connect to Ovrimos SQL Server, prepare a statement and the execute it.

ovrimos_result_all

(PHP 4 >= 4.0.3)

ovrimos_result_all -- Prints the whole result set as an HTML table

Description

bool **ovrimos_result_all** (int result_id [, string format])

ovrimos_result_all() is used to print an HTML table containing the whole result set.

ovrimos_result_all() prints the whole result set as an HTML table. Returns **TRUE** or **FALSE**.

Example 1. Prepare a statement, execute, and view the result

```
<?php
$conn = ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res = ovrimos_prepare ($conn, "select table_id, table_name
                                    from sys.tables where table_id = 7");
    if ($res != 0) {
        echo "Prepare ok!";
        if (ovrimos_execute ($res, array(3))) {
            echo "Execute ok!\n";
            ovrimos_result_all ($res);
        } else {
            echo "Execute not ok!";
        }
        ovrimos_free_result ($res);
    } else {
        echo "Prepare not ok!\n";
    }
    ovrimos_close($conn);
}
?>
```

This will execute an SQL statement and print the result in an HTML table.

Example 2. Ovrimos_result_all with meta-information

```
<?php
$conn = ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res = ovrimos_exec ($conn, "select table_id, table_name
                                 from sys.tables where table_id = 1")
    if ($res != 0) {
        echo "Statement ok! cursor=".ovrimos_cursor ($res)."\n";
        $colnb = ovrimos_num_fields ($res);
        echo "Output columns=".$colnb."\n";
        for ($i=1; $i <= $colnb; $i++) {
            $name = ovrimos_field_name ($res, $i);
            $type = ovrimos_field_type ($res, $i);
            $len = ovrimos_field_len ($res, $i);
            echo "Column ".$i." name=".$name." type=".$type." len=".$len."\n";
        }
        ovrimos_result_all ($res);
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

Example 3. ovrimos_result_all example

```
<?php
$conn = ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res = ovrimos_exec ($conn, "update test set i=5");
    if ($res != 0) {
        echo "Statement ok!";
        echo ovrimos_num_rows ($res)." rows affected\n";
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

ovrimos_result

(PHP 4 >= 4.0.3)

ovrimos_result -- Retrieves the output column

Description

int **ovrimos_result** (int result_id, mixed field)

ovrimos_result() is used to retrieve the output column.

ovrimos_result() retrieves the output column specified by 'field', either as a string or as an 1-based index.

ovrimos_rollback

(PHP 4 >= 4.0.3)

ovrimos_rollback -- Rolls back the transaction

Description

int **ovrimos_rollback** (int connection_id)

ovrimos_rollback() is used to roll back the transaction.

ovrimos_rollback() rolls back the transaction.

LXXV. Output Control Functions

Introduction

The Output Control functions allow you to control when output is sent from the script. This can be useful in several different situations, especially if you need to send headers to the browser after your script has began outputting data. The Output Control functions do not affect headers sent using **header()** or **setcookie()**, only functions such as **echo()** and data between blocks of PHP code.

Requirements

No external libraries are needed to build this extension.

Installation

There is no installation needed to use these functions; they are part of the PHP core.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. Output Control configuration options

Name	Default	Changeable
output_buffering	"0"	PHP_INI_PERDIR|PHP_INI_SYSTEM
output_handler	NULL	PHP_INI_PERDIR|PHP_INI_SYSTEM
implicit_flush	"0"	PHP_INI_PERDIR|PHP_INI_SYSTEM

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Here is a short explanation of the configuration directives.

output_buffering **boolean**/**integer**

You can enable output buffering for all files by setting this directive to 'On'. If you wish to limit the size of the buffer to a certain size - you can use a maximum number of bytes instead of 'On', as a value for this directive (e.g., output_buffering=4096).

output_handler **string**

You can redirect all of the output of your scripts to a function. For example, if you set output_handler to **mb_output_handler()**, character encoding will be transparently converted to the specified encoding. Setting any output handler automatically turns on output buffering.

Note: You cannot use both **mb_output_handler()** with **ob_inconv_handler()** and you cannot use both **ob_gzhandler()** and zlib.output_compression.

implicit_flush **boolean**

FALSE by default. Changing this to **TRUE** tells PHP to tell the output layer to flush itself automatically after every output block. This is equivalent to calling the PHP function **flush()** after each and every call to **print()** or **echo()** and each and every HTML block.

When using PHP within an web environment, turning this option on has serious performance implications and is generally recommended for debugging purposes only. This value defaults to **TRUE** when operating under the CLI SAPI.

See also **ob_implicit_flush()**.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Examples

Example 1. Output Control example

```
<?php

ob_start();
echo "Hello\n";

setcookie ("cookiename", "cookiedata");
```

```
ob_end_flush();
?>
```

In the above example, the output from **echo()** would be stored in the output buffer until **ob_end_flush()** was called. In the mean time, the call to **setcookie()** successfully stored a cookie without causing an error. (You can not normally send headers to the browser after data has already been sent.)

> **Note:** When upgrading from PHP 4.1 (and 4.2) to 4.3 that due to a bug in earlier versions you must ensure that `implict_flush` is `OFF` in your `php.ini`, otherwise any output with **ob_start()** will be not be hidden from output.

See Also

See also **header()** and **setcookie()**.

Table of Contents

flush

(PHP 3, PHP 4)

flush -- Flush the output buffer

Description

void **flush** (void)

Flushes the output buffers of PHP and whatever backend PHP is using (CGI, a web server, etc). This effectively tries to push all the output so far to the user's browser.

> **Note: flush()** has no effect on the buffering scheme of your webserver or the browser on the client side.
>
> Several servers, especially on Win32, will still buffer the output from your script until it terminates before transmitting the results to the browser.
>
> Server modules for Apache like mod_gzip may do buffering of their own that will cause **flush()** to not result in data being sent immediately to the client.
>
> Even the browser may buffer its input before displaying it. Netscape, for example, buffers text until it receives an end-of-line or the beginning of a tag, and it won't render tables until the </table> tag of the outermost table is seen.
>
> Some versions of Microsoft Internet Explorer will only start to display the page after they have received 256 bytes of output, so you may need to send extra whitespace before flushing to get those browsers to display the page.

ob_clean

(PHP 4 >= 4.2.0)

ob_clean -- Clean (erase) the output buffer

Description

void **ob_clean** (void)

This function discards the contents of the output buffer.

This function does not destroy the output buffer like **ob_end_clean()** does.

See also **ob_flush()**, **ob_end_flush()** and **ob_end_clean()**.

ob_end_clean

(PHP 4)

ob_end_clean -- Clean (erase) the output buffer and turn off output buffering

Description

void **ob_end_clean** (void)

This function discards the contents of the output buffer and turns off output buffering.

See also **ob_start()**, **ob_clean()** and **ob_end_flush()**.

ob_end_flush

(PHP 4)

ob_end_flush -- Flush (send) the output buffer and turn off output buffering

Description

void **ob_end_flush** (void)

This function will send the contents of the output buffer (if any) and turn output buffering off. If you want to further process the buffer's contents you have to call **ob_get_contents()** before **ob_end_flush()** as the buffer contents are discarded after **ob_end_flush()** is called.

See also **ob_start()**, **ob_get_contents()**, **ob_flush()** and **ob_end_clean()**.

ob_flush

(PHP 4 >= 4.2.0)

ob_flush -- Flush (send) the output buffer

Description

void **ob_flush** (void)

This function will send the contents of the output buffer (if any). If you want to further process the buffer's contents you have to call **ob_get_contents()** before **ob_flush()** as the buffer contents are discarded after **ob_flush()** is called.

This function does not destroy the output buffer like **ob_end_flush()** does.

See also **ob_get_contents()**, **ob_clean()**, **ob_end_flush()** and **ob_end_clean()**.

ob_get_contents

(PHP 4)

ob_get_contents -- Return the contents of the output buffer

Description

string **ob_get_contents** (void)

This will return the contents of the output buffer or **FALSE**, if output buffering isn't active.

See also **ob_start()** and **ob_get_length()**.

ob_get_length

(PHP 4 >= 4.0.2)

ob_get_length -- Return the length of the output buffer

Description

string **ob_get_length** (void)

This will return the length of the contents in the output buffer or **FALSE**, if output buffering isn't active.

See also **ob_start()** and **ob_get_contents()**.

ob_get_level

(PHP 4 >= 4.2.0)

ob_get_level -- Return the nesting level of the output buffering mechanism

Description

int **ob_get_level** (void)

This will return the level of nested output buffering handlers.

See also **ob_start()** and **ob_get_contents()**.

ob_get_status

(PHP 4 >= 4.2.0)

ob_get_status -- Get status of output buffers

Description

array **ob_get_status** ([bool full_status])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

This will return the current status of output buffers. It returns array contains buffer status or **FALSE** for error.

See also **ob_get_level()**.

ob_gzhandler

(PHP 4 >= 4.0.4)

ob_gzhandler -- ob_start callback function to gzip output buffer

Description

string **ob_gzhandler** (string buffer [, int mode])

> **Note:** *mode* was added in PHP 4.0.5.

ob_gzhandler() is intended to be used as a callback function for **ob_start()** to help facilitate sending gz-encoded data to web browsers that support compressed web pages. Before **ob_gzhandler()** actually sends compressed data, it determines what type of content encoding the browser will accept ("gzip", "deflate" or none at all) and will return it's output accordingly. All browsers are supported since it's up to the browser to send the correct header saying that it accepts compressed web pages.

> **Note:** You cannot use both **ob_gzhandler()** and ini.zlib.output_compression. Also note that using ini.zlib.output_compression is preferred over **ob_gzhandler()**.

Example 1. ob_gzhandler() Example

```
<?php

ob_start("ob_gzhandler");

?>
<html>
<body>
<p>This should be a compressed page.
</html>
<body>
```

See also **ob_start()** and **ob_end_flush()**.

ob_implicit_flush

(PHP 4)

ob_implicit_flush -- Turn implicit flush on/off

Description

void **ob_implicit_flush** ([int flag])

ob_implicit_flush() will turn implicit flushing on or off (if no *flag* is given, it defaults to on). Implicit flushing will result in a flush operation after every output call, so that explicit calls to **flush()** will no longer be needed.

Turning implicit flushing on will disable output buffering, the output buffers current output will be sent as if **ob_end_flush()** had been called.

See also **flush()**, **ob_start()**, and **ob_end_flush()**.

ob_start

(PHP 4)

ob_start -- Turn on output buffering

Description

void **ob_start** ([string output_callback])

This function will turn output buffering on. While output buffering is active no output is sent from the script (other than headers), instead the output is stored in an internal buffer.

The contents of this internal buffer may be copied into a string variable using **ob_get_contents()**. To output what is stored in the internal buffer, use **ob_end_flush()**. Alternatively, **ob_end_clean()** will silently discard the buffer contents.

An optional *output_callback* function may be specified. This function takes a string as a parameter and should return a string.

The function will be called when ob_end_flush() is called, or when the output buffer is flushed to the browser at the end of the request. When *output_callback* is called, it will receive the contents of the output buffer as its parameter and is expected to return a new output buffer as a result, which will be sent to the browser.

> **Note:** In PHP 4.0.4, ob_gzhandler() was introduced to facilitate sending gz-encoded data to web browsers that support compressed web pages. ob_gzhandler() determines what type of content encoding the browser will accept and will return it's output accordingly.

Output buffers are stackable, that is, you may call **ob_start()** while another **ob_start()** is active. Just make sure that you call ob_end_flush() the appropriate number of times. If multiple output callback functions are active, output is being filtered sequentially through each of them in nesting order.

ob_end_clean(), ob_end_flush(), ob_clean(), ob_flush() and **ob_start()** may not be called from a callback function. If you call them from callback function, the behavior is undefined. If you would like to delete the contents of a buffer, return "" (a null string) from callback function.

Example 1. User defined callback function example

```
<?php

function callback($buffer) {

  // replace all the apples with oranges
  return (ereg_replace("apples", "oranges", $buffer));

}

ob_start("callback");

?>

<html>
<body>
<p>It's like comparing apples to oranges.
</body>
</html>

<?php

ob_end_flush();

?>
```

Would produce:

```
<html>
<body>
<p>It's like comparing oranges to oranges.
</body>
</html>
```

See also ob_get_contents(), ob_end_flush(), ob_end_clean(), ob_implicit_flush() and ob_gzhandler().

LXXVI. Object property and method call overloading

Introduction

The purpose of this extension is to allow overloading of object property access and method calls. Only one function is defined in this extension, overload() which takes the name of the class that should have this functionality enabled. The class named has to define appropriate methods if it wants to have this functionality: `__get()`, `__set()` and `__call()` respectively for getting/setting a property, or calling a method. This way overloading can be selective. Inside these handler functions the overloading is disabled so you can access object properties normally.

Warning
This extension is *EXPERIMENTAL*. The behaviour of this extension -- including the names of its functions and anything else documented about this extension -- may change without notice in a future release of PHP. Use this extension at your own risk.

Requirements

No external libraries are needed to build this extension.

Installation

In order to use these functions, you must compile PHP with the `--enable-overload` option. Starting with PHP 4.3.0 this extension is enabled by default. You can disable overload support with `--disable--overload`.

The windows version of `PHP` has built in support for this extension. You do not need to load any additional extension in order to use these functions.

> **Note:** Builtin support for overload is available with PHP 4.3.0.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Examples

Some simple examples on using the **overload()** function:

Example 1. Overloading a PHP class

```
<?php

class OO
{
    var $a = 111;
    var $elem = array('b' => 9, 'c' => 42);

    // Callback method for getting a property
    function __get($prop_name, &$prop_value)
    {
        if (isset($this->elem[$prop_name])) {
            $prop_value = $this->elem[$prop_name];
            return true;
        } else {
            return false;
        }
    }

    // Callback method for setting a property
    function __set($prop_name, $prop_value)
    {
        $this->elem[$prop_name] = $prop_value;
        return true;
    }
}

// Here we overload the OO object
overload('OO');

$o = new OO;
print "\$o->a: $o->a\n"; // print: $o->a:
print "\$o->b: $o->b\n"; // print: $o->b: 9
print "\$o->c: $o->c\n"; // print: $o->c: 42
print "\$o->d: $o->d\n"; // print: $o->d:

// add a new item to the $elem array in OO
$o->x = 56;

// instantiate stdclass (it is built-in in PHP 4)
```

```
// $val is not overloaded!
$val = new stdclass;
$val->prop = 555;

// Set "a" to be an array with the $val object in it
// But __set() will put this in the $elem array
$o->a = array($val);
var_dump($o->a[0]->prop);

?>
```

Warning
As this is an experimental extension, not all things work. There is no `__call()` support currently, you can only overload the get and set operations for properties. You cannot invoke the original overloading handlers of the class, and `__set()` only works to one level of property access.

Table of Contents

overload

(4.2.0 - 4.3.0 only)

overload -- Enable property and method call overloading for a class

Description

void **overload** ([string class_name])

The **overload()** function will enable property and method call overloading for a class identified by *class_name*. See an example in the introductory section of this part.

LXXVII. PDF functions

Introduction

The PDF functions in PHP can create PDF files using the PDFlib library created by Thomas Merz.

The documentation in this section is only meant to be an overview of the available functions in the PDFlib library and should not be considered an exhaustive reference. Please consult the documentation included in the source distribution of PDFlib for the full and detailed explanation of each function here. It provides a very good overview of what PDFlib is capable of doing and contains the most up-to-date documentation of all functions.

All of the functions in PDFlib and the PHP module have identical function names and parameters. You will need to understand some of the basic concepts of PDF and PostScript to efficiently use this extension. All lengths and coordinates are measured in PostScript points. There are generally 72 PostScript points to an inch, but this depends on the output resolution. Please see the PDFlib documentation included with the source distribution of PDFlib for a more thorough explanation of the coordinate system used.

Please note that most of the PDF functions require a *pdf object* as its first parameter. Please see the examples below for more information.

Requirements

PDFlib is available for download at http://www.pdflib.com/pdflib/index.html, but requires that you purchase a license for commercial use. The JPEG and TIFF libraries are required to compile this extension.

Issues with older versions of PDFlib

Any version of PHP 4 after March 9, 2000 does not support versions of PDFlib older than 3.0.

PDFlib 3.0 or greater is supported by PHP 3.0.19 and later.

Installation

To get these functions to work, you have to compile PHP with `--with-pdflib[=DIR]`. DIR is the PDFlib base install directory, defaults to `/usr/local`. In addition you can specify the jpeg, tiff, and pnglibrary for PDFlib to use, which is optional for PDFlib 4.x. To do so add to your configure line the options `--with-jpeg-dir[=DIR] --with-png-dir[=DIR] --with-tiff-dir[=DIR]`.

When using version 3.x of PDFlib, you should configure PDFlib with the option `--enable-shared-pdflib`.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Confusion with old PDFlib versions

Starting with PHP 4.0.5, the PHP extension for PDFlib is officially supported by PDFlib GmbH. This means that all the functions described in the PDFlib manual (V3.00 or greater) are supported by PHP 4 with exactly the same meaning and the same parameters. Only the return values may differ from the PDFlib manual, because the PHP convention of returning **FALSE** was adopted. For compatibility reasons, this binding for PDFlib still supports the old functions, but they should be replaced by their new versions. PDFlib GmbH will not support any problems arising from the use of these deprecated functions.

Table 1. Deprecated functions and their replacements

Old function	Replacement
pdf_put_image()	Not needed anymore.
pdf_execute_image()	Not needed anymore.
pdf_get_annotation()	**pdf_get_bookmark()** using the same parameters.
pdf_get_font()	pdf_get_value() passing `"font"` as the second parameter.
pdf_get_fontsize()	pdf_get_value() passing `"fontsize"` as the second parameter.
pdf_get_fontname()	pdf_get_parameter() passing `"fontname"` as the second parameter.
pdf_set_info_creator()	pdf_set_info() passing `"Creator"` as the second parameter.
pdf_set_info_title()	pdf_set_info() passing `"Title"` as the second parameter.
pdf_set_info_subject()	pdf_set_info() passing `"Subject"` as the second parameter.
pdf_set_info_author()	pdf_set_info() passing `"Author"` as the second parameter.
pdf_set_info_keywords()	pdf_set_info() passing `"Keywords"` as the second parameter.
pdf_set_leading()	pdf_set_value() passing `"leading"` as the second parameter.
pdf_set_text_rendering()	pdf_set_value() passing `"textrendering"` as the second parameter.
pdf_set_text_rise()	pdf_set_value() passing `"textrise"` as the second parameter.
pdf_set_horiz_scaling()	pdf_set_value() passing `"horizscaling"` as the second parameter.
pdf_set_text_matrix()	Not available anymore
pdf_set_char_spacing()	pdf_set_value() passing `"charspacing"` as the second parameter.
pdf_set_word_spacing()	pdf_set_value() passing `"wordspacing"` as the second parameter.
pdf_set_transition()	pdf_set_parameter() passing `"transition"` as the second parameter.
pdf_open()	pdf_new() plus an subsequent call of pdf_open_file()
pdf_set_font()	pdf_findfont() plus an subsequent call of pdf_setfont()
pdf_set_duration()	pdf_set_value() passing `"duration"` as the second parameter.
pdf_open_gif()	pdf_open_image_file() passing `"gif"` as the second parameter.
pdf_open_jpeg()	pdf_open_image_file() passing `"jpeg"` as the second parameter.
pdf_open_tiff()	pdf_open_image_file() passing `"tiff"` as the second parameter.
pdf_open_png()	pdf_open_image_file() passing `"png"` as the second parameter.

Old function	Replacement
pdf_get_image_width()	pdf_get_value() passing "imagewidth" as the second parameter and the image as the third parameter.
pdf_get_image_height()	pdf_get_value() passing "imageheight" as the second parameter and the image as the third parameter.

Examples

Most of the functions are fairly easy to use. The most difficult part is probably creating your first PDF document. The following example should help to get you started. It creates test.pdf with one page. The page contains the text "Times Roman outlined" in an outlined, 30pt font. The text is also underlined.

Example 1. Creating a PDF document with PDFlib

```
<?php
$pdf = pdf_new();
pdf_open_file($pdf, "test.pdf");
pdf_set_info($pdf, "Author", "Uwe Steinmann");
pdf_set_info($pdf, "Title", "Test for PHP wrapper of PDFlib 2.0");
pdf_set_info($pdf, "Creator", "See Author");
pdf_set_info($pdf, "Subject", "Testing");
pdf_begin_page($pdf, 595, 842);
pdf_add_outline($pdf, "Page 1");
$font = pdf_findfont($pdf, "Times New Roman", "winansi", 1);
pdf_setfont($pdf, $font, 10);
pdf_set_value($pdf, "textrendering", 1);
pdf_show_xy($pdf, "Times Roman outlined", 50, 750);
pdf_moveto($pdf, 50, 740);
pdf_lineto($pdf, 330, 740);
pdf_stroke($pdf);
pdf_end_page($pdf);
pdf_close($pdf);
pdf_delete($pdf);
echo "<A HREF=getpdf.php>finished</A>";
?>
```

The script getpdf.php just returns the pdf document.

```
<?php
$len = filesize($filename);
header("Content-type: application/pdf");
header("Content-Length: $len");
header("Content-Disposition: inline; filename=foo.pdf");
readfile($filename);
?>
```

The PDFlib distribution contains a more complex example which creates a page with an analog clock. Here we use the in-memory creation feature of PDFlib to alleviate the need to use temporary files. The example was converted to PHP from the PDFlib example. (The same example is available in the CLibPDF documentation.)

Example 2. pdfclock example from PDFlib distribution

```
<?php
$radius = 200;
$margin = 20;
$pagecount = 10;

$pdf = pdf_new();

if (!pdf_open_file($pdf, "")) {
    print error;
    exit;
};

pdf_set_parameter($pdf, "warning", "true");

pdf_set_info($pdf, "Creator", "pdf_clock.php");
pdf_set_info($pdf, "Author", "Uwe Steinmann");
pdf_set_info($pdf, "Title", "Analog Clock");

while($pagecount-- > 0) {
    pdf_begin_page($pdf, 2 * ($radius + $margin), 2 * ($radius + $margin));

    pdf_set_parameter($pdf, "transition", "wipe");
    pdf_set_value($pdf, "duration", 0.5);

    pdf_translate($pdf, $radius + $margin, $radius + $margin);
    pdf_save($pdf);
```

```
    pdf_setrgbcolor($pdf, 0.0, 0.0, 1.0);

    /* minute strokes */
    pdf_setlinewidth($pdf, 2.0);
    for ($alpha = 0; $alpha < 360; $alpha += 6) {
        pdf_rotate($pdf, 6.0);
        pdf_moveto($pdf, $radius, 0.0);
        pdf_lineto($pdf, $radius-$margin/3, 0.0);
        pdf_stroke($pdf);
    }

    pdf_restore($pdf);
    pdf_save($pdf);

    /* 5 minute strokes */
    pdf_setlinewidth($pdf, 3.0);
    for ($alpha = 0; $alpha < 360; $alpha += 30) {
        pdf_rotate($pdf, 30.0);
        pdf_moveto($pdf, $radius, 0.0);
        pdf_lineto($pdf, $radius-$margin, 0.0);
        pdf_stroke($pdf);
    }

    $ltime = getdate();

    /* draw hour hand */
    pdf_save($pdf);
    pdf_rotate($pdf,-(($ltime['minutes']/60.0)+$ltime['hours']-3.0)*30.0);
    pdf_moveto($pdf, -$radius/10, -$radius/20);
    pdf_lineto($pdf, $radius/2, 0.0);
    pdf_lineto($pdf, -$radius/10, $radius/20);
    pdf_closepath($pdf);
    pdf_fill($pdf);
    pdf_restore($pdf);

    /* draw minute hand */
    pdf_save($pdf);
    pdf_rotate($pdf,-(($ltime['seconds']/60.0)+$ltime['minutes']-15.0)*6.0);
    pdf_moveto($pdf, -$radius/10, -$radius/20);
    pdf_lineto($pdf, $radius * 0.8, 0.0);
    pdf_lineto($pdf, -$radius/10, $radius/20);
    pdf_closepath($pdf);
    pdf_fill($pdf);
    pdf_restore($pdf);

    /* draw second hand */
    pdf_setrgbcolor($pdf, 1.0, 0.0, 0.0);
    pdf_setlinewidth($pdf, 2);
    pdf_save($pdf);
    pdf_rotate($pdf, -(($ltime['seconds'] - 15.0) * 6.0));
    pdf_moveto($pdf, -$radius/5, 0.0);
    pdf_lineto($pdf, $radius, 0.0);
    pdf_stroke($pdf);
    pdf_restore($pdf);

    /* draw little circle at center */
    pdf_circle($pdf, 0, 0, $radius/30);
    pdf_fill($pdf);

    pdf_restore($pdf);

    pdf_end_page($pdf);

    # to see some difference
    sleep(1);
}

pdf_close($pdf);

$buf = pdf_get_buffer($pdf);
$len = strlen($buf);

header("Content-type: application/pdf");
header("Content-Length: $len");
header("Content-Disposition: inline; filename=foo.pdf");
print $buf;

pdf_delete($pdf);
?>
```

See Also

Note: An alternative PHP module for PDF document creation based on FastIO's ClibPDF is available. Please see the ClibPDF section for details. Note that ClibPDF has a slightly different API than PDFlib.

Table of Contents

pdf_add_launchlink -- Add a launch annotation for current page
pdf_add_locallink -- Add a link annotation for current page
pdf_add_note -- Add a note annotation for current page
pdf_add_outline -- Deprecated: Adds bookmark for current page
pdf_add_pdflink -- Adds file link annotation for current page
pdf_add_thumbnail -- Adds thumbnail for current page
pdf_add_weblink -- Adds weblink for current page
pdf_arc -- Draws an arc (counterclockwise)
pdf_arcn -- Draws an arc (clockwise)
pdf_attach_file -- Adds a file attachement for current page
pdf_begin_page -- Starts new page
pdf_begin_pattern -- Starts new pattern
pdf_begin_template -- Starts new template
pdf_circle -- Draws a circle
pdf_clip -- Clips to current path
pdf_close_image -- Closes an image
pdf_close_pdi_page -- Close the page handle
pdf_close_pdi -- Close the input PDF document
pdf_close -- Closes a pdf object
pdf_closepath_fill_stroke -- Closes, fills and strokes current path
pdf_closepath_stroke -- Closes path and draws line along path
pdf_closepath -- Closes path
pdf_concat -- Concatenate a matrix to the CTM
pdf_continue_text -- Outputs text in next line
pdf_curveto -- Draws a curve
pdf_delete -- Deletes a PDF object
pdf_end_page -- Ends a page
pdf_end_pattern -- Finish pattern
pdf_end_template -- Finish template
pdf_endpath -- Deprecated: Ends current path
pdf_fill_stroke -- Fills and strokes current path
pdf_fill -- Fills current path
pdf_findfont -- Prepare font for later use with **pdf_setfont()**.
pdf_get_buffer -- Fetch the buffer containig the generated PDF data.
pdf_get_font -- Deprecated: font handling
pdf_get_fontname -- Deprecated: font handling
pdf_get_fontsize -- Deprecated: font handling
pdf_get_image_height -- Returns height of an image
pdf_get_image_width -- Returns width of an image
pdf_get_majorversion -- Returns the major version number of the PDFlib
pdf_get_minorversion -- Returns the minor version number of the PDFlib
pdf_get_parameter -- Gets certain parameters
pdf_get_pdi_parameter -- Get some PDI string parameters
pdf_get_pdi_value -- Gets some PDI numerical parameters
pdf_get_value -- Gets certain numerical value
pdf_initgraphics -- Resets graphic state
pdf_lineto -- Draws a line
pdf_makespotcolor -- Makes a spotcolor
pdf_moveto -- Sets current point
pdf_new -- Creates a new pdf object
pdf_open_CCITT -- Opens a new image file with raw CCITT data
pdf_open_file -- Opens a new pdf object
pdf_open_gif -- Deprecated: Opens a GIF image
pdf_open_image_file -- Reads an image from a file
pdf_open_image -- Versatile function for images
pdf_open_jpeg -- Deprecated: Opens a JPEG image
pdf_open_memory_image -- Opens an image created with PHP's image functions
pdf_open_pdi_page -- Prepare a page
pdf_open_pdi -- Opens a PDF file
pdf_open_png -- Deprecated: Opens a PNG image
pdf_open_tiff -- Deprecated: Opens a TIFF image
pdf_open -- Deprecated: Open a new pdf object
pdf_place_image -- Places an image on the page
pdf_place_pdi_page -- Places an image on the page
pdf_rect -- Draws a rectangle
pdf_restore -- Restores formerly saved environment
pdf_rotate -- Sets rotation
pdf_save -- Saves the current environment
pdf_scale -- Sets scaling
pdf_set_border_color -- Sets color of border around links and annotations

pdf_set_border_dash -- Sets dash style of border around links and annotations
pdf_set_border_style -- Sets style of border around links and annotations
pdf_set_char_spacing -- Deprecated: Sets character spacing
pdf_set_duration -- Deprecated: Sets duration between pages
pdf_set_font -- Deprecated: Selects a font face and size
pdf_set_horiz_scaling -- Sets horizontal scaling of text
pdf_set_info_author -- Fills the author field of the document
pdf_set_info_creator -- Fills the creator field of the document
pdf_set_info_keywords -- Fills the keywords field of the document
pdf_set_info_subject -- Fills the subject field of the document
pdf_set_info_title -- Fills the title field of the document
pdf_set_info -- Fills a field of the document information
pdf_set_leading -- Deprecated: Sets distance between text lines
pdf_set_parameter -- Sets certain parameters
pdf_set_text_matrix -- Deprecated: Sets the text matrix
pdf_set_text_pos -- Sets text position
pdf_set_text_rendering -- Deprecated: Determines how text is rendered
pdf_set_text_rise -- Deprecated: Sets the text rise
pdf_set_value -- Sets certain numerical value
pdf_set_word_spacing -- Depriciated: Sets spacing between words
pdf_setcolor -- Sets fill and stroke color
pdf_setdash -- Sets dash pattern
pdf_setflat -- Sets flatness
pdf_setfont -- Set the current font
pdf_setgray_fill -- Sets filling color to gray value
pdf_setgray_stroke -- Sets drawing color to gray value
pdf_setgray -- Sets drawing and filling color to gray value
pdf_setlinecap -- Sets linecap parameter
pdf_setlinejoin -- Sets linejoin parameter
pdf_setlinewidth -- Sets line width
pdf_setmatrix -- Sets current transformation matrix
pdf_setmiterlimit -- Sets miter limit
pdf_setpolydash -- Sets complicated dash pattern
pdf_setrgbcolor_fill -- Sets filling color to rgb color value
pdf_setrgbcolor_stroke -- Sets drawing color to rgb color value
pdf_setrgbcolor -- Sets drawing and filling color to rgb color value
pdf_show_boxed -- Output text in a box
pdf_show_xy -- Output text at given position
pdf_show -- Output text at current position
pdf_skew -- Skews the coordinate system
pdf_stringwidth -- Returns width of text using current font
pdf_stroke -- Draws line along path
pdf_translate -- Sets origin of coordinate system

pdf_add_annotation

pdf_add_annotation -- Deprecated: Adds annotation

Description

pdf_add_outline() is replaced by **pdf_add_note()**

See also **pdf_add_note()**.

pdf_add_bookmark

(PHP 4 >= 4.0.1)

pdf_add_bookmark -- Adds bookmark for current page

Description

int **pdf_add_bookmark** (int pdf object, string text [, int parent [, int open]])

Add a nested bookmark under *parent*, or a new top-level bookmark if *parent* = 0. Returns a bookmark descriptor which may be used as parent for subsequent nested bookmarks. If open = 1, child bookmarks will be folded out, and invisible if open = 0.

pdf_add_launchlink

(PHP 4 >= 4.0.5)

pdf_add_launchlink -- Add a launch annotation for current page

Description

int **pdf_add_launchlink** (int pdf object, float llx, float lly, float urx, float ury, string filename)

Add a launch annotation (to a target of arbitrary file type).

pdf_add_locallink

(PHP 4 >= 4.0.5)

pdf_add_locallink -- Add a link annotation for current page

Description

int **pdf_add_locallink** (int pdf object, float llx, float lly, float urx, float ury, int page, string dest)

Add a link annotation to a target within the current PDF file.

pdf_add_note

(PHP 4 >= 4.0.5)

pdf_add_note -- Add a note annotation for current page

Description

int **pdf_add_note** (int pdf object, float llx, float lly, float urx, float ury, string contents, string title, string icon, int open)

Add a note annotation. icon is one of of "comment, "insert", "note", "paragraph", "newparagraph", "key", or "help".

pdf_add_outline

pdf_add_outline -- Deprecated: Adds bookmark for current page

Description

Deprecated.

See **pdf_add_bookmark()**.

pdf_add_pdflink

(PHP 3>= 3.0.12, PHP 4)

pdf_add_pdflink -- Adds file link annotation for current page

Description

int **pdf_add_pdflink** (int pdf object, float llx, float lly, float urx, float ury, string filename, int page, string dest)

Add a file link annotation (to a PDF target).

pdf_add_thumbnail

(PHP 4 >= 4.0.5)

pdf_add_thumbnail -- Adds thumbnail for current page

Description

int **pdf_add_thumbnail** (int pdf object, int image)

Add an existing image as thumbnail for the current page.

pdf_add_weblink

(PHP 3>= 3.0.12, PHP 4)

pdf_add_weblink -- Adds weblink for current page

Description

int **pdf_add_weblink** (int pdf object, float llx, float lly, float urx, float ury, string url)

Add a weblink annotation to a target URL on the Web.

pdf_arc

(PHP 3>= 3.0.6, PHP 4)

pdf_arc -- Draws an arc (counterclockwise)

Description

void **pdf_arc** (resource pdf object, float x, float y, float r, float alpha, float beta)

Draw a counterclockwise circular arc from alpha to beta degrees

See also: **pdf_arcn()**

pdf_arcn

(PHP 4 >= 4.0.5)

pdf_arcn -- Draws an arc (clockwise)

Description

void **pdf_arc** (resource pdf object, float x, float y, float r, float alpha, float beta)

Draw a clockwise circular arc from alpha to beta degrees

See also: **pdf_arc()**

pdf_attach_file

(PHP 4 >= 4.0.5)

pdf_attach_file -- Adds a file attachement for current page

Description

int **pdf_attach_file** (int pdf object, float llx, float lly, float urx, float ury, string filename, string description, string author, string mimetype, string icon)

Add a file attachment annotation. icon is one of "graph, "paperclip", "pushpin", or "tag".

pdf_begin_page

(PHP 3>= 3.0.6, PHP 4)

pdf_begin_page -- Starts new page

Description

void **pdf_begin_page** (int pdf object, float width, float height)

Add a new page to the document. The *width* and *height* are specified in points, which are 1/72 of an inch.

Table 1. Common Page Sizes in Points

name	size
A0	2380?3368
A1	1684?2380
A2	1190?1684
A3	842?1190
A4	595?842
A5	421?595
A6	297?421
B5	501?709
letter (8.5"?11")	612?792
legal (8.5"?14")	612?1008
ledger (17"?11")	1224?792
11"?17"	792?1224

pdf_begin_pattern

(PHP 4 >= 4.0.5)

pdf_begin_pattern -- Starts new pattern

Description

int **pdf_begin_pattern** (int pdf object, float width, float height, float xstep, float ystep, int painttype)

Starts a new pattern definition and returns a pattern handle. *width*, and *height* define the bounding box for the pattern. *xstep* and *ystep* give the repeated pattern offsets. *painttype*=1 means that the pattern has its own colour settings whereas a value of 2 indicates that the current colour is used when the pattern is applied.

pdf_begin_template

(PHP 4 >= 4.0.5)

pdf_begin_template -- Starts new template

Description

int **pdf_begin_template** (int pdf object, float width, float height)

Start a new template definition.

pdf_circle

(PHP 3>= 3.0.6, PHP 4)

pdf_circle -- Draws a circle

Description

void **pdf_circle** (int pdf object, float x, float y, float r)

Draw a circle with center (x, y) and radius r.

pdf_clip

(PHP 3>= 3.0.6, PHP 4)

pdf_clip -- Clips to current path

Description

void **pdf_clip** (int pdf object)

Use the current path as clipping path.

pdf_close_image

(PHP 3>= 3.0.7, PHP 4)

pdf_close_image -- Closes an image

Description

void **pdf_close_image** (int pdf object, int image)

Close an *image* retrieved with one of the **pdf_open_image*()** functions.

pdf_close_pdi_page

(PHP 4 >= 4.0.5)

pdf_close_pdi_page -- Close the page handle

Description

void **pdf_close_pdi_page** (int pdf object, int pagehandle)

Close the page handle, and free all page-related resources.

pdf_close_pdi

(PHP 4 >= 4.0.5)

pdf_close_pdi -- Close the input PDF document

Description

void **pdf_close_pdi** (int pdf object, int dochandle)

Close all open page handles, and close the input PDF document.

pdf_close

(PHP 3>= 3.0.6, PHP 4)

pdf_close -- Closes a pdf object

Description

void **pdf_close** (int pdf object)

Close the generated PDF file, and free all document-related resources.

pdf_closepath_fill_stroke

(PHP 3>= 3.0.6, PHP 4)

pdf_closepath_fill_stroke -- Closes, fills and strokes current path

Description

void **pdf_closepath_fill_stroke** (int pdf object)

Close the path, fill, and stroke it.

pdf_closepath_stroke

(PHP 3>= 3.0.6, PHP 4)

pdf_closepath_stroke -- Closes path and draws line along path

Description

void **pdf_closepath_stroke** (int pdf object)

Close the path, and stroke it.

pdf_closepath

(PHP 3>= 3.0.6, PHP 4)

pdf_closepath -- Closes path

Description

void **pdf_closepath** (int pdf object)

Close the current path.

pdf_concat

(PHP 4 >= 4.0.5)

pdf_concat -- Concatenate a matrix to the CTM

Description

void **pdf_concat** (int pdf object, float a, float b, float c, float d, float e, float f)

Concatenate a matrix to the CTM.

pdf_continue_text

(PHP 3>= 3.0.6, PHP 4)

pdf_continue_text -- Outputs text in next line

Description

void **pdf_continue_text** (int pdf object, string text)

Print text at the next line. The spacing between lines is determined by the *leading* parameter.

pdf_curveto

(PHP 3>= 3.0.6, PHP 4)

pdf_curveto -- Draws a curve

Description

void **pdf_curveto** (int pdf object, float x1, float y1, float x2, float y2, float x3, float y3)

Draw a Bezier curve from the current point, using 3 more control points.

pdf_delete

(PHP 4 >= 4.0.5)

pdf_delete -- Deletes a PDF object

Description

void **pdf_delete** (int pdf object)

Delete the PDF object, and free all internal resources.

pdf_end_page

(PHP 3>= 3.0.6, PHP 4)

pdf_end_page -- Ends a page

Description

void **pdf_end_page** (int pdf object)

Finish the page.

pdf_end_pattern

(PHP 4 >= 4.0.5)

pdf_end_pattern -- Finish pattern

Description

void **pdf_end_pattern** (int pdf object)

Finish the pattern definition.

pdf_end_template

(PHP 4 >= 4.0.5)

pdf_end_template -- Finish template

Description

void **pdf_end_template** (int pdf object)

Finish the template definition.

pdf_endpath

pdf_endpath -- Deprecated: Ends current path

Description

Deprecated, use one of the stroke, fill, or clip functions instead.

pdf_fill_stroke

(PHP 3>= 3.0.6, PHP 4)

pdf_fill_stroke -- Fills and strokes current path

Description

void **pdf_fill_stroke** (int pdf object)

Fill and stroke the path with the current fill and stroke color.

pdf_fill

(PHP 3>= 3.0.6, PHP 4)

pdf_fill -- Fills current path

Description

void **pdf_fill_stroke** (int pdf object)

Fill the interior of the path with the current fill color.

pdf_findfont

(PHP 4 >= 4.0.5)

pdf_findfont -- Prepare font for later use with **pdf_setfont()**.

Description

int **pdf_findfont** (int pdf object, string fontname, string encoding, int embed)

Prepare a font for later use with **pdf_setfont()**. The metrics will be loaded, and if embed is nonzero, the font file will be checked, but not yet used. *encoding* is one of "builtin", "macroman", "winansi", "host", or a user-defined encoding name, or the name of a CMap.

pdf_findfont() returns a font handle or **FALSE** on error.

Example 1. pdf_findfont() example

```
<?php

$font = pdf_findfont($pdf, "Times New Roman", "winansi", 1);
if ($font) {
    pdf_setfont($pdf, $font, 10);
}

?>
```

pdf_get_buffer

(PHP 4 >= 4.0.5)

pdf_get_buffer -- Fetch the buffer containig the generated PDF data.

Description

string **pdf_get_buffer** (int pdf object)

Get the contents of the PDF output buffer. The result must be used by the client before calling any other PDFlib function.

pdf_get_font

pdf_get_font -- Deprecated: font handling

Description

Deprecated.

See **pdf_get_value()**.

pdf_get_fontname

pdf_get_fontname -- Deprecated: font handling

Description

Deprecated.

See **pdf_get_parameter()**.

pdf_get_fontsize

pdf_get_fontsize -- Deprecated: font handling

Description

Deprecated.

See **pdf_get_value()**.

pdf_get_image_height

(PHP 3>= 3.0.12, PHP 4)

pdf_get_image_height -- Returns height of an image

Description

string **pdf_get_image_height** (int pdf object, int image)

pdf_get_image_height() is deprecated, use **pdf_get_value()** instead.

pdf_get_image_width

(PHP 3>= 3.0.12, PHP 4)

pdf_get_image_width -- Returns width of an image

Description

string **pdf_get_image_width** (int pdf object, int image)

The **pdf_get_image_width()** is deprecated, use **pdf_get_value()** instead.

pdf_get_majorversion

(PHP 4 >= 4.2.0)

pdf_get_majorversion -- Returns the major version number of the PDFlib

Description

int **pdf_get_majorversion** (void)

Returns the major version number of the PDFlib.

pdf_get_minorversion

(PHP 4 >= 4.2.0)

pdf_get_minorversion -- Returns the minor version number of the PDFlib

Description

int **pdf_get_majorversion** (void)

Returns the minor version number of the PDFlib.

pdf_get_parameter

(PHP 4 >= 4.0.1)

pdf_get_parameter -- Gets certain parameters

Description

string **pdf_get_parameter** (int pdf object, string key [, float modifier])

Get the contents of some PDFlib parameter with string type.

pdf_get_pdi_parameter

(PHP 4 >= 4.0.5)

pdf_get_pdi_parameter -- Get some PDI string parameters

Description

string **pdf_get_pdi_parameter** (int pdf object, string key, int doc, int page, int index)

Get the contents of some PDI document parameter with string type.

pdf_get_pdi_value

(PHP 4 >= 4.0.5)

pdf_get_pdi_value -- Gets some PDI numerical parameters

Description

string **pdf_get_pdi_value** (int pdf object, string key, int doc, int page, int index)

Get the contents of some PDI document parameter with numerical type.

pdf_get_value

(PHP 4 >= 4.0.1)

pdf_get_value -- Gets certain numerical value

Description

float **pdf_get_value** (int pdf object, string key [, float modifier])

Get the contents of some PDFlib parameter with float type.

pdf_initgraphics

(PHP 4 >= 4.0.5)

pdf_initgraphics -- Resets graphic state

Description

void **pdf_initgraphics** (int pdf object)

Reset all implicit color and graphics state parameters to their defaults.

pdf_lineto

(PHP 3>= 3.0.6, PHP 4)

pdf_lineto -- Draws a line

Description

void **pdf_lineto** (int pdf object, float x, float y)

Draw a line from the current point to (*x*, *y*).

pdf_makespotcolor

(PHP 4 >= 4.0.5)

pdf_makespotcolor -- Makes a spotcolor

Description

void **pdf_makespotcolor** (int pdf object, string spotname)

Make a named spot color from the current color.

pdf_moveto

(PHP 3>= 3.0.6, PHP 4)

pdf_moveto -- Sets current point

Description

void **pdf_moveto** (int pdf object, float x, float y)

Set the current point.

> **Note:** The current point for graphics and the current text output position are maintained separately. See **pdf_set_text_pos()** to set the text output position.

pdf_new

(PHP 4 >= 4.0.5)

pdf_new -- Creates a new pdf object

Description

int **pdf_new** ( )

Create a new PDF object, using default error handling and memory management.

pdf_open_CCITT

(PHP 4 >= 4.0.5)

pdf_open_CCITT -- Opens a new image file with raw CCITT data

Description

int **pdf_open_CCITT** (int pdf object, string filename, int width, int height, int BitReverse, int k, int BlackIs1)

Open a raw CCITT image.

pdf_open_file

(PHP 4 >= 4.0.5)

pdf_open_file -- Opens a new pdf object

Description

int **pdf_open_file** (int pdf object [, string filename])

Create a new PDF file using the supplied file name. If *filename* is empty the PDF document will be generated in memory instead of on file. The result must be fetched by the client with the **pdf_get_buffer()** function.

The following example shows how to create a pdf document in memory and how to output it correctly.

Example 1. Creating a PDF document in memory

```
<?php

$pdf = pdf_new();

pdf_open_file($pdf);
pdf_begin_page($pdf, 595, 842);
pdf_set_font($pdf, "Times-Roman", 30, "host");
pdf_set_value($pdf, "textrendering", 1);
pdf_show_xy($pdf, "A PDF document created in memory!", 50, 750);
pdf_end_page($pdf);
pdf_close($pdf);

$data = pdf_get_buffer($pdf);

header("Content-type: application/pdf");
header("Content-disposition: inline; filename=test.pdf");
header("Content-length: " . strlen($data));

echo $data;

?>
```

pdf_open_gif

pdf_open_gif -- Deprecated: Opens a GIF image

Description

Deprecated.

See **pdf_open_image()**,

pdf_open_image_file

(PHP 3 CVS only, PHP 4)

pdf_open_image_file -- Reads an image from a file

Description

int **pdf_open_image_file** (int PDF-document, string imagetype, string filename [, string stringparam [, string intparam]])

Open an image file. Supported types are "jpeg", "tiff", "gif", and "png". *stringparam* is either "", "mask", "masked", or "page". *intparam*is either 0, the image id of the applied mask, or the page.

pdf_open_image

(PHP 4 >= 4.0.5)

pdf_open_image -- Versatile function for images

Description

int **pdf_open_image** (int PDF-document, string imagetype, string source, string data, long length, int width, int height, int components, int bpc, string params)

Use image data from a variety of data sources. Supported types are "jpeg", "ccitt", "raw". Supported sources are "memory", "fileref", "url". len is only used for type="raw", params is only used for type="ccitt".

pdf_open_jpeg

pdf_open_jpeg -- Deprecated: Opens a JPEG image

Description

Deprecated.

See also **pdf_open_image()**,

pdf_open_memory_image

(PHP 3>= 3.0.10, PHP 4)

pdf_open_memory_image -- Opens an image created with PHP's image functions

Description

int **pdf_open_memory_image** (int pdf object, int image)

The **pdf_open_memory_image()** function takes an image created with the PHP's image functions and makes it available for the pdf object. The function returns a pdf image identifier.

Example 1. Including a memory image

```
<?php
$im = ImageCreate(100, 100);
$col = ImageColorAllocate($im, 80, 45, 190);
ImageFill($im, 10, 10, $col);
$pim = pdf_open_memory_image($pdf, $im);
ImageDestroy($im);
pdf_place_image($pdf, $pim, 100, 100, 1);
pdf_close_image($pdf, $pim);
?>
```

See also **pdf_close_image()**, **pdf_place_image()**.

pdf_open_pdi_page

(PHP 4 >= 4.0.5)

pdf_open_pdi_page -- Prepare a page

Description

int **pdf_open_pdi_page** (int pdf object, int dochandle, int pagenumber, string pagelabel)

Prepare a page for later use with **pdf_place_image()**

pdf_open_pdi

(PHP 4 >= 4.0.5)

pdf_open_pdi -- Opens a PDF file

Description

int **pdf_open_pdi** (int pdf object, string filename, string stringparam, int intparam)

Open an existing PDF document for later use.

pdf_open_png

pdf_open_png -- Deprecated: Opens a PNG image

Description

Deprecated.

See **pdf_open_image()**.

pdf_open_tiff

(PHP 4)

pdf_open_tiff -- Deprecated: Opens a TIFF image

Description

int **pdf_open_tiff** (int PDF-document, string filename)

Deprecated.

See also **pdf_open_image()**,

pdf_open

pdf_open -- Deprecated: Open a new pdf object

Description

pdf_open() is deprecated, use **pdf_new()** plus **pdf_open_file()** instead.

See also **pdf_new()**, **pdf_open_file()**.

pdf_place_image

(PHP 3>= 3.0.7, PHP 4)

pdf_place_image -- Places an image on the page

Description

void **pdf_place_image** (int pdf object, int image, float x, float y, float scale)

Place an image with the lower left corner at (*x*, *y*), and scale it.

pdf_place_pdi_page

(PHP 4 >= 4.0.6)

pdf_place_pdi_page -- Places an image on the page

Description

void **pdf_place_pdi_page** (int pdf object, int page, float x, float y, float sx, float sy)

Place a PDF page with the lower left corner at (*x*, *y*), and scale it.

pdf_rect

(PHP 3>= 3.0.6, PHP 4)

pdf_rect -- Draws a rectangle

Description

void **pdf_rect** (int pdf object, float x, float y, float width, float height)

Draw a rectangle at lower left (x, y) with width and height.

pdf_restore

(PHP 3>= 3.0.6, PHP 4)

pdf_restore -- Restores formerly saved environment

Description

void **pdf_restore** (int pdf object)

Restore the most recently saved graphics state.

pdf_rotate

(PHP 3>= 3.0.6, PHP 4)

pdf_rotate -- Sets rotation

Description

void **pdf_rotate** (int pdf object, float phi)

Rotate the coordinate system by phi degrees.

pdf_save

(PHP 3>= 3.0.6, PHP 4)

pdf_save -- Saves the current environment

Description

void **pdf_save** (int pdf object)

Save the current graphics state.

pdf_scale

(PHP 3>= 3.0.6, PHP 4)

pdf_scale -- Sets scaling

Description

void **pdf_scale** (int pdf object, float x-scale, float y-scale)

Scale the coordinate system.

pdf_set_border_color

(PHP 3>= 3.0.12, PHP 4)

pdf_set_border_color -- Sets color of border around links and annotations

Description

void **pdf_set_border_color** (int pdf object, float red, float green, float blue)

Set the border color for all kinds of annotations.

pdf_set_border_dash

(PHP 4 >= 4.0.1)

pdf_set_border_dash -- Sets dash style of border around links and annotations

Description

void **pdf_set_border_dash** (int pdf object, float black, float white)

Set the border dash style for all kinds of annotations. See **pdf_setdash()**.

pdf_set_border_style

(PHP 3>= 3.0.12, PHP 4)

pdf_set_border_style -- Sets style of border around links and annotations

Description

void **pdf_set_border_style** (int pdf object, string style, float width)

Set the border style for all kinds of annotations. *style* is "solid" or "dashed".

pdf_set_char_spacing

pdf_set_char_spacing -- Deprecated: Sets character spacing

Description

Deprecated.

See also **pdf_set_value()**,

pdf_set_duration

pdf_set_duration -- Deprecated: Sets duration between pages

Description

Deprecated.

See **pdf_set_value()**.

pdf_set_font

pdf_set_font -- Deprecated: Selects a font face and size

Description

Deprecated. You should use **pdf_findfont()** plus **pdf_setfont()** instead.

See **pdf_findfont()**, **pdf_setfont()**.

pdf_set_horiz_scaling

(PHP 3>= 3.0.6, PHP 4)

pdf_set_horiz_scaling -- Sets horizontal scaling of text

Description

void **pdf_set_horiz_scaling** (int pdf object, float scale)

Deprecated.

See also **pdf_set_value()**,

pdf_set_info_author

(PHP 3>= 3.0.6, PHP 4)

pdf_set_info_author -- Fills the author field of the document

Description

bool **pdf_set_info_author** (int pdfdoc, string author)

This function is deprecate, use **pdf_set_info()** instead.

pdf_set_info_creator

(PHP 3>= 3.0.6, PHP 4)

pdf_set_info_creator -- Fills the creator field of the document

Description

bool **pdf_set_info_creator** (int pdfdoc, string creator)

This function is deprecate, use **pdf_set_info()** instead.

pdf_set_info_keywords

(PHP 3>= 3.0.6, PHP 4)

pdf_set_info_keywords -- Fills the keywords field of the document

Description

bool **pdf_set_info_keywords** (int pdfdoc, string keywords)

This function is deprecate, use **pdf_set_info()** instead.

pdf_set_info_subject

(PHP 3>= 3.0.6, PHP 4)

pdf_set_info_subject -- Fills the subject field of the document

Description

bool **pdf_set_info_subject** (int pdfdoc, string subject)

This function is deprecate, use **pdf_set_info()** instead.

pdf_set_info_title

(PHP 3>= 3.0.6, PHP 4)

pdf_set_info_title -- Fills the title field of the document

Description

bool **pdf_set_info_title** (int pdfdoc, string title)

This function is deprecate, use **pdf_set_info()** instead.

pdf_set_info

(PHP 4 >= 4.0.1)

pdf_set_info -- Fills a field of the document information

Description

void **pdf_set_info** (int pdf object, string key, string value)

Fill document information field key with value. *key* is one of "Subject", "Title", "Creator", "Author", "Keywords", or a user-defined key.

pdf_set_leading

pdf_set_leading -- Deprecated: Sets distance between text lines

Description

Deprecated.

See also **pdf_set_value()**,

pdf_set_parameter

(PHP 4)

pdf_set_parameter -- Sets certain parameters

Description

void **pdf_set_parameter** (int pdf object, string key, string value)

Set some PDFlib parameter with string type.

pdf_set_text_matrix

pdf_set_text_matrix -- Deprecated: Sets the text matrix

Description

See **pdf_set_paramter()**.

pdf_set_text_pos

(PHP 3>= 3.0.6, PHP 4)

pdf_set_text_pos -- Sets text position

Description

void **pdf_set_text_pos** (int pdf object, float x, float y)

Set the text output position.

pdf_set_text_rendering

pdf_set_text_rendering -- Deprecated: Determines how text is rendered

Description

Deprecated.

See **pdf_set_value()**,

pdf_set_text_rise

pdf_set_text_rise -- Deprecated: Sets the text rise

Description

Deprecated.

See **pdf_set_value()**,

pdf_set_value

(PHP 4 >= 4.0.1)

pdf_set_value -- Sets certain numerical value

Description

void **pdf_set_value** (int pdf object, string key, float value)

Set the value of some PDFlib parameter with float type.

pdf_set_word_spacing

pdf_set_word_spacing -- Depriciated: Sets spacing between words

Description

Deprecated.

See also **pdf_set_value()**,

pdf_setcolor

(PHP 4 >= 4.0.5)

pdf_setcolor -- Sets fill and stroke color

Description

void **pdf_setcolor** (int pdf object, string type, string colorspace, float c1 [, float c2 [, float c3 [, float c4]]])

Set the current color space and color. The parameter *type* can be "fill", "stroke", or "both" to specify that the color is set for filling, stroking or both filling and stroking. The parameter *colorspace* can be `gray`, `rgb`, `cmyk`, `spot` or `pattern`. The parameters *c1*, *c2*, *c3* and *c4* represent the color components for the color space specified by *colorspace*. Except as otherwise noted, the color components are floating-point values that range from 0 to 1.

For `gray` only *c1* is used.

For `rgb` parameters *c1*, *c2*, and *c3* specify the red, green and blue values respectively.

```
// Set fill and stroke colors to white.
pdf_setcolor($pdf, "both", "rgb", 1, 1, 1);
```

For `cmyk`, parameters *c1*, *c2*, *c3*, and *c4* are the cyan, magenta, yellow and black values, respectively.

```
// Set fill and stroke colors to white.
pdf_setcolor($pdf, "both", "cmyk", 0, 0, 0, 1);
```

For `spot`, *c1* should be a spot color handles returned by **pdf_makespotcolor()** and *c2* is a tint value between 0 and 1.

For `pattern`, *c1* should be a pattern handle returned by **pdf_begin_pattern()**.

pdf_setdash

(PHP 3>= 3.0.6, PHP 4)

pdf_setdash -- Sets dash pattern

Description

void **pdf_setdash** (int pdf object, float b, float w)

Set the current dash pattern to *b* black and *w* white units.

pdf_setflat

(PHP 3>= 3.0.6, PHP 4)

pdf_setflat -- Sets flatness

Description

void **pdf_setflat** (int pdf object, float flatness)

Set the flatness to a value between 0 and 100 inclusive.

pdf_setfont

(PHP 4 >= 4.0.5)

pdf_setfont -- Set the current font

Description

void **pdf_setfont** (int pdf object, int font, float size)

Set the current font in the given size, using a *font* handle returned by **pdf_findfont()**

See Also: **pdf_findfont()**.

pdf_setgray_fill

(PHP 3>= 3.0.6, PHP 4)

pdf_setgray_fill -- Sets filling color to gray value

Description

void **pdf_setgray_fill** (int pdf object, float gray)

Set the current fill color to a gray value between 0 and 1 inclusive.

Note: PDFlib V4.0: Deprecated, use **pdf_setcolor()** instead.

pdf_setgray_stroke

(PHP 3>= 3.0.6, PHP 4)

pdf_setgray_stroke -- Sets drawing color to gray value

Description

void **pdf_setgray_stroke** (int pdf object, float gray)

Set the current stroke color to a gray value between 0 and 1 inclusive

Note: PDFlib V4.0: Deprecated, use **pdf_setcolor()** instead.

pdf_setgray

(PHP 3>= 3.0.6, PHP 4)

pdf_setgray -- Sets drawing and filling color to gray value

Description

void **pdf_setgray** (int pdf object, float gray)

Set the current fill and stroke color.

Note: PDFlib V4.0: Deprecated, use **pdf_setcolor()** instead.

pdf_setlinecap

(PHP 3>= 3.0.6, PHP 4)

pdf_setlinecap -- Sets linecap parameter

Description

void **pdf_setlinecap** (int pdf object, int linecap)

Set the *linecap* parameter to a value between 0 and 2 inclusive.

pdf_setlinejoin

(PHP 3>= 3.0.6, PHP 4)

pdf_setlinejoin -- Sets linejoin parameter

Description

void **pdf_setlinejoin** (int pdf object, long linejoin)

Set the line join parameter to a value between 0 and 2 inclusive.

pdf_setlinewidth

(PHP 3>= 3.0.6, PHP 4)

pdf_setlinewidth -- Sets line width

Description

void **pdf_setlinewidth** (int pdf object, float width)

Set the current linewidth to width.

pdf_setmatrix

(PHP 4 >= 4.0.5)

pdf_setmatrix -- Sets current transformation matrix

Description

void **pdf_setmatrix** (int pdf object, float a, float b, float c, float d, float e, float f)

Explicitly set the current transformation matrix.

pdf_setmiterlimit

(PHP 3>= 3.0.6, PHP 4)

pdf_setmiterlimit -- Sets miter limit

Description

void **pdf_setmiterlimit** (int pdf object, float miter)

Set the miter limit to a value greater than or equal to 1.

pdf_setpolydash

(PHP 4 >= 4.0.5)

pdf_setpolydash -- Sets complicated dash pattern

Description

void **pdf_setpolydash** (int pdf object, float * dasharray)

Set a more complicated dash pattern defined by an array.

pdf_setrgbcolor_fill

(PHP 3>= 3.0.6, PHP 4)

pdf_setrgbcolor_fill -- Sets filling color to rgb color value

Description

void **pdf_setrgbcolor_fill** (int pdf object, float red value, float green value, float blue value)

Set the current fill color to the supplied RGB values.

Note: PDFlib V4.0: Deprecated, use **pdf_setcolor()** instead.

pdf_setrgbcolor_stroke

(PHP 3>= 3.0.6, PHP 4)

pdf_setrgbcolor_stroke -- Sets drawing color to rgb color value

Description

void **pdf_setrgbcolor_stroke** (int pdf object, float red value, float green value, float blue value)

Set the current stroke color to the supplied RGB values.

Note: PDFlib V4.0: Deprecated, use **pdf_setcolor()** instead.

pdf_setrgbcolor

(PHP 3>= 3.0.6, PHP 4)

pdf_setrgbcolor -- Sets drawing and filling color to rgb color value

Description

void **pdf_setrgbcolor** (int pdf object, float red value, float green value, float blue value)

Set the current fill and stroke color to the supplied RGB values.

Note: PDFlib V4.0: Deprecated, use **pdf_setcolor()** instead.

pdf_show_boxed

(PHP 4)

pdf_show_boxed -- Output text in a box

Description

int **pdf_show_boxed** (int pdf object, string text, float left, float top, float width, float height, string hmode [, string feature])

Format text in the current font and size into the supplied text box according to the requested formatting mode, which must be one of "left", "right", "center", "justify", or "fulljustify". If width and height are o, only a single line is placed at the point (left, top) in the requested mode.

Returns the number of characters that did not fit in the specified box. Returns o if all characters fit or the *width* and *height* parameters were set to o for single-line formattting.

pdf_show_xy

(PHP 3>= 3.0.6, PHP 4)

pdf_show_xy -- Output text at given position

Description

void **pdf_show_xy** (int pdf object, string text, float x, float y)

Print text in the current font at (x, y).

pdf_show

(PHP 3>= 3.0.6, PHP 4)

pdf_show -- Output text at current position

Description

void **pdf_show** (int pdf object, string text)

Print text in the current font and size at the current position.

pdf_skew

(PHP 4)

pdf_skew -- Skews the coordinate system

Description

void **pdf_skew** (int pdf object, float alpha, float beta)

Skew the coordinate system in x and y direction by alpha and beta degrees.

pdf_stringwidth

(PHP 3>= 3.0.6, PHP 4)

pdf_stringwidth -- Returns width of text using current font

Description

float **pdf_stringwidth** (int pdf object, string text [, int font [, float size]])

Returns the width of *text* using the last font set by **pdf_setfont()**. If the optional parameters *font* and *size* are specified, the width will be calculated using that font and size instead. Please note that *font* is a font handle returned by **pdf_findfont()**.

> **Note:** Both the *font* and *size* parameters must used together.

See Also: **pdf_setfont()** and **pdf_findfont()**.

pdf_stroke

(PHP 3>= 3.0.6, PHP 4)

pdf_stroke -- Draws line along path

Description

void **pdf_stroke** (int pdf object)

Stroke the path with the current color and line width, and clear it.

pdf_translate

(PHP 3>= 3.0.6, PHP 4)

pdf_translate -- Sets origin of coordinate system

Description

void **pdf_translate** (int pdf object, float tx, float ty)

Translate the origin of the coordinate system.

LXXVIII. Verisign Payflow Pro functions

Introduction

This extension allows you to process credit cards and other financial transactions using Verisign Payment Services, formerly known as Signio (http://www.verisign.com/products/payflow/pro/index.html).

When using these functions, you may omit calls to **pfpro_init()** and **pfpro_cleanup()** as this extension will do so automatically if required. However the functions are still available in case you are processing a number of transactions and require fine control over the library. You may perform any number of transactions using **pfpro_process()** between the two.

These functions were added in PHP 4.0.2.

> **Note:** These functions only provide a link to Verisign Payment Services. Be sure to read the Payflow Pro Developers Guide for full details of the required parameters.

> **Note:** This extension is not available on Windows platforms.

Requirements

You will require the appropriate SDK for your platform, which may be downloaded from within the manager interface once you have registered. If you are going to use this extension in an SSL-enabled webserver or with other SSL components (such as the CURL+SSL extension) you MUST get the beta SDK.

Once you have downloaded the SDK you should copy the files from the `lib` directory of the distribution. Copy the header file `pfpro.h` to `/usr/local/include` and the library file `libpfpro.so` to `/usr/local/lib`.

Installation

These functions are only available if PHP has been compiled with the `--with-pfpro[=DIR]` option.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. Verisign Payflow Pro configuration options

Name	Default	Changeable
pfpro.defaulthost/PFPRO_VERSION < 3	"test.signio.com"	PHP_INI_ALL
pfpro.defaulthost	"test-payflow.verisign.com"	PHP_INI_ALL
pfpro.defaultport	"443"	PHP_INI_ALL
pfpro.defaulttimeout	"30"	PHP_INI_ALL
pfpro.proxyaddress	""	PHP_INI_ALL
pfpro.proxyport	""	PHP_INI_ALL
pfpro.proxylogon	""	PHP_INI_ALL
pfpro.proxypassword	""	PHP_INI_ALL

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Table of Contents

pfpro_cleanup

(PHP 4 >= 4.0.2)

pfpro_cleanup -- Shuts down the Payflow Pro library

Description

void **pfpro_cleanup** (void)

pfpro_cleanup() is used to shutdown the Payflow Pro library cleanly. It should be called after you have processed any transactions and before the end of your script. However you may omit this call, in which case this extension will automatically call **pfpro_cleanup()** after your script terminates.

See also **pfpro_init()**.

pfpro_init

(PHP 4 >= 4.0.2)

pfpro_init -- Initialises the Payflow Pro library

Description

void **pfpro_init** (void)

pfpro_init() is used to initialise the Payflow Pro library. You may omit this call, in which case this extension will automatically call **pfpro_init()** before the first transaction.

See also **pfpro_cleanup()**.

pfpro_process_raw

(PHP 4 >= 4.0.2)

pfpro_process_raw -- Process a raw transaction with Payflow Pro

Description

string **pfpro_process_raw** (string parameters [, string address [, int port [, int timeout [, string proxy address [, int proxy port [, string proxy logon [, string proxy password]]]]]]])

Returns: A string containing the response.

pfpro_process_raw() processes a raw transaction string with Payflow Pro. You should really use **pfpro_process()** instead, as the encoding rules of these transactions are non-standard.

The first parameter in this case is a string containing the raw transaction request. All other parameters are the same as with **pfpro_process()**. The return value is a string containing the raw response.

> **Note:** Be sure to read the Payflow Pro Developers Guide for full details of the required parameters and encoding rules. You would be well advised to use **pfpro_process()** instead.

Example 1. Payflow Pro raw example

```
<?php

pfpro_init();

$response = pfpro_process_raw("USER=mylogin&PWD[5]=m&ndy&PARTNER=VeriSign&TRXTYPE=S&TENDER=C&AMT=1.50&ACCT=4111111111111111&

if (!$response) {
  die("Couldn't establish link to Verisign.\n");
}

echo "Verisign raw response was ".$response;

pfpro_cleanup();

?>
```

pfpro_process

(PHP 4 >= 4.0.2)

pfpro_process -- Process a transaction with Payflow Pro

Description

array **pfpro_process** (array parameters [, string address [, int port [, int timeout [, string proxy address [, int proxy port [, string proxy logon [, string proxy password]]]]]]])

Returns: An associative array containing the response

pfpro_process() processes a transaction with Payflow Pro. The first parameter is an associative array containing keys and values that will be encoded and passed to the processor.

The second parameter is optional and specifies the host to connect to. By default this is "test.signio.com", so you will certainly want to change this to "connect.signio.com" in order to process live transactions.

The third parameter specifies the port to connect on. It defaults to 443, the standard SSL port.

The fourth parameter specifies the timeout to be used, in seconds. This defaults to 30 seconds. Note that this timeout appears to only begin once a link to the processor has been established and so your script could potentially continue for a very long time in the event of DNS or network problems.

The fifth parameter, if required, specifies the hostname of your SSL proxy. The sixth parameter specifies the port to use.

The seventh and eighth parameters specify the logon identity and password to use on the proxy.

The function returns an associative array of the keys and values in the response.

> **Note:** Be sure to read the Payflow Pro Developers Guide for full details of the required parameters.

Example 1. Payflow Pro example

```
<?php

pfpro_init();

$transaction = array('USER'    => 'mylogin',
                     'PWD'     => 'mypassword',
                     'PARTNER' => 'VeriSign',
                     'TRXTYPE' => 'S',
                     'TENDER'  => 'C',
                     'AMT'     => 1.50,
                     'ACCT'    => '4111111111111111',
                     'EXPDATE' => '0904'
                    );

$response = pfpro_process($transaction);

if (!$response) {
  die("Couldn't establish link to Verisign.\n");
}

echo "Verisign response code was ".$response['RESULT'];
echo ", which means: ".$response['RESPMSG']."\n";

echo "\nThe transaction request: ";
print_r($transaction);

echo "\nThe response: ";
print_r($response);

pfpro_cleanup();

?>
```

pfpro_version

(PHP 4 >= 4.0.2)

pfpro_version -- Returns the version of the Payflow Pro software

Description

string **pfpro_version** (void)

pfpro_version() returns the version string of the Payflow Pro library. At the time of writing, this was L211.

LXXIX. PHP Options&Information

Introduction

This functions enable you to get a lot of information about PHP itself, e.g. runtime configuration, loaded extensions, version and much more. You'll also find functions to set options for your running PHP. The probably best known function of PHP - **phpinfo()** - can be found here.

Requirements

No external libraries are needed to build this extension.

Installation

There is no installation needed to use these functions; they are part of the PHP core.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. PHP Options/Inf Configuration Options

Name	Default	Changeable
assert.active	"1"	PHP_INI_ALL
assert.bail	"0"	PHP_INI_ALL
assert.warning	"1"	PHP_INI_ALL
assert.callback	NULL	PHP_INI_ALL
assert.quiet_eval	"0"	PHP_INI_ALL
enable_dl	"1"	PHP_INI_SYSTEM
max_execution_time	"30"	PHP_INI_ALL
magic_quotes_gpc	"1"	PHP_INI_PERDIR|PHP_INI_SYSTEM
magic_quotes_runtime	"0"	PHP_INI_ALL

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Here is a short explanation of the configuration directives.

assert.active **boolean**

Enable **assert()** evaluation.

assert.bail **boolean**

Terminate script execution on failed assertions.

assert.warning **boolean**

Issue a PHP warning for each failed assertion.

assert.callback **string**

user function to call on failed assertions

assert.quiet_eval **boolean**

Use the current setting of **error_reporting()** during assertion expression evaluation. If enabled, no errors are shown (implicit error_reporting(0)) while evaluation. If disabled, errors are shown according to the settings of **error_reporting()**

enable_dl **boolean**

This directive is really only useful in the Apache module version of PHP. You can turn dynamic loading of PHP extensions with **dl()** on and off per virtual server or per directory.

The main reason for turning dynamic loading off is security. With dynamic loading, it's possible to ignore all open_basedir

restrictions. The default is to allow dynamic loading, except when using safe_mode. In safe-mode, it's always imposible to use **dl()**.

max_execution_time **integer**

This sets the maximum time in seconds a script is allowed to run before it is terminated by the parser. This helps prevent poorly written scripts from tying up the server. The default setting is `30`.

The maximum execution time is not affected by system calls, the **sleep()** function, etc. Please see the **set_time_limit()** function for more details.

magic_quotes_gpc **boolean**

Sets the magic_quotes state for GPC (Get/Post/Cookie) operations. When magic_quotes are on, all ' (single-quote), " (double quote), \ (backslash) and NUL's are escaped with a backslash automatically.

Note: If the magic_quotes_sybase directive is also ON it will completely override magic_quotes_gpc. Having both directives enabled means only single quotes are escaped as ''. Double quotes, backslashes and NUL's will remain untouched and unescaped.

See also **get_magic_quotes_gpc()**

magic_quotes_runtime **boolean**

If *magic_quotes_runtime* is enabled, most functions that return data from any sort of external source including databases and text files will have quotes escaped with a backslash. If magic_quotes_sybase is also on, a single-quote is escaped with a single-quote instead of a backslash.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are always available as part of the PHP core.

Table 2. Pre-defined phpcredits() constants

Constant	Description
CREDITS_ALL	All the credits, equivalent to using: CREDITS_DOCS + CREDITS_GENERAL + CREDITS_GROUP + CREDITS_MODULES + CREDITS_FULLPAGE. It generates a complete stand-alone HTML page with the appropriate tags.
CREDITS_DOCS	The credits for the documentation team
CREDITS_FULLPAGE	Usually used in combination with the other flags. Indicates that the a complete stand-alone HTML page needs to be printed including the information indicated by the other flags.
CREDITS_GENERAL	General credits: Language design and concept, PHP 4.0 authors and SAPI module.
CREDITS_GROUP	A list of the core developers
CREDITS_MODULES	A list of the extension modules for PHP, and their authors
CREDITS_SAPI	A list of the server API modules for PHP, and their authors

Table 3. phpinfo() constants

Constant	Value	Description
INFO_GENERAL	1	The configuration line, `php.ini` location, build date, Web Server, System and more.
INFO_CREDITS	2	PHP 4 Credits. See also **phpcredits()**.
INFO_CONFIGURATION	4	Current Local and Master values for php directives. See also **ini_get()**.
INFO_MODULES	8	Loaded modules and their respective settings.
INFO_ENVIRONMENT	16	Environment Variable information that's also available in `$_ENV`.
INFO_VARIABLES	32	Shows all predefined variables from EGPCS (Environment, GET, POST, Cookie, Server).
INFO_LICENSE	64	PHP License information. See also the license faq.

Constant	Value	Description
INFO_ALL	-1	Shows all of the above. This is the default value.

ASSERT_ACTIVE (integer)

ASSERT_CALLBACK (integer)

ASSERT_BAIL (integer)

ASSERT_WARNING (integer)

ASSERT_QUIET_EVAL (integer)

Table of Contents

assert_options

(PHP 4)

assert_options -- Set/get the various assert flags

Description

mixed **assert_options** (int what [, mixed value])

Using **assert_options()** you may set the various **assert()** control options or just query their current settings.

Table 1. Assert Options

option	ini-parameter	default	description
ASSERT_ACTIVE	assert.active	1	enable **assert()** evaluation
ASSERT_WARNING	assert.warning	1	issue a PHP warning for each failed assertion
ASSERT_BAIL	assert.bail	0	terminate execution on failed assertions
ASSERT_QUIET_EVAL	assert.quiet_eval	0	disable error_reporting during assertion expression evaluation
ASSERT_CALLBACK	assert_callback	(NULL)	user function to call on failed assertions

assert_options() will return the original setting of any option or FALSE on errors.

assert

(PHP 4)

assert -- Checks if assertion is FALSE

Description

int **assert** (string|bool assertion)

assert() will check the given *assertion* and take appropriate action if its result is FALSE.

If the *assertion* is given as a string it will be evaluated as PHP code by **assert()**. The advantages of a string *assertion* are less overhead when assertion checking is off and messages containing the *assertion* expression when an assertion fails. This means that if you pass a boolean condition as *assertion* this condition will not show up as parameter to the assertion function which you may have defined with the **assert_options()** function, the condition is converted to a string before calling that handler function, and the boolean FALSE is converted as the empty string.

Assertions should be used as a debugging feature only. You may use them for sanity-checks that test for conditions that should always be TRUE and that indicate some programming errors if not or to check for the presence of certain features like extension functions or certain system limits and features.

Assertions should not be used for normal runtime operations like input parameter checks. As a rule of thumb your code should always be able to work correctly if assertion checking is not activated.

The behavior of **assert()** may be configured by **assert_options()** or by .ini-settings described in that functions manual page.

The **assert_options()** function and/or ASSERT_CALLBACK configuration directive allow a callback function to be set to handle failed assertions.

assert() callbacks are particularly useful for building automated test suites because they allow you to easily capture the code passed to the assertion, along with information on where the assertion was made. While this information can be captured via other methods, using assertions makes it much faster and easier!

The callback function should accept three arguments. The first argument will contain the file the assertion failed in. The second argument will contain the line the assertion failed on and the third argument will contain the expression that failed (if any - literal values such as 1 or "two" will not be passed via this argument)

Example 1. Handle a failed assertion with a custom handler

```
<?php
// Active assert and make it quiet
assert_options (ASSERT_ACTIVE, 1);
assert_options (ASSERT_WARNING, 0);
assert_options (ASSERT_QUIET_EVAL, 1);

// Create a handler function
function my_assert_handler ($file, $line, $code) {
    echo "<hr>Assertion Failed:
        File '$file'<br>
        Line '$line'<br>
        Code '$code'<br><hr>";
}

// Set up the callback
assert_options (ASSERT_CALLBACK, 'my_assert_handler');

// Make an assertion that should fail
assert ('mysql_query ("")');
?>
```

dl

(PHP 3, PHP 4)

dl -- Loads a PHP extension at runtime

Description

bool **dl** (string library)

Loads the PHP extension given by the parameter *library*. The *library* parameter is *only* the filename of the extension to load which also depends on your platform. For example, the sockets extension (if compiled as a shared module, not the default!) would be called `sockets.so` on unix platforms whereas it is called `php_sockets.dll` on the windows platform.

Returns **TRUE** on success or **FALSE** on failure. If the functionality of loading modules is not available (see Note) or has been disabled (either by turning it off `enable_dl` or by enabling `safe_mode` in `php.ini`) an **E_ERROR** is emitted and execution is stopped. If **dl()** fails because the specified library couldn't be loaded, in addition to **FALSE** an **E_WARNING** message is emitted.

Use **extension_loaded()** to test whether a given extension is already available or not. This works on both built-in extensions and dynamically loaded ones (either through `php.ini` or **dl()**).

Example:

```
if (!extension_loaded('gd')) {
    if (!dl('gd.so')) {
        exit;
    }
}
```

The directory where the extension is loaded from depends on your platform:

Windows - If not explicitly set in the `php.ini`, the extension is loaded from `c:\php4\extensions\` by default.

Unix - If not explicitly set in the `php.ini`, the default extension directory depends on

- whether PHP has been built with `--enable-debug` or not
- whether PHP has been built with (experimental) ZTS (Zend Thread Safety) support or not
- the current internal `ZEND_MODULE_API_NO` (Zend internal module API number, which is basically the date on which a major module API change happened, e.g. `20010901`)

Taking into account the above, the directory then defaults to `<php-install-directory>/lib/php/extension/<debug-or-not>-<zts-or-not>-ZEND_MODULE_API_NO`, e.g. `/usr/local/php/lib/php/extensions/debug-non-zts-20010901` or `/usr/local/php/lib/php/extensions/no-debug-zts-20010901`.

> **Note: dl()** is *not* supported in multithreaded Web servers. Use the `extensions` statement in your `php.ini` when operating under such an environment. However, the `CGI` and `CLI` build are **not** affected !
>
> **Note: dl()** is case sensitive on unix platforms.

See also Extension Loading Directives and **extension_loaded()**.

extension_loaded

(PHP 3>= 3.0.10, PHP 4)

extension_loaded -- Find out whether an extension is loaded

Description

bool **extension_loaded** (string name)

Returns **TRUE** if the extension identified by *name* is loaded, **FALSE** otherwise.

Example:

```
if (!extension_loaded('gd')) {
    if (!dl('gd.so')) {
        exit;
```

```
    }
}
```

You can see the names of various extensions by using **phpinfo()** or if you're usnig the CGI or CLI version of PHP you can use the -m switch to list all available extensions:

```
$ php -m
[PHP Modules]
xml
tokenizer
standard
sockets
session
posix
pcre
overload
mysql
mbstring
ctype

[Zend Modules]
```

> **Note: extension_loaded()** uses the internal extension name to test whether a certain extension is available or not. Most internal extension names are written in lower case but there may be extension available which also use uppercase letters. Be warned that this function compares **case sensitive** !

See also **get_loaded_extensions()**, **get_extension_funcs()**, **phpinfo()** and **dl()**.

get_cfg_var

(PHP 3, PHP 4)

get_cfg_var -- Gets the value of a PHP configuration option

Description

string **get_cfg_var** (string varname)

Returns the current value of the PHP configuration variable specified by *varname*, or **FALSE** if an error occurs.

It will not return configuration information set when the PHP was compiled, or read from an Apache configuration file (using the php3_configuration_option directives).

To check whether the system is using a configuration file, try retrieving the value of the cfg_file_path configuration setting. If this is available, a configuration file is being used.

See also **ini_get()**.

get_current_user

(PHP 3, PHP 4)

get_current_user -- Gets the name of the owner of the current PHP script

Description

string **get_current_user** (void)

Returns the name of the owner of the current PHP script.

See also **getmyuid()**, **getmygid()**, **getmypid()**, **getmyinode()**, and **getlastmod()**.

get_defined_constants

(PHP 4 >= 4.1.0)

get_defined_constants -- Returns an associative array with the names of all the constants and their values

Description

array **get_defined_constants** (void)

This function returns the names and values of all the constants currently defined. This includes those created by extensions as well as those created with the **define()** function.

For example the line below

```
print_r (get_defined_constants());
```

will print a list like:

```
Array
(
    [E_ERROR] => 1
    [E_WARNING] => 2
    [E_PARSE] => 4
    [E_NOTICE] => 8
    [E_CORE_ERROR] => 16
    [E_CORE_WARNING] => 32
    [E_COMPILE_ERROR] => 64
    [E_COMPILE_WARNING] => 128
    [E_USER_ERROR] => 256
    [E_USER_WARNING] => 512
    [E_USER_NOTICE] => 1024
    [E_ALL] => 2047
    [TRUE] => 1
)
```

See also **get_loaded_extensions()**, **get_defined_functions()** and **get_defined_vars()**.

get_extension_funcs

(PHP 4)

get_extension_funcs -- Returns an array with the names of the functions of a module

Description

array **get_extension_funcs** (string module_name)

This function returns the names of all the functions defined in the module indicated by *module_name*.

For example the lines below

```
print_r (get_extension_funcs ("xml"));
print_r (get_extension_funcs ("gd"));
```

will print a list of the functions in the modules `xml` and `gd` respectively.

See also: **get_loaded_extensions()**

get_included_files

(PHP 4)

get_included_files -- Returns an array with the names of included or required files

Description

array **get_included_files** (void)

Returns an array of the names of all files that have been included using **include()**, **include_once()**, **require()** or **require_once()**.

Files that are included or required multiple times only show up once in the returned array.

> **Note:** Files included using the `auto_prepend_file` configuration directive are not included in the returned array.

Example 1. get_included_files() Example

```
<?php

include("test1.php");
include_once("test2.php");
require("test3.php");
require_once("test4.php");

$included_files = get_included_files();

foreach($included_files as $filename) {
    echo "$filename\n";
}

?>
```

will generate the following output:

```
test1.php
test2.php
test3.php
test4.php
```

> **Note:** In PHP 4.0.1pl2 and previous versions **get_included_files()** assumed that the required files ended in the extension `.php`; other extensions would not be returned. The array returned by **get_included_files()** was an associative array and only listed files included by **include()** and **include_once()**.

See also: **include()**, **include_once()**, **require()**, **require_once()**, and **get_required_files()**.

get_loaded_extensions

(PHP 4)

get_loaded_extensions -- Returns an array with the names of all modules compiled and loaded

Description

array **get_loaded_extensions** (void)

This function returns the names of all the modules compiled and loaded in the PHP interpreter.

For example the line below

```
<?php
print_r (get_loaded_extensions());
?>
```

will print a list like:

```
Array
(
   [0] => xml
   [1] => wddx
   [2] => standard
   [3] => session
   [4] => posix
   [5] => pgsql
   [6] => pcre
   [7] => gd
   [8] => ftp
   [9] => db
   [10] => calendar
   [11] => bcmath
)
```

See also **get_extension_funcs()**, **extension_loaded()**, **dl()** and **phpinfo()**.

get_magic_quotes_gpc

(PHP 3>= 3.0.6, PHP 4)

get_magic_quotes_gpc -- Gets the current active configuration setting of magic quotes gpc

Description

int **get_magic_quotes_gpc** (void)

Returns the current active configuration setting of magic_quotes_gpc (0 for off, 1 for on).

> **Note:** If the directive magic_quotes_sybase is ON it will completely override *magic_quotes_gpc*. So even when **get_magic_quotes()** returns **TRUE** neither double quotes, backslashes or NUL's will be escaped. Only single quotes will be escaped. In this case they'll look like: ''

Keep in mind that magic_quotes_gpc can not be set at runtime.

See also **addslashes()**, **stripslashes()**, **get_magic_quotes_runtime()**, and **ini_get()**.

get_magic_quotes_runtime

(PHP 3>= 3.0.6, PHP 4)

get_magic_quotes_runtime -- Gets the current active configuration setting of magic_quotes_runtime

Description

long **get_magic_quotes_runtime** (void)

Returns the current active configuration setting of magic_quotes_runtime (0 for off, 1 for on).

See also **get_magic_quotes_gpc()** and **set_magic_quotes_runtime()**.

get_required_files

(PHP 4)

get_required_files -- Returns an array with the names of included or required files

Description

array **get_required_files** (void)

As of PHP 4.0.4, this function is an alias for **get_included_files()**

In PHP 4.0.1pl2 and previous versions **get_required_files()** assumed that the required files ended in the extension `.php`, other extensions would not be returned. The array returned by **get_required_files()** was an associative array and only listed files included by **require()** and **require_once()**.

See also: **require()**, **require_once()**, **include()**, **include_once()**, and **get_included_files()**.

getenv

(PHP 3, PHP 4)

getenv -- Gets the value of an environment variable

Description

string **getenv** (string varname)

Returns the value of the environment variable *varname*, or **FALSE** on an error.

```
$ip = getenv ("REMOTE_ADDR"); // get the ip number of the user
```

You can see a list of all the environmental variables by using **phpinfo()**. You can find out what many of them mean by taking a look at the CGI specification, specifically the page on environmental variables.

Note: This function does not work in ISAPI mode.

See also **putenv()**.

getlastmod

(PHP 3, PHP 4)

getlastmod -- Gets time of last page modification

Description

int **getlastmod** (void)

Returns the time of the last modification of the current page. The value returned is a Unix timestamp, suitable for feeding to **date()**. Returns **FALSE** on error.

Example 1. getlastmod() example

```
<?php
// outputs e.g. 'Last modified: March 04 1998 20:43:59.'
echo "Last modified: " . date ("F d Y H:i:s.", getlastmod());
?>
```

Note: If you're interested in getting the last modification time of a different file, consider using **filemtime()**.

See also **date()**, **getmyuid()**, **getmygid()**, **get_current_user()**, **getmyinode()**, **getmypid()**, and **filemtime()**.

getmygid

(PHP 4 >= 4.1.0)

getmygid -- Get PHP script owner's GID

Description

int **getmygid** (void)

Returns the group ID of the current script, or **FALSE** on error.

See also **getmyuid()**, **getmypid()**, **get_current_user()**, **getmyinode()**, and **getlastmod()**.

getmyinode

(PHP 3, PHP 4)

getmyinode -- Gets the inode of the current script

Description

int **getmyinode** (void)

Returns the current script's inode, or **FALSE** on error.

See also **getmygid()**, **getmyuid()**, **get_current_user()**, **getmypid()**, and **getlastmod()**.

Note: This function is not implemented on Windows platforms.

getmypid

(PHP 3, PHP 4)

getmypid -- Gets PHP's process ID

Description

int **getmypid** (void)

Returns the current PHP process ID, or **FALSE** on error.

Warning
Process IDs are not unique, thus they are a weak entropy source. We recommend against relying on pids in security-dependent contexts.

See also **getmygid()**, **getmyuid()**, **get_current_user()**, **getmyinode()**, and **getlastmod()**.

getmyuid

(PHP 3, PHP 4)

getmyuid -- Gets PHP script owner's UID

Description

int **getmyuid** (void)

Returns the user ID of the current script, or **FALSE** on error.

See also **getmygid()**, **getmypid()**, **get_current_user()**, **getmyinode()**, and **getlastmod()**.

getopt

(PHP 4 >= 4.3.0)

getopt -- Gets options from the command line argument list

Description

string **getopt** (string options)

Returns an associative array of option / argument pairs based on the options format specified in *options*, or **FALSE** on an error.

```
$options = getopt("f:hp:"); // parse the command line ($_GLOBALS['argv'])
```

The *options* parameter may contain the following elements: individual characters, and characters followed by a colon to indicate an option argument is to follow. For example, an option string `x` recognizes an option `-x`, and an option string `x:` recognizes an option and argument `-x argument`. It does not matter if an argument has leading white space.

This function will return an array of option / argument pairs. If an option does not have an argument, the value will be set to **FALSE**.

> **Note:** This function is currently not available on Windows

getrusage

(PHP 3>= 3.0.7, PHP 4)

getrusage -- Gets the current resource usages

Description

array **getrusage** ([int who])

This is an interface to getrusage(2). It returns an associative array containing the data returned from the system call. If who is 1, getrusage will be called with RUSAGE_CHILDREN.

All entries are accessible by using their documented field names.

Example 1. Getrusage Example

```
$dat = getrusage();
echo $dat["ru_nswap"];         # number of swaps
echo $dat["ru_majflt"];        # number of page faults
echo $dat["ru_utime.tv_sec"];  # user time used (seconds)
echo $dat["ru_utime.tv_usec"]; # user time used (microseconds)
```

See your system's man page on getrusage(2) for more details.

ini_alter

(PHP 4)

ini_alter -- Changes the value of a configuration option

Description

string **ini_alter** (string varname, string newvalue)

Changes the value of a configuration option, returns **FALSE** on failure, and the previous value of the configuration option on success.

> **Note:** This is an alias of **ini_set()**

See also **ini_get()**, **ini_get_all()**, **ini_restore()**, and **ini_set()**

ini_get_all

(PHP 4 >= 4.2.0)

ini_get_all -- Gets all configuration options

Description

array **ini_get_all** ([string extension])

Returns all the registered configuration options as an associative array. If optional *extension* parameter is set, returns only options specific for that extension.

See also: **ini_alter()**, **ini_restore()**, **ini_get()**, and **ini_set()**

ini_get

(PHP 4)

ini_get -- Gets the value of a configuration option

Description

string **ini_get** (string varname)

Returns the value of the configuration option on success. Failure, such as querying for a non-existant value, will return an empty string.

> **When querying boolean values:** A boolean ini value of `off` will be returned as an empty string while a boolean ini value of `on` will be returned as "1".
>
> **When querying memory size values:** Many ini memory size values, such as `upload_max_filesize` are stored in the .ini file in shorthand notation. **ini_get()** will return the exact string stored in the php.ini file, *NOT* its integer equivalent. Attempting normal arithmetic functions on these values will not have otherwise expected results.
>
> ```
> <?php
> ```

```
/*
Our php.ini contains the following settings:

display_errors = On
register_globals = Off
post_max_size = 8M
*/

print 'display_errors = ' . ini_get('display_errors') . "\n";
print 'register_globals = ' . ini_get('register_globals') . "\n";
print 'post_max_size = ' . ini_get('post_max_size') . "\n";
print 'post_max_size+1 = ' . (ini_get('post_max_size')+1) . "\n";

/*
This script will produce:

display_errors = 1
register_globals =
post_max_size = 8M
post_max_size+1 = 9
*/
?>
```

See also **get_cfg_var()**, **ini_get_all()**, **ini_alter()**, **ini_restore()**, and **ini_set()**

ini_restore

(PHP 4)

ini_restore -- Restores the value of a configuration option

Description

string **ini_restore** (string varname)

Restores a given configuration option to its original value.

See also: **ini_alter()**, **ini_get()**, **ini_get_all()**, and **ini_set()**

ini_set

(PHP 4)

ini_set -- Sets the value of a configuration option

Description

string **ini_set** (string varname, string newvalue)

Sets the value of the given configuration option. Returns the old value on success, **FALSE** on failure. The configuration option will keep this new value during the script's execution, and will be restored at the script's ending.

Not all the available options can be changed using **ini_set()**. Below is a table with a list of all PHP options (as of PHP 4.2.0), indicating which ones can be changed/set and at what level.

Table 1. Configuration options

Name	Default	Changeable
com.allow_dcom	"0"	PHP_INI_SYSTEM
com.autoregister_typelib	"0"	PHP_INI_SYSTEM
com.autoregister_verbose	"0"	PHP_INI_SYSTEM
com.autoregister_casesensitive	"1"	PHP_INI_SYSTEM
com.typelib_file	""	PHP_INI_SYSTEM
crack.default_dictionary	NULL	PHP_INI_SYSTEM
exif.encode_unicode	"ISO-8859-15"	PHP_INI_ALL
exif.decode_unicode_motorola	"UCS-2BE"	PHP_INI_ALL
exif.decode_unicode_intel	"UCS-2LE"	PHP_INI_ALL

Name	Default	Changeable
exif.encode_jis	""	PHP_INI_ALL
exif.decode_jis_motorola	"JIS"	PHP_INI_ALL
exif.decode_jis_intel	"JIS"	PHP_INI_ALL
fbsql.allow_persistent	"1"	PHP_INI_SYSTEM
fbsql.generate_warnings	"0"	PHP_INI_SYSTEM
fbsql.autocommit	"1"	PHP_INI_SYSTEM
fbsql.max_persistent	"-1"	PHP_INI_SYSTEM
fbsql.max_links	"128"	PHP_INI_SYSTEM
fbsql.max_connections	"128"	PHP_INI_SYSTEM
fbsql.max_results	"128"	PHP_INI_SYSTEM
fbsql.batchSize	"1000"	PHP_INI_SYSTEM
fbsql.default_host	NULL	PHP_INI_SYSTEM
fbsql.default_user	"_SYSTEM"	PHP_INI_SYSTEM
fbsql.default_password	""	PHP_INI_SYSTEM
fbsql.default_database	""	PHP_INI_SYSTEM
fbsql.default_database_password	""	PHP_INI_SYSTEM
hwapi.allow_persistent	"0"	PHP_INI_SYSTEM
hyerwave.allow_persistent	"0"	PHP_INI_SYSTEM
hyperwave.default_port	"418"	PHP_INI_ALL
iconv.input_encoding	ICONV_INPUT_ENCODING	PHP_INI_ALL
iconv.output_encoding	ICONV_OUTPUT_ENCODING	PHP_INI_ALL
iconv.internal_encoding	ICONV_INTERNAL_ENCODING	PHP_INI_ALL
ifx.allow_persistent	"1"	PHP_INI_SYSTEM
ifx.max_persistent	"-1"	PHP_INI_SYSTEM
ifx.max_links	"-1"	PHP_INI_SYSTEM
ifx.default_host	NULL	PHP_INI_SYSTEM
ifx.default_user	NULL	PHP_INI_SYSTEM
ifx.default_password	NULL	PHP_INI_SYSTEM
ifx.blobinfile	"1"	PHP_INI_ALL
ifx.textasvarchar	"0"	PHP_INI_ALL
ifx.byteasvarchar	"0"	PHP_INI_ALL
ifx.charasvarchar	"0"	PHP_INI_ALL
ifx.nullformat	"0"	PHP_INI_ALL
ingres.allow_persistent	"1"	PHP_INI_SYSTEM
ingres.max_persistent	"-1"	PHP_INI_SYSTEM
ingres.max_links	"-1"	PHP_INI_SYSTEM
ingres.default_database	NULL	PHP_INI_ALL
ingres.default_user	NULL	PHP_INI_ALL
ingres.default_password	NULL	PHP_INI_ALL
ibase.allow_persistent	"1"	PHP_INI_SYSTEM
ibase.max_persistent	"-1"	PHP_INI_SYSTEM
ibase.max_links	"-1"	PHP_INI_SYSTEM
ibase.default_user	NULL	PHP_INI_ALL
ibase.default_password	NULL	PHP_INI_ALL
ibase.timestampformat	"%m/%d/%Y%H:%M:%S"	PHP_INI_ALL
ibase.dateformat	"%m/%d/%Y"	PHP_INI_ALL
ibase.timeformat	"%H:%M:%S"	PHP_INI_ALL
java.class.path	NULL	PHP_INI_ALL
java.home	NULL	PHP_INI_ALL
java.library.path	NULL	PHP_INI_ALL
java.library	JAVALIB	PHP_INI_ALL

Name	Default	Changeable
java.library	NULL	PHP_INI_ALL
ldap.max_links	"-1"	PHP_INI_SYSTEM
mbstring.detect_order	NULL	PHP_INI_ALL
mbstring.http_input	NULL	PHP_INI_ALL
mbstring.http_output	NULL	PHP_INI_ALL
mbstring.internal_encoding	NULL	PHP_INI_ALL
mbstring.substitute_character	NULL	PHP_INI_ALL
mbstring.func_overload	"0"	PHP_INI_SYSTEM
mcrypt.algorithms_dir	NULL	PHP_INI_ALL
mcrypt.modes_dir	NULL	PHP_INI_ALL
mime_magic.magicfile	"/usr/share/misc/magic.mime"	PHP_INI_SYSTEM
mssql.allow_persistent	"1"	PHP_INI_SYSTEM
mssql.max_persistent	"-1"	PHP_INI_SYSTEM
mssql.max_links	"-1"	PHP_INI_SYSTEM
mssql.max_procs	"25"	PHP_INI_ALL
mssql.min_error_severity	"10"	PHP_INI_ALL
mssql.min_message_severity	"10"	PHP_INI_ALL
mssql.compatability_mode	"0"	PHP_INI_ALL
mssql.connect_timeout	"5"	PHP_INI_ALL
mssql.timeout	"60"	PHP_INI_ALL
mssql.textsize	"-1"	PHP_INI_ALL
mssql.textlimit	"-1"	PHP_INI_ALL
mssql.batchsize	"0"	PHP_INI_ALL
mssql.datetimeconvert	"1"	PHP_INI_ALL
mysql.allow_persistent	"1"	PHP_INI_SYSTEM
mysql.max_persistent	"-1"	PHP_INI_SYSTEM
mysql.max_links	"-1"	PHP_INI_SYSTEM
mysql.default_host	NULL	PHP_INI_ALL
mysql.default_user	NULL	PHP_INI_ALL
mysql.default_password	NULL	PHP_INI_ALL
mysql.default_port	NULL	PHP_INI_ALL
mysql.default_socket	NULL	PHP_INI_ALL
ncurses.value	"42"	PHP_INI_ALL
ncurses.string	"foobar"	PHP_INI_ALL
odbc.allow_persistent	"1"	PHP_INI_SYSTEM
odbc.max_persistent	"-1"	PHP_INI_SYSTEM
odbc.max_links	"-1"	PHP_INI_SYSTEM
odbc.default_db	NULL	PHP_INI_ALL
odbc.default_user	NULL	PHP_INI_ALL
odbc.default_pw	NULL	PHP_INI_ALL
odbc.defaultlrl	"4096"	PHP_INI_ALL
odbc.defaultbinmode	"1"	PHP_INI_ALL
odbc.check_persistent	"1"	PHP_INI_SYSTEM
pfpro.defaulthost	"test.signio.com"	
pfpro.defaulthost	"test-payflow.verisign.com"	
pfpro.defaultport	"443"	PHP_INI_ALL
pfpro.defaulttimeout	"30"	PHP_INI_ALL
pfpro.proxyaddress	""	PHP_INI_ALL
pfpro.proxyport	""	PHP_INI_ALL
pfpro.proxylogon	""	PHP_INI_ALL
pfpro.proxypassword	""	PHP_INI_ALL

Name	Default	Changeable
pgsql.allow_persistent	"1"	PHP_INI_SYSTEM
pgsql.max_persistent	"-1"	PHP_INI_SYSTEM
pgsql.max_links	"-1"	PHP_INI_SYSTEM
pgsql.auto_reset_persistent	"0"	PHP_INI_SYSTEM
pgsql.ignore_notice	"0"	PHP_INI_ALL
pgsql.log_notice	"0"	PHP_INI_ALL
session.save_path	"/tmp"	PHP_INI_ALL
session.name	"PHPSESSID"	PHP_INI_ALL
session.save_handler	"files"	PHP_INI_ALL
session.auto_start	"0"	PHP_INI_ALL
session.gc_probability	"1"	PHP_INI_ALL
session.gc_maxlifetime	"1440"	PHP_INI_ALL
session.serialize_handler	"php"	PHP_INI_ALL
session.cookie_lifetime	"0"	PHP_INI_ALL
session.cookie_path	"/"	PHP_INI_ALL
session.cookie_domain	""	PHP_INI_ALL
session.cookie_secure	""	PHP_INI_ALL
session.use_cookies	"1"	PHP_INI_ALL
session.use_only_cookies	"0"	PHP_INI_ALL
session.referer_check	""	PHP_INI_ALL
session.entropy_file	""	PHP_INI_ALL
session.entropy_length	"0"	PHP_INI_ALL
session.cache_limiter	"nocache"	PHP_INI_ALL
session.cache_expire	"180"	PHP_INI_ALL
session.use_trans_sid	"1"	PHP_INI_SYSTEM|PHP_INI_PERDIR
session.encode_sources	"globals,track"	PHP_INI_ALL
assert.active	"1"	PHP_INI_ALL
assert.bail	"0"	PHP_INI_ALL
assert.warning	"1"	PHP_INI_ALL
assert.callback	NULL	PHP_INI_ALL
assert.quiet_eval	"0"	PHP_INI_ALL
safe_mode_protected_env_vars	SAFE_MODE_PROTECTED_ENV_VARS	PHP_INI_SYSTEM
safe_mode_allowed_env_vars	SAFE_MODE_ALLOWED_ENV_VARS	PHP_INI_SYSTEM
url_rewriter.tags	"a=href,area=href,frame=src,form=fakeentry"	PHP_INI_ALL
sybct.allow_persistent	"1"	PHP_INI_SYSTEM
sybct.max_persistent	"-1"	PHP_INI_SYSTEM
sybct.max_links	"-1"	PHP_INI_SYSTEM
sybct.min_server_severity	"10"	PHP_INI_ALL
sybct.min_client_severity	"10"	PHP_INI_ALL
sybct.hostname	NULL	PHP_INI_ALL
vpopmail.directory	""	PHP_INI_ALL
zlib.output_compression	"0"	PHP_INI_SYSTEM|PHP_INI_PERDIR
zlib.output_compression_level	"-1"	PHP_INI_ALL
define_syslog_variables	"0"	PHP_INI_ALL
highlight.bg	HL_BG_COLOR	PHP_INI_ALL
highlight.comment	HL_COMMENT_COLOR	PHP_INI_ALL
highlight.default	HL_DEFAULT_COLOR	PHP_INI_ALL
highlight.html	HL_HTML_COLOR	PHP_INI_ALL
highlight.keyword	HL_KEYWORD_COLOR	PHP_INI_ALL
highlight.string	HL_STRING_COLOR	PHP_INI_ALL
allow_call_time_pass_reference	"1"	PHP_INI_SYSTEM|PHP_INI_PERDIR

Name	Default	Changeable
asp_tags	"0"	PHP_INI_SYSTEM|PHP_INI_PERDIR
display_errors	"1"	PHP_INI_ALL
display_startup_errors	"0"	PHP_INI_ALL
enable_dl	"1"	PHP_INI_SYSTEM
expose_php	"1"	PHP_INI_SYSTEM
html_errors	"1"	PHP_INI_SYSTEM
xmlrpc_errors	"0"	PHP_INI_SYSTEM
xmlrpc_error_number	"0"	PHP_INI_ALL
ignore_user_abort	"0"	PHP_INI_ALL
implicit_flush	"0"	PHP_INI_PERDIR|PHP_INI_SYSTEM
log_errors	"0"	PHP_INI_ALL
log_errors_max_len	"1024"	PHP_INI_ALL
ignore_repeated_errors	"0"	PHP_INI_ALL
ignore_repeated_source	"0"	PHP_INI_ALL
magic_quotes_gpc	"1"	PHP_INI_PERDIR|PHP_INI_SYSTEM
magic_quotes_runtime	"0"	PHP_INI_ALL
magic_quotes_sybase	"0"	PHP_INI_ALL
output_buffering	"0"	PHP_INI_PERDIR|PHP_INI_SYSTEM
output_handler	NULL	PHP_INI_PERDIR|PHP_INI_SYSTEM
register_argc_argv	"1"	PHP_INI_PERDIR|PHP_INI_SYSTEM
register_globals	"0"	PHP_INI_PERDIR|PHP_INI_SYSTEM
safe_mode	"1"	PHP_INI_SYSTEM
safe_mode	"0"	PHP_INI_SYSTEM
safe_mode_include_dir	NULL	PHP_INI_SYSTEM
safe_mode_gid	"0"	PHP_INI_SYSTEM
short_open_tag	DEFAULT_SHORT_OPEN_TAG	PHP_INI_SYSTEM|PHP_INI_PERDIR
sql.safe_mode	"0"	PHP_INI_SYSTEM
track_errors	"0"	PHP_INI_ALL
y2k_compliance	"0"	PHP_INI_ALL
unserialize_callback_func	NULL	PHP_INI_ALL
arg_separator.output	"&"	PHP_INI_ALL
arg_separator.input	"&"	PHP_INI_SYSTEM|PHP_INI_PERDIR
auto_append_file	NULL	PHP_INI_SYSTEM|PHP_INI_PERDIR
auto_prepend_file	NULL	PHP_INI_SYSTEM|PHP_INI_PERDIR
doc_root	NULL	PHP_INI_SYSTEM
default_charset	SAPI_DEFAULT_CHARSET	PHP_INI_ALL
default_mimetype	SAPI_DEFAULT_MIMETYPE	PHP_INI_ALL
error_log	NULL	PHP_INI_ALL
extension_dir	PHP_EXTENSION_DIR	PHP_INI_SYSTEM
gpc_order	"GPC"	PHP_INI_ALL
include_path	PHP_INCLUDE_PATH	PHP_INI_ALL
max_execution_time	"30"	PHP_INI_ALL
open_basedir	NULL	PHP_INI_SYSTEM
safe_mode_exec_dir	"1"	PHP_INI_SYSTEM
upload_max_filesize	"2M"	PHP_INI_SYSTEM
file_uploads	"1"	PHP_INI_SYSTEM
post_max_size	"8M"	PHP_INI_SYSTEM
upload_tmp_dir	NULL	PHP_INI_SYSTEM
user_dir	NULL	PHP_INI_SYSTEM
variables_order	NULL	PHP_INI_ALL
error_append_string	NULL	PHP_INI_ALL

Name	Default	Changeable
error_prepend_string	NULL	PHP_INI_ALL
SMTP	"localhost"	PHP_INI_ALL
smtp_port	25	PHP_INI_ALL
browscap	NULL	PHP_INI_SYSTEM
error_reporting	NULL	PHP_INI_ALL
memory_limit	"8M"	PHP_INI_ALL
precision	"14"	PHP_INI_ALL
sendmail_from	NULL	PHP_INI_ALL
sendmail_path	DEFAULT_SENDMAIL_PATH	PHP_INI_SYSTEM
disable_functions	""	PHP_INI_SYSTEM
allow_url_fopen	"1"	PHP_INI_ALL
always_populate_raw_post_data	"0"	PHP_INI_ALL
xbithack	"0"	PHP_INI_ALL
engine	"1"	PHP_INI_ALL
last_modified	"0"	PHP_INI_ALL
child_terminate	"0"	PHP_INI_ALL
async_send	"0"	PHP_INI_ALL

Table 2. Definition of PHP_INI_* constants

Constant	Value	Meaning
PHP_INI_USER	1	Entry can be set in user scripts
PHP_INI_PERDIR	2	Entry can be set in `php.ini`, `.htaccess` or `httpd.conf`
PHP_INI_SYSTEM	4	Entry can be set in `php.ini` or `httpd.conf`
PHP_INI_ALL	7	Entry can be set anywhere

See also: **ini_alter()**, **ini_get()**, and **ini_restore()**

php_ini_scanned_files

(PHP 4 >= 4.3.0)

php_ini_scanned_files -- Return a list of .ini files parsed from the additional ini dir

Description

string **php_ini_scanned_files** (void)

php_ini_scanned_files() returns a comma-separated list of configuration files parsed after `php.ini`. These files are found in a directory defined by the `--with-config-file-scan-dir`. option which is set during compilation.

Returns a comma-separated string of .ini files on success. If the directive `--with-config-files-scan-dir` wasn't set, **FALSE** is returned. If it was set and the directory was empty, an empty string is returned. If a file is unreconizable, the file will still make it into the returned string but a PHP error will also result. This PHP error will be seen both at compile time and while using **php_ini_scanned_files()**.

The returned configuration files also include the path as declared in the `--with-config-file-scan-dir` directive. Also, each comma is followed by a newline.

Example 1. A simple example to list the returned ini files

```
<?php
if ($filelist = php_ini_scanned_files()) {
    if (strlen($filelist) > 0) {
        $files = explode(',', $filelist);
        foreach ($files as $file) {
```

```
            echo "<li>" . trim($file) . "</li>\n";
        }
    }
}
?>
```

See also **ini_set()** and **phpinfo()**.

php_logo_guid

(PHP 4)

php_logo_guid -- Gets the logo guid

Description

string **php_logo_guid** (void)

> **Note:** This functionality was added in PHP 4.0.0.

See also: **phpinfo()**, **phpversion()**, and **phpcredits()**.

php_sapi_name

(PHP 4 >= 4.0.1)

php_sapi_name -- Returns the type of interface between web server and PHP

Description

string **php_sapi_name** (void)

php_sapi_name() returns a lowercase string which describes the type of interface between web server and PHP (Server API, SAPI). In CGI PHP, this string is "cgi", in mod_php for Apache, this string is "apache" and so on.

Example 1. php_sapi_name() Example

```
$sapi_type = php_sapi_name();
if ($sapi_type == "cgi")
    print "You are using CGI PHP\n";
else
    print "You are not using CGI PHP\n";
```

php_uname

(PHP 4 >= 4.0.2)

php_uname -- Returns information about the operating system PHP was built on

Description

string **php_uname** (void)

php_uname() returns a string with a description of the operating system PHP is built on.

Example 1. php_uname() Example

```
if (substr(php_uname(), 0, 7) == "Windows") {
   die ("Sorry, this script doesn't run on Windows.\n");
}
```

phpcredits

(PHP 4)

phpcredits -- Prints out the credits for PHP

Description

void **phpcredits** ([int flag])

This function prints out the credits listing the PHP developers, modules, etc. It generates the appropriate HTML codes to insert the information in a page. *flag* is optional, and it defaults to **CREDITS_ALL**. To generate a custom credits page, you may want to use the *flag* parameter. For example to print the general credits, you will use somewhere in your code:

```
...
phpcredits(CREDITS_GENERAL);
...
```

And if you want to print the core developers and the documentation group, in a page of its own, you will use:

```
<?php
phpcredits(CREDITS_GROUP + CREDITS_DOCS + CREDITS_FULLPAGE);
?>
```

And if you feel like embedding all the credits in your page, then code like the one below will do it:

```
<html>
 <head>
  <title>My credits page</title>
 </head>
 <body>
  <?php
 // some code of your own
 phpcredits(CREDITS_ALL);
 // some more code
 ?>
 </body>
</html>
```

Table 1. Pre-defined phpcredits() flags

name	description
CREDITS_ALL	All the credits, equivalent to using: CREDITS_DOCS + CREDITS_GENERAL + CREDITS_GROUP + CREDITS_MODULES + CREDITS_FULLPAGE. It generates a complete stand-alone HTML page with the appropriate tags.
CREDITS_DOCS	The credits for the documentation team
CREDITS_FULLPAGE	Usually used in combination with the other flags. Indicates that the a complete stand-alone HTML page needs to be printed including the information indicated by the other flags.
CREDITS_GENERAL	General credits: Language design and concept, PHP 4.0 authors and SAPI module.
CREDITS_GROUP	A list of the core developers
CREDITS_MODULES	A list of the extension modules for PHP, and their authors
CREDITS_SAPI	A list of the server API modules for PHP, and their authors

See also: **phpinfo()**, **phpversion()**, and **php_logo_guid()**.

phpinfo

(PHP 3, PHP 4)

phpinfo -- Outputs lots of PHP information

Description

int **phpinfo** ([int what])

Outputs a large amount of information about the current state of PHP. This includes information about PHP compilation options and extensions, the PHP version, server information and environment (if compiled as a module), the PHP environment, OS version information, paths, master and local values of configuration options, HTTP headers, and the PHP License.

Because every system is setup differently, **phpinfo()** is commonly used to check configuration settings and for available predefined variables on a given system. Also, **phpinfo()** is a valuable debugging tool as it contains all EGPCS (Environment, GET, POST, Cookie, Server) data.

The output may be customized by passing one or more of the following *constants* bitwise values summed together in the optional *what* parameter. One can also combine the respective constants or bitwise values together with the or operator.

Table 1. phpinfo() options

Name (constant)	Value	Description
INFO_GENERAL	1	The configuration line, `php.ini` location, build date, Web Server, System and more.
INFO_CREDITS	2	PHP 4 Credits. See also **phpcredits()**.
INFO_CONFIGURATION	4	Current Local and Master values for php directives. See also **ini_get()**.
INFO_MODULES	8	Loaded modules and their respective settings.
INFO_ENVIRONMENT	16	Environment Variable information that's also available in `$_ENV`.
INFO_VARIABLES	32	Shows all predefined variables from EGPCS (Environment, GET, POST, Cookie, Server).
INFO_LICENSE	64	PHP License information. See also the license faq.
INFO_ALL	-1	Shows all of the above. This is the default value.

Example 1. phpinfo() examples

```
<?php

// Show all information, defaults to INFO_ALL
phpinfo();

// Show just the module information.
// phpinfo(8) yields identical results.
phpinfo(INFO_MODULES);

?>
```

> **Note:** Parts of the information displayed are disabled when the `expose_php` configuration setting is set to `off`. This includes the PHP and Zend logos, and the credits.

See also: **phpversion()**, **phpcredits()**, **php_logo_guid()**, **ini_get()**, **ini_set()**, and the section on Predefined Variables.

phpversion

(PHP 3, PHP 4)

phpversion -- Gets the current PHP version

Description

string **phpversion** (void)

Returns a string containing the version of the currently running PHP parser.

> **Note:** This information is also available in the predefined constant **PHP_VERSION**.

Example 1. phpversion() Example

```
<?php
// prints e.g. 'Current PHP version: 4.1.1'
echo 'Current PHP version: ' . phpversion();
?>
```

See also **version_compare()**, **phpinfo()**, **phpcredits()**, **php_logo_guid()**, and **zend_version()**.

putenv

(PHP 3, PHP 4)

putenv -- Sets the value of an environment variable

Description

void **putenv** (string setting)

Adds *setting* to the server environment. The environment variable will only exist for the duration of the current request. At the end of the request the environment is restored to its original state.

Setting certain environment variables may be a potential security breach. The safe_mode_allowed_env_vars directive contains a comma-delimited list of prefixes. In Safe Mode, the user may only alter environment variables whose names begin with the prefixes supplied by this directive. By default, users will only be able to set environment variables that begin with PHP_ (e.g. PHP_FOO=BAR). Note: if this directive is empty, PHP will let the user modify ANY environment variable!

The safe_mode_protected_env_vars directive contains a comma-delimited list of environment variables, that the end user won't be able to change using **putenv()**. These variables will be protected even if safe_mode_allowed_env_vars is set to allow to change them.

Warning
These directives have only effect when safe-mode itself is enabled!

Example 1. Setting an Environment Variable

```
putenv ("UNIQID=$uniqid");
```

See also **getenv()**.

set_magic_quotes_runtime

(PHP 3>= 3.0.6, PHP 4)

set_magic_quotes_runtime -- Sets the current active configuration setting of magic_quotes_runtime

Description

long **set_magic_quotes_runtime** (int new_setting)

Set the current active configuration setting of magic_quotes_runtime (0 for off, 1 for on).

See also: **get_magic_quotes_gpc()** and **get_magic_quotes_runtime()**.

set_time_limit

(PHP 3, PHP 4)

set_time_limit -- Limits the maximum execution time

Description

void **set_time_limit** (int seconds)

Set the number of seconds a script is allowed to run. If this is reached, the script returns a fatal error. The default limit is 30 seconds or, if it exists, the max_execution_time value defined in the configuration file. If seconds is set to zero, no time limit is imposed.

When called, **set_time_limit()** restarts the timeout counter from zero. In other words, if the timeout is the default 30 seconds, and 25 seconds into script execution a call such as set_time_limit(20) is made, the script will run for a total of 45 seconds before timing out.

set_time_limit() has no effect when PHP is running in safe mode. There is no workaround other than turning off safe mode or changing the time limit in the configuration file.

> **Note:** The **set_time_limit()** function and the configuration directive max_execution_time only affect the execution time of the script itself. Any time spent on activity that happens outside the execution of the script such as system calls using **system()**, the **sleep()** function, database queries, etc. is not included when determining the maximum time

that the script has been running.

version_compare

(PHP 4 >= 4.1.0)

version_compare -- Compares two "PHP-standardized" version number strings

Description

int **version_compare** (string version1, string version2 [, string operator])

version_compare() compares two "PHP-standardized" version number strings. This is useful if you would like to write programs working only on some versions of PHP.

version_compare() returns -1 if the first version is lower than the second, 0 if they are equal, and +1 if the second is lower.

If you specify the third optional *operator* argument, you can test for a particular relationship. The possible operators are: <, lt, <=, le, >, gt, >=, ge, ==, =, eq, !=, <>, ne respectively. Using this argument, the function will return 1 if the relationship is the one specified by the operator, 0 otherwise.

Example 1. version_compare() Example

```
// prints -1
echo version_compare("4.0.4", "4.0.6");

// these all print 1
echo version_compare("4.0.4", "4.0.6", "<");
echo version_compare("4.0.6", "4.0.6", "eq");
```

zend_logo_guid

(PHP 4)

zend_logo_guid -- Gets the zend guid

Description

string **zend_logo_guid** (void)

Note: This functionality was added in PHP 4.0.0.

zend_version

(PHP 4)

zend_version -- Gets the version of the current Zend engine

Description

string **zend_version** (void)

Returns a string containing the version of the currently running PHP parser.

Example 1. zend_version() Example

```
// prints e.g. 'Zend engine version: 1.0.4'
echo "Zend engine version: " . zend_version();
```

See also **phpinfo()**, **phpcredits()**, **php_logo_guid()**, and **phpversion()**.

LXXX. POSIX functions

Introduction

This module contains an interface to those functions defined in the IEEE 1003.1 (POSIX.1) standards document which are not accessible through other means. POSIX.1 for example defined the open(), read(), write() and close() functions, too, which traditionally have been part of PHP 3 for a long time. Some more system specific functions have not been available before, though, and this module tries to remedy this by providing easy access to these functions.

Warning
Sensitive data can be retrieved with the POSIX functions, e.g. **posix_getpwnam()** and friends. None of the POSIX function perform any kind of access checking when safe mode is enabled. It's therefore **strongly** advised to disable the POSIX extension at all (use `--disable-posix` in your configure line) if you're operating in such an environment.

Note: This extension is not available on Windows platforms.

Installation

POSIX functions are enabled by default. You can disable POSIX-like functions with `--disable-posix`.

See Also

The section about Process Control Functions maybe of interest for you.

Table of Contents

posix_ctermid

(PHP 3>= 3.0.13, PHP 4)

posix_ctermid -- Get path name of controlling terminal

Description

string **posix_ctermid** (void)

Needs to be written.

posix_getcwd

(PHP 3>= 3.0.13, PHP 4)

posix_getcwd -- Pathname of current directory

Description

string **posix_getcwd** (void)

Needs to be written ASAP.

posix_getegid

(PHP 3>= 3.0.10, PHP 4)

posix_getegid -- Return the effective group ID of the current process

Description

int **posix_getegid** (void)

Return the numeric effective group ID of the current process. See also **posix_getgrgid()** for information on how to convert this into a useable group name.

posix_geteuid

(PHP 3>= 3.0.10, PHP 4)

posix_geteuid -- Return the effective user ID of the current process

Description

int **posix_geteuid** (void)

Return the numeric effective user ID of the current process. See also **posix_getpwuid()** for information on how to convert this into a useable username.

posix_getgid

(PHP 3>= 3.0.10, PHP 4)

posix_getgid -- Return the real group ID of the current process

Description

int **posix_getgid** (void)

Return the numeric real group ID of the current process. See also **posix_getgrgid()** for information on how to convert this into a useable group name.

posix_getgrgid

(PHP 3>= 3.0.13, PHP 4)

posix_getgrgid -- Return info about a group by group id

Description

array **posix_getgrgid** (int gid)

Needs to be written.

posix_getgrnam

(PHP 3>= 3.0.13, PHP 4)

posix_getgrnam -- Return info about a group by name

Description

array **posix_getgrnam** (string name)

Needs to be written.

posix_getgroups

(PHP 3>= 3.0.10, PHP 4)

posix_getgroups -- Return the group set of the current process

Description

array **posix_getgroups** (void)

Returns an array of integers containing the numeric group ids of the group set of the current process. See also **posix_getgrgid()** for information on how to convert this into useable group names.

posix_getlogin

(PHP 3>= 3.0.13, PHP 4)

posix_getlogin -- Return login name

Description

string **posix_getlogin** (void)

Returns the login name of the user owning the current process. See **posix_getpwnam()** for information how to get more information about this user.

posix_getpgid

(PHP 3>= 3.0.10, PHP 4)

posix_getpgid -- Get process group id for job control

Description

int **posix_getpgid** (int pid)

Returns the process group identifier of the process *pid*.

This is not a POSIX function, but is common on BSD and System V systems. If your system does not support this function at system level, this PHP function will always return **FALSE**.

posix_getpgrp

(PHP 3>= 3.0.10, PHP 4)

posix_getpgrp -- Return the current process group identifier

Description

int **posix_getpgrp** (void)

Return the process group identifier of the current process. See POSIX.1 and the getpgrp(2) manual page on your POSIX system for more information on process groups.

posix_getpid

(PHP 3>= 3.0.10, PHP 4)

posix_getpid -- Return the current process identifier

Description

int **posix_getpid** (void)

Return the process identifier of the current process.

posix_getppid

(PHP 3>= 3.0.10, PHP 4)

posix_getppid -- Return the parent process identifier

Description

int **posix_getppid** (void)

Return the process identifier of the parent process of the current process.

posix_getpwnam

(PHP 3>= 3.0.13, PHP 4)

posix_getpwnam -- Return info about a user by username

Description

array **posix_getpwnam** (string username)

Returns an associative array containing information about a user referenced by an alphanumeric username, passed in the *username* parameter.

The array elements returned are:

Table 1. The user information array

Element	Description
name	The name element contains the username of the user. This is a short, usually less than 16 character "handle" of the user, not her real, full name. This should be the same as the *username* parameter used when calling the function, and hence redundant.
passwd	The passwd element contains the user's password in an encrypted format. Often, for example on a system employing "shadow" passwords, an asterisk is returned instead.
uid	User ID of the user in numeric form.
gid	The group ID of the user. Use the function **posix_getgrgid()** to resolve the group name and a list of its members.
gecos	GECOS is an obsolete term that refers to the finger information field on a Honeywell batch processing system. The field, however, lives on, and its contents have been formalized by POSIX. The field contains a comma separated list containing the user's full name, office phone, office number, and home phone number. On most systems, only the user's full name is available.
dir	This element contains the absolute path to the home directory of the user.
shell	The shell element contains the absolute path to the executable of the user's default shell.

posix_getpwuid

(PHP 3>= 3.0.13, PHP 4)

posix_getpwuid -- Return info about a user by user id

Description

array **posix_getpwuid** (int uid)

Returns an associative array containing information about a user referenced by a numeric user ID, passed in the *uid* parameter.

The array elements returned are:

Table 1. The user information array

Element	Description
name	The name element contains the username of the user. This is a short, usually less than 16 character "handle" of the user, not her real, full name.
passwd	The passwd element contains the user's password in an encrypted format. Often, for example on a system employing "shadow" passwords, an asterisk is returned instead.
uid	User ID, should be the same as the *uid* parameter used when calling the function, and hence redundant.
gid	The group ID of the user. Use the function **posix_getgrgid()** to resolve the group name and a list of its members.
gecos	GECOS is an obsolete term that refers to the finger information field on a Honeywell batch processing system. The field, however, lives on, and its contents have been formalized by POSIX. The field contains a comma separated list containing the user's full name, office phone, office number, and home phone number. On most systems, only the user's full name is available.
dir	This element contains the absolute path to the home directory of the user.
shell	The shell element contains the absolute path to the executable of the user's default shell.

posix_getrlimit

(PHP 3>= 3.0.10, PHP 4)

posix_getrlimit -- Return info about system ressource limits

Description

array **posix_getrlimit** (void)

Needs to be written ASAP.

posix_getsid

(PHP 3>= 3.0.10, PHP 4)

posix_getsid -- Get the current sid of the process

Description

int **posix_getsid** (int pid)

Return the sid of the process *pid*. If *pid* is 0, the sid of the current process is returned.

This is not a POSIX function, but is common on System V systems. If your system does not support this function at system level, this PHP function will always return **FALSE**.

posix_getuid

(PHP 3>= 3.0.10, PHP 4)

posix_getuid -- Return the real user ID of the current process

Description

int **posix_getuid** (void)

Return the numeric real user ID of the current process. See also **posix_getpwuid()** for information on how to convert this into a useable username.

posix_isatty

(PHP 3>= 3.0.13, PHP 4)

posix_isatty -- Determine if a file descriptor is an interactive terminal

Description

bool **posix_isatty** (int fd)

Needs to be written.

posix_kill

(PHP 3>= 3.0.13, PHP 4)

posix_kill -- Send a signal to a process

Description

bool **posix_kill** (int pid, int sig)

Send the signal *sig* to the process with the process identifier *pid*. Returns **FALSE**, if unable to send the signal, **TRUE** otherwise.

See also the kill(2) manual page of your POSIX system, which contains additional information about negative process identifiers, the special pid 0, the special pid -1, and the signal number 0.

posix_mkfifo

(PHP 3>= 3.0.13, PHP 4)

posix_mkfifo -- Create a fifo special file (a named pipe)

Description

bool **posix_mkfifo** (string pathname, int mode)

posix_mkfifo() creates a special `FIFO` file which exists in the file system and acts as a bidirectional communication endpoint for processes.

The second parameter *mode* has to be given in octal notation (e.g. 0644). The permission of the newly created `FIFO` also depends on the setting of the current **umask()**. The permissions of the created file are (mode & ~umask).

> **Note:** When safe mode is enabled, PHP checks whether the directory in which you are about to operate has the same UID (owner) as the script that is being executed.

posix_setegid

(PHP 4 >= 4.0.2)

posix_setegid -- Set the effective GID of the current process

Description

bool **posix_setegid** (int gid)

Set the effective group ID of the current process. This is a privileged function and you need appropriate privileges (usually root) on your system to be able to perform this function.

Returns **TRUE** on success, **FALSE** otherwise.

posix_seteuid

(PHP 4 >= 4.0.2)

posix_seteuid -- Set the effective UID of the current process

Description

bool **posix_seteuid** (int uid)

Set the real user ID of the current process. This is a privileged function and you need appropriate privileges (usually root) on your system to be able to perform this function.

Returns **TRUE** on success, **FALSE** otherwise. See also **posix_setgid()**.

posix_setgid

(PHP 3>= 3.0.13, PHP 4)

posix_setgid -- Set the GID of the current process

Description

bool **posix_setgid** (int gid)

Set the real group ID of the current process. This is a privileged function and you need appropriate privileges (usually root) on your system to be able to perform this function. The appropriate order of function calls is **posix_setgid()** first, **posix_setuid()** last.

Returns **TRUE** on success, **FALSE** otherwise.

posix_setpgid

(PHP 3>= 3.0.13, PHP 4)

posix_setpgid -- set process group id for job control

Description

int **posix_setpgid** (int pid, int pgid)

Let the process *pid* join the process group *pgid*. See POSIX.1 and the setsid(2) manual page on your POSIX system for more informations on process groups and job control. Returns **TRUE** on success, **FALSE** otherwise.

posix_setsid

(PHP 3>= 3.0.13, PHP 4)

posix_setsid -- Make the current process a session leader

Description

int **posix_setsid** (void)

Make the current process a session leader. See POSIX.1 and the setsid(2) manual page on your POSIX system for more informations on process groups and job control. Returns the session id.

posix_setuid

(PHP 3>= 3.0.13, PHP 4)

posix_setuid -- Set the UID of the current process

Description

bool **posix_setuid** (int uid)

Set the real user ID of the current process. This is a privileged function and you need appropriate privileges (usually root) on your system to be able to perform this function.

Returns **TRUE** on success, **FALSE** otherwise. See also **posix_setgid()**.

posix_times

(PHP 3>= 3.0.13, PHP 4)

posix_times -- Get process times

Description

array **posix_times** (void)

Returns a hash of strings with information about the current process CPU usage. The indices of the hash are

- ticks - the number of clock ticks that have elapsed since reboot.
- utime - user time used by the current process.
- stime - system time used by the current process.
- cutime - user time used by current process and children.
- cstime - system time used by current process and children.

posix_ttyname

(PHP 3>= 3.0.13, PHP 4)

posix_ttyname -- Determine terminal device name

Description

string **posix_ttyname** (int fd)

Needs to be written.

posix_uname

(PHP 3>= 3.0.10, PHP 4)

posix_uname -- Get system name

Description

array **posix_uname** (void)

Returns a hash of strings with information about the system. The indices of the hash are

- sysname - operating system name (e.g. Linux)
- nodename - system name (e.g. valiant)
- release - operating system release (e.g. 2.2.10)
- version - operating system version (e.g. #4 Tue Jul 20 17:01:36 MEST 1999)
- machine - system architecture (e.g. i586)
- domainname - DNS domainname (e.g. php.net)

domainname is a GNU extension and not part of POSIX.1, so this field is only available on GNU systems or when using the GNU libc.

Posix requires that you must not make any assumptions about the format of the values, e.g. you cannot rely on three digit version numbers or anything else returned by this function.

LXXXI. PostgreSQL functions

Introduction

PostgreSQL database is Open Source product and available without cost. Postgres, developed originally in the UC Berkeley Computer Science Department, pioneered many of the object-relational concepts now becoming available in some commercial databases. It provides SQL92/SQL99 language support, transaction integrity and type extensibility. PostgreSQL is an open source descendant of this original Berkeley code.

Requirements

To use PostgreSQL support, you need PostgreSQL 6.5 or later, PostgreSQL 7.0 or later to enable all PostgreSQL module features. PostgreSQL supports many character encoding including multibyte character encoding. The current version and more information about PostgreSQL is available at http://www.postgresql.org/.

Installation

In order to enable PostgreSQL support, `--with-pgsql[=DIR]` is required when you compile PHP. DIR is the PostgreSQL base install directory, defaults to `/usr/local/pgsql`. If shared object module is available, PostgreSQL module may be loaded using extension directive in `php.ini` or **dl()** function.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. PostgreSQL configuration options

Name	Default	Changeable
pgsql.allow_persistent	"1"	PHP_INI_SYSTEM
pgsql.max_persistent	"-1"	PHP_INI_SYSTEM
pgsql.max_links	"-1"	PHP_INI_SYSTEM
pgsql.auto_reset_persistent	"0"	PHP_INI_SYSTEM
pgsql.ignore_notice	"0"	PHP_INI_ALL
pgsql.log_notice	"0"	PHP_INI_ALL

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Here is a short explanation of the configuration directives.

`pgsql.allow_persistent` **boolean**

Whether to allow persistent Postgres connections.

`pgsql.max_persistent` **integer**

The maximum number of persistent Postgres connections per process.

`pgsql.max_links` **integer**

The maximum number of Postgres connections per process, including persistent connections.

How to use and hints

Warning
Using the PostgreSQL module with PHP 4.0.6 is not recommended due to a bug in the notice message handling code. Use 4.1.0 or later.

Warning
PostgreSQL function names will be changed in 4.2.0 release to confirm to current coding standards. Most of new names will have additional underscores, e.g. pg_lo_open(). Some functions are renamed to different name for consistency. e.g. pg_exec() to pg_query(). Older names can be used in 4.2.0 and a few releases from 4.2.0, but they may be deleted in the future.

Table 2. Function names changed

Old name	New name
pg_exec()	**pg_query()**
pg_getlastoid()	**pg_last_oid()**
pg_cmdtuples()	**pg_affected_rows()**
pg_numrows()	**pg_num_rows()**
pg_numfields()	**pg_num_fields()**
pg_fieldname()	**pg_field_name()**
pg_fieldsize()	**pg_field_size()**
pg_fieldnum()	**pg_field_num()**
pg_fieldprtlen()	**pg_field_prtlen()**
pg_fieldisnull()	**pg_field_is_null()**

Old name	New name
pg_freeresult()	pg_free_result()
pg_result()	pg_fetch_result()
pg_loreadall()	pg_lo_read_all()
pg_locreate()	pg_lo_create()
pg_lounlink()	pg_lo_unlink()
pg_loopen()	pg_lo_open()
pg_loclose()	pg_lo_close()
pg_loread()	pg_lo_read()
pg_lowrite()	pg_lo_write()
pg_loimport()	pg_lo_import()
pg_loexport()	pg_lo_export()

The old pg_connect()/pg_pconnect() syntax will be deprecated to support asynchronous connections in the future. Please use a connection string for pg_connect() and pg_pconnect().

Not all functions are supported by all builds. It depends on your libpq (The PostgreSQL C Client interface) version and how libpq is compiled. If there is missing function, libpq does not support the feature required for the function.

It is also important that you do not use an older libpq than the PostgreSQL Server to which you will be connecting. If you use libpq older than PostgreSQL Server expects, you may have problems.

Since version 6.3 (03/02/1998) PostgreSQL uses unix domain sockets by default. TCP port will NOT be opened by default. A table is shown below describing these new connection possibilities. This socket will be found in `/tmp/.s.PGSQL.5432`. This option can be enabled with the '-i' flag to **postmaster** and it's meaning is: "listen on TCP/IP sockets as well as Unix domain sockets".

Table 3. Postmaster and PHP

Postmaster	PHP	Status
postmaster &	pg_connect("dbname=MyDbName");	OK
postmaster -i &	pg_connect("dbname=MyDbName");	OK
postmaster &	pg_connect("host=localhost dbname=MyDbName");	Unable to connect to PostgreSQL server: connectDB() failed: Is the postmaster running and accepting TCP/IP (with -i) connection at 'localhost' on port '5432'? in /path/to/file.php on line 20.
postmaster -i &	pg_connect("host=localhost dbname=MyDbName");	OK

A connection to PostgreSQL server can be established with the following value pairs set in the command string: **$conn = pg_connect("host=myHost port=myPort tty=myTTY options=myOptions dbname=myDB user=myUser password=myPassword ");**

The previous syntax of: **$conn = pg_connect ("host", "port", "options", "tty", "dbname")** has been deprecated.

Environmental variables affect PostgreSQL server/client behavior. For example, PostgreSQL module will lookup PGHOST environment variable when the hostname is omitted in the connection string. Supported environment variables are different from version to version. Refer to PostgreSQL Programmer's Manual (libpq - Environment Variables) for details.

Make sure you set environment variables for appropriate user. Use `$_ENV` or getenv() to check which environment variables are available to the current process.

Example 1. Setting default parameters

```
PGHOST=pgsql.example.com
PGPORT=7890
PGDATABASE=web-system
PGUSER=web-user
PGPASSWORD=secret
PGDATESTYLE=ISO
PGTZ=JST
PGCLIENTENCODING=EUC-JP

export PGHOST PGPORT PGDATABASE PGUSER PGPASSWORD PGDATESTYLE PGTZ PGCLIENTENCODING
```

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

PGSQL_ASSOC (integer)

PGSQL_NUM (integer)

PGSQL_BOTH (integer)

PGSQL_CONNECTION_BAD (integer)

PGSQL_CONNECTION_OK (integer)

PGSQL_SEEK_SET (integer)

PGSQL_SEEK_CUR (integer)

PGSQL_SEEK_END (integer)

PGSQL_ESCAPE_STRING (integer)

PGSQL_ESCAPE_BYTEA (integer)

PGSQL_EMPTY_QUERY (integer)

PGSQL_COMMAND_OK (integer)

PGSQL_TUPLES_OK (integer)

PGSQL_COPY_OUT (integer)

PGSQL_COPY_IN (integer)

PGSQL_BAD_RESPONSE (integer)

PGSQL_NONFATAL_ERROR (integer)

PGSQL_FATAL_ERROR (integer)

Examples

Starting with PostgreSQL 7.1.0, you can store up to 1GB into a field of type text. In older versions, this was limited to the block size (default was 8KB, maximum was 32KB, defined at compile time)

To use the large object (lo) interface, it is required to enclose large object functions within a transaction block. A transaction block starts with a SQL statement **BEGIN** and if the transaction was valid ends with **COMMIT** or **END**. If the transaction fails the transaction should be closed with **ROLLBACK** or **ABORT**.

Example 2. Using Large Objects

```
<?php
    $database = pg_connect ("dbname=jacarta");
    pg_query ($database, "begin");
    $oid = pg_lo_create ($database);
    echo "$oid\n";
    $handle = pg_lo_open ($database, $oid, "w");
    echo "$handle\n";
    pg_lo_write ($handle, "large object data");
    pg_lo_close ($handle);
    pg_query ($database, "commit");
?>
```

You should not close the connection to the PostgreSQL server before closing the large object.

Table of Contents

pg_connection_reset -- Reset connection (reconnect)
pg_connection_status -- Get connection status
pg_convert -- Convert associative array value into suitable for SQL statement.
pg_copy_from -- Insert records into a table from an array
pg_copy_to -- Copy a table to an array
pg_dbname -- Get the database name
pg_delete -- Delete records.
pg_end_copy -- Sync with PostgreSQL backend
pg_escape_bytea -- Escape binary for bytea type
pg_escape_string -- Escape string for text/char type
pg_fetch_all -- Fetch a row as an array
pg_fetch_array -- Fetch a row as an array
pg_fetch_assoc -- Fetch a row as an array
pg_fetch_object -- Fetch a row as an object
pg_fetch_result -- Returns values from a result resource
pg_fetch_row -- Get a row as an enumerated array
pg_field_is_null -- Test if a field is NULL
pg_field_name -- Returns the name of a field
pg_field_num -- Returns the field number of the named field
pg_field_prtlen -- Returns the printed length
pg_field_size -- Returns the internal storage size of the named field
pg_field_type -- Returns the type name for the corresponding field number
pg_free_result -- Free result memory
pg_get_notify -- Ping database connection
pg_get_pid -- Ping database connection
pg_get_result -- Get asynchronous query result
pg_host -- Returns the host name associated with the connection
pg_insert -- Insert array into table.
pg_last_error -- Get the last error message string of a connection
pg_last_notice -- Returns the last notice message from PostgreSQL server
pg_last_oid -- Returns the last object's oid
pg_lo_close -- Close a large object
pg_lo_create -- Create a large object
pg_lo_export -- Export a large object to file
pg_lo_import -- Import a large object from file
pg_lo_open -- Open a large object
pg_lo_read_all -- Read a entire large object and send straight to browser
pg_lo_read -- Read a large object
pg_lo_seek -- Seeks position of large object
pg_lo_tell -- Returns current position of large object
pg_lo_unlink -- Delete a large object
pg_lo_write -- Write a large object
pg_meta_data -- Get meta data for table.
pg_num_fields -- Returns the number of fields
pg_num_rows -- Returns the number of rows
pg_options -- Get the options associated with the connection
pg_pconnect -- Open a persistent PostgreSQL connection
pg_ping -- Ping database connection
pg_port -- Return the port number associated with the connection
pg_put_line -- Send a NULL-terminated string to PostgreSQL backend
pg_query -- Execute a query
pg_result_error -- Get error message associated with result
pg_result_seek -- Set internal row offset in result resource
pg_result_status -- Get status of query result
pg_select -- Select records.
pg_send_query -- Send asynchronous query
pg_set_client_encoding -- Set the client encoding
pg_trace -- Enable tracing a PostgreSQL connection
pg_tty -- Return the tty name associated with the connection
pg_unescape_bytea -- Escape binary for bytea type
pg_untrace -- Disable tracing of a PostgreSQL connection
pg_update -- Update table.

pg_affected_rows

(PHP 4 >= 4.2.0)

pg_affected_rows -- Returns number of affected records(tuples)

Description

int **pg_affected_rows** (resource result)

pg_affected_rows() returns the number of tuples (instances/records/rows) affected by INSERT, UPDATE, and DELETE queries executed by **pg_query()**. If no tuple is affected by this function, it will return 0.

Example 1. pg_affected_rows()

```
<?php
     $result = pg_query ($conn, "INSERT INTO publisher VALUES ('Author')");
     $cmdtuples = pg_affected_rows ($result);
     echo $cmdtuples . " tuples are affected.";
?>
```

> **Note:** This function used to be called `pg_cmdtuples()`.

See also **pg_query()** and **pg_num_rows()**.

pg_cancel_query

(PHP 4 >= 4.2.0)

pg_cancel_query -- Cancel async query

Description

bool **pg_cancel_query** (resource connection)

pg_cancel_query() cancel asynchronous query sent by **pg_send_query()**. You cannot cancel query executed by **pg_query()**.

See also **pg_send_query()** and **pg_connection_busy()**

pg_client_encoding

(PHP 3 CVS only, PHP 4 >= 4.0.3)

pg_client_encoding -- Get the client encoding

Description

string **pg_client_encoding** ([resource connection])

pg_client_encoding() returns the client encoding as the string. The returned string should be either : SQL_ASCII, EUC_JP, EUC_CN, EUC_KR, EUC_TW, UNICODE, MULE_INTERNAL, LATINX (X=1...9), KOI8, WIN, ALT, SJIS, BIG5, WIN1250.

> **Note:** This function requires PHP-4.0.3 or higher and PostgreSQL-7.0 or higher. If libpq is compiled without multibyte encoding support, **pg_set_client_encoding()** always return "SQL_ASCII". Supported encoding depends on PostgreSQL version. Refer to PostgreSQL manual for details to enable multibyte support and encoding supported.
>
> The function used to be called **pg_clientencoding()**.

See also **pg_set_client_encoding()**.

pg_close

(PHP 3, PHP 4)

pg_close -- Close a PostgreSQL connection

Description

bool **pg_close** (resource connection)

pg_close() closes the non-persistent connection to a PostgreSQL database associated with the given *connection* resource. Returns **TRUE** on success or **FALSE** on failure.

> **Note:** Using **pg_close()** is not usually necessary, as non-persistent open connections are automatically closed at the end of the script.

If there is open large object resource on the connection, do not close the connection before closing all large object resources.

pg_connect

(PHP 3, PHP 4)

pg_connect -- Open a PostgreSQL connection

Description

resource **pg_connect** (string connection_string)

pg_connect() returns a connection resource that is needed by other PostgreSQL functions.

pg_connect() opens a connection to a PostgreSQL database specified by the *connection_string*. It returns a connection resource on success. It returns **FALSE** if the connection could not be made. *connection_string* should be a quoted string.

Example 1. Using pg_connect

```
<?php
$dbconn = pg_connect ("dbname=mary");
//connect to a database named "mary"
$dbconn2 = pg_connect ("host=localhost port=5432 dbname=mary");
// connect to a database named "mary" on "localhost" at port "5432"
$dbconn3 = pg_connect ("host=sheep port=5432 dbname=mary user=lamb password=foo");
//connect to a database named "mary" on the host "sheep" with a username and password
$conn_string = "host=sheep port=5432 dbname=test user=lamb password=bar";
$dbconn4 = pg_connect ($conn_string);
//connect to a database named "test" on the host "sheep" with a username and password
?>
```

The arguments available for *connection_string* includes *host*, *port*, *tty*, *options*, *dbname*, *user*, and *password*.

If a second call is made to **pg_connect()** with the same *connection_string*, no new connection will be established, but instead, the connection resource of the already opened connection will be returned. You can have multiple connections to the same database if you use different connection string.

The old syntax with multiple parameters **$conn = pg_connect ("host", "port", "options", "tty", "dbname")** has been deprecated.

See also **pg_pconnect()**, **pg_close()**, **pg_host()**, **pg_port()**, **pg_tty()**, **pg_options()** and **pg_dbname()**.

pg_connection_busy

(PHP 4 >= 4.2.0)

pg_connection_busy -- Get connection is busy or not

Description

bool **pg_connection_busy** (resource connection)

pg_connection_busy() returns **TRUE** if the connection is busy. If it is busy, a previous query is still executing. If **pg_get_result()** is called, it will be blocked.

See also **pg_connection_status()** and **pg_get_result()**

pg_connection_reset

(PHP 4 >= 4.2.0)

pg_connection_reset -- Reset connection (reconnect)

Description

bool **pg_connection_reset** (resource connection)

pg_connection_reset() resets the connection. It is useful for error recovery. Returns **TRUE** on success or **FALSE** on failure.

See also **pg_connect()**, **pg_pconnect()** and **pg_connection_status()**

pg_connection_status

(PHP 4 >= 4.2.0)

pg_connection_status -- Get connection status

Description

int **pg_connection_status** (resource connection)

pg_connection_status() returns a connection status. Possible statuses are `PGSQL_CONNECTION_OK` and `PGSQL_CONNECTION_BAD`.

See also **pg_connection_busy()**.

pg_convert

(PHP 4 >= 4.3.0)

pg_convert -- Convert associative array value into suitable for SQL statement.

Description

array **pg_convert** (resource connection, string table_name, array assoc_array [, int options])

pg_convert() check and convert `assoc_array` suitable for SQL statement.

> **Note:** This function is experimental.

See also **pg_metadata()**

pg_copy_from

(PHP 4 >= 4.2.0)

pg_copy_from -- Insert records into a table from an array

Description

bool **pg_copy_from** (resource connection, string table_name, array rows [, string delimiter [, string null_as]])

pg_copy_from() insert records into a table from *rows*. It issues `COPY FROM` SQL command internally to insert records. Returns **TRUE** on success or **FALSE** on failure.

See also **pg_copy_to()**

pg_copy_to

(PHP 4 >= 4.2.0)

pg_copy_to -- Copy a table to an array

Description

array **pg_copy_to** (resource connection, string table_name [, string delimiter [, string null_as]])

pg_copy_to() copies a table to an array. It issues `COPY TO` SQL command internally to insert records. The resulting array is returned. It returns **FALSE** on failure.

See also **pg_copy_from()**

pg_dbname

(PHP 3, PHP 4)

pg_dbname -- Get the database name

Description

string **pg_dbname** (resource connection)

pg_dbname() returns the name of the database that the given PostgreSQL *connection* resource. It returns **FALSE**, if *connection* is not a valid PostgreSQL connection resource.

pg_delete

(PHP 4 >= 4.3.0)

pg_delete -- Delete records.

Description

long **pg_delete** (resource connection, string table_name, array assoc_array [, int options])

pg_delete() deletes record condition specified by `assoc_array` which has `field=>value`. If `option` is specified, **pg_convert()** is applied to `assoc_array` with specified option.

Example 1. pg_delete

```
<?php
    $db = pg_connect ('dbname=foo');
    // This is safe, since $_POST is converted automatically
    $res = pg_delete($db, 'post_log', $_POST);
    if ($res) {
        echo "POST data is deleted: $res\n";
    }
    else {
        echo "User must have sent wrong inputs\n";
    }
?>
```

Note: This function is experimental.

See also **pg_convert()**

pg_end_copy

(PHP 4 >= 4.0.3)

pg_end_copy -- Sync with PostgreSQL backend

Description

bool **pg_end_copy** ([resource connection])

pg_end_copy() syncs the PostgreSQL frontend (usually a web server process) with the PostgreSQL server after doing a copy operation performed by **pg_put_line()**. **pg_end_copy()** must be issued, otherwise the PostgreSQL server may get out of sync with the frontend and will report an error. Returns **TRUE** on success or **FALSE** on failure.

For further details and an example, see also **pg_put_line()**.

pg_escape_bytea

(PHP 4 >= 4.2.0)

pg_escape_bytea -- Escape binary for bytea type

Description

string **pg_escape_bytea** (string data)

pg_escape_bytea() escapes string for bytea datatype. It returns escaped string.

> **Note:** When you SELECT bytea type, PostgreSQL returns octal byte value prefixed by \ (e.g. \032). Users are supposed to convert back to binary format by yourself.
>
> This function requires PostgreSQL 7.2 or later. With PostgreSQL 7.2.0 and 7.2.1, bytea type must be casted when you enable multi-byte support. i.e. `INSERT INTO test_table (image) VALUES ('$image_escaped'::bytea);` PostgreSQL 7.2.2 or later does not need cast. Exception is when client and backend character encoding does not match, there may be multi-byte stream error. User must cast to bytea to avoid this error.
>
> Newer PostgreSQL will support unescape function. Support for built-in unescape function will be added when it's available.

See also **pg_escape_string()**

pg_escape_string

(PHP 4 >= 4.2.0)

pg_escape_string -- Escape string for text/char type

Description

string **pg_escape_string** (string data)

pg_escape_string() escapes string for text/char datatype. It returns escaped string for PostgreSQL. Use of this function is recommended instead of **addslashes()**.

> **Note:** This function requires PostgreSQL 7.2 or later.

See also **pg_escape_bytea()**

pg_fetch_all

(PHP 4 >= 4.3.0)

pg_fetch_all -- Fetch a row as an array

Description

array **pg_fetch_all** (resource result [, int row])

pg_fetch_all() returns an array that contains all row (tuples/records) in result resource. It returns **FALSE**, if there are no more rows.

See also **pg_fetch_row()**, **pg_fetch_array()**, **pg_fetch_object()** and **pg_fetch_result()**.

Example 1. PostgreSQL fetch array

```
<?php
$conn = pg_pconnect ("dbname=publisher");
if (!$conn) {
    echo "An error occured.\n";
    exit;
}

$result = pg_query ($conn, "SELECT * FROM authors");
if (!$result) {
    echo "An error occured.\n";
    exit;
}

$arr = pg_fetch_all ($result, 0, PGSQL_NUM);

var_dump($arr);

?>
```

pg_fetch_array

(PHP 3>= 3.0.1, PHP 4)

pg_fetch_array -- Fetch a row as an array

Description

array **pg_fetch_array** (resource result [, int row [, int result_type]])

pg_fetch_array() returns an array that corresponds to the fetched row (tuples/records). It returns **FALSE**, if there are no more rows.

pg_fetch_array() is an extended version of **pg_fetch_row()**. In addition to storing the data in the numeric indices (field index) to the result array, it also stores the data in associative indices (field name) by default.

row is row (record) number to be retrieved. First row is 0.

result_type is optional parameter controls how return value is initialized. *result_type* is a constant and can take the following values: PGSQL_ASSOC, PGSQL_NUM, and PGSQL_BOTH. **pg_fetch_array()** returns associative array that has field name as key for PGSQL_ASSOC. field index as key with PGSQL_NUM and both field name/index as key with PGSQL_BOTH. Default is PGSQL_BOTH.

> **Note:** *result_type* was added in PHP 4.0.

pg_fetch_array() is NOT significantly slower than using **pg_fetch_row()**, while it provides a significant ease of use.

See also **pg_fetch_row()** and **pg_fetch_object()** and **pg_fetch_result()**.

Example 1. PostgreSQL fetch array

```
<?php
$conn = pg_pconnect ("dbname=publisher");
if (!$conn) {
    echo "An error occured.\n";
    exit;
}

$result = pg_query ($conn, "SELECT * FROM authors");
if (!$result) {
    echo "An error occured.\n";
    exit;
}

$arr = pg_fetch_array ($result, 0, PGSQL_NUM);
echo $arr[0] . " <- array\n";

$arr = pg_fetch_array ($result, 1, PGSQL_ASSOC);
echo $arr["author"] . " <- array\n";
?>
```

> **Note:** From 4.1.0, *row* became optional. Calling **pg_fetch_array()** will increment internal row counter by 1.

pg_fetch_assoc

(PHP 4 >= 4.3.0)

pg_fetch_assoc -- Fetch a row as an array

Description

array **pg_fetch_assoc** (resource result [, int row])

pg_fetch_assoc() returns an associative array that corresponds to the fetched row (tuples/records). It returns **FALSE**, if there are no more rows.

pg_fetch_assoc() is an extended version of **pg_fetch_row()**. In addition to storing the data in the numeric indices (field index) to the result array, it also stores the data in associative indices (field name) by default.

row is row (record) number to be retrieved. First row is 0.

pg_fetch_assoc() is NOT significantly slower than using **pg_fetch_row()**, while it provides a significant ease of use.

See also **pg_fetch_row()**, **pg_fetch_array()**, **pg_fetch_assoc()**, **pg_fetch_object()** and **pg_fetch_result()**.

Example 1. PostgreSQL fetch array

```
<?php
$conn = pg_pconnect ("dbname=publisher");
if (!$conn) {
    echo "An error occured.\n";
    exit;
}

$result = pg_query ($conn, "SELECT * FROM authors");
if (!$result) {
    echo "An error occured.\n";
    exit;
}

$arr = pg_fetch_assoc ($result, 1, PGSQL_ASSOC);
echo $arr["author"] . " <- array\n";
?>
```

pg_fetch_object

(PHP 3>= 3.0.1, PHP 4)

pg_fetch_object -- Fetch a row as an object

Description

object **pg_fetch_object** (resource result [, int row [, int result_type]])

pg_fetch_object() returns an object with properties that correspond to the fetched row. It returns **FALSE** if there are no more rows or error.

pg_fetch_object() is similar to **pg_fetch_array()**, with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

row is row (record) number to be retrieved. First row is 0.

Speed-wise, the function is identical to **pg_fetch_array()**, and almost as quick as **pg_fetch_row()** (the difference is insignificant).

> **Note:** From 4.1.0, *row* is optional.
>
> From 4.3.0, *result_type* is default to PGSQL_ASSOC while older versions' default was PGSQL_BOTH. There is no use for numeric property, since numeric property name is invalid in PHP.
>
> *result_type* may be deleted in future versions.

See also **pg_query()**, **pg_fetch_array()**, **pg_fetch_row()** and **pg_fetch_result()**.

Example 1. Postgres fetch object

```
<?php
$database = "verlag";
$db_conn = pg_connect ("host=localhost port=5432 dbname=$database");
if (!$db_conn): ?>
    <H1>Failed connecting to postgres database <?php echo $database ?></H1> <?php
    exit;
endif;

$qu = pg_query ($db_conn, "SELECT * FROM verlag ORDER BY autor");
$row = 0; // postgres needs a row counter other dbs might not

while ($data = pg_fetch_object ($qu, $row)) {
    echo $data->autor." (";
    echo $data->jahr .."): ";
    echo $data->titel."<BR>";
    $row++;
}
?>
<PRE>
<?php
$fields[] = Array ("autor", "Author");
$fields[] = Array ("jahr",  "  Year");
$fields[] = Array ("titel", " Title");

$row= 0; // postgres needs a row counter other dbs might not
while ($data = pg_fetch_object ($qu, $row)) {
    echo "----------\n";
    reset ($fields);
    while (list (,$item) = each ($fields)):
        echo $item[1].": ".$data->$item[0]."\n";
    endwhile;
    $row++;
}
echo "----------\n";
?>
</PRE>
<?php
pg_free_result ($qu);
pg_close ($db_conn);
?>
```

Note: From 4.1.0, *row* became optional. Calling **pg_fetch_object()** will increment internal row counter counter by 1.

pg_fetch_result

(PHP 4 >= 4.2.0)

pg_fetch_result -- Returns values from a result resource

Description

mixed **pg_fetch_result** (resource result, int row, mixed field)

pg_fetch_result() returns values from a *result* resource returned by **pg_query()**. *row* is integer. *field* is field name (string) or field index (integer). The *row* and *field* specify what cell in the table of results to return. Row numbering starts from 0. Instead of naming the field, you may use the field index as an unquoted number. Field indices start from 0.

PostgreSQL has many built in types and only the basic ones are directly supported here. All forms of **integer** types are returned as **integer** values. All forms of float, and real types are returned as **float** values. Boolean is returned as "t" or "f". All other types, including arrays are returned as strings formatted in the same default PostgreSQL manner that you would see in the **psql** program.

pg_fetch_row

(PHP 3>= 3.0.1, PHP 4)

pg_fetch_row -- Get a row as an enumerated array

Description

array **pg_fetch_row** (resource result, int row)

pg_fetch_row() fetches one row of data from the result associated with the specified *result* resource. The row (record) is returned as an array. Each result column is stored in an array offset, starting at offset 0.

It returns an array that corresponds to the fetched row, or **FALSE** if there are no more rows.

See also: pg_query(), pg_fetch_array(), pg_fetch_object() and pg_fetch_result().

Example 1. Postgres fetch row

```
<?php
$conn = pg_pconnect ("dbname=publisher");
if (!$conn) {
    echo "An error occured.\n";
    exit;
}

$result = pg_query ($conn, "SELECT * FROM authors");
if (!$result) {
    echo "An error occured.\n";
    exit;
}

while ($row = pg_fetch_row($result, $i)) {
  for ($j=0; $j < count($row); $j++) {
    echo "$row[$j] ";
  }

  echo "<BR>";

}

?>
```

Note: From 4.1.0, *row* became optional. Calling **pg_fetch_row()** will increment internal row counter by 1.

pg_field_is_null

(PHP 4 >= 4.2.0)

pg_field_is_null -- Test if a field is **NULL**

Description

int **pg_field_is_null** (resource result, int row, mixed field)

pg_field_is_null() test if a field is **NULL** or not. It returns 1 if the field in the given row is **NULL**. It returns 0 if the field in the given row is NOT **NULL**. Field can be specified as column index (number) or fieldname (string). Row numbering starts at 0.

Note: This function used to be called `pg_fieldisnull()`.

pg_field_name

(PHP 4 >= 4.2.0)

pg_field_name -- Returns the name of a field

Description

string **pg_field_name** (resource result, int field_number)

pg_field_name() returns the name of the field occupying the given *field_number* in the given PostgreSQL *result* resource. Field numbering starts from 0.

Note: This function used to be called `pg_fieldname()`.

See also pg_field_num().

pg_field_num

(PHP 4 >= 4.2.0)

pg_field_num -- Returns the field number of the named field

Description

int **pg_field_num** (resource result, string field_name)

pg_field_num() will return the number of the column (field) slot that corresponds to the *field_name* in the given PostgreSQL *result* resource. Field numbering starts at 0. This function will return -1 on error.

> **Note:** This function used to be called `pg_fieldnum()`.

See also **pg_field_name()**.

pg_field_prtlen

(PHP 4 >= 4.2.0)

pg_field_prtlen -- Returns the printed length

Description

int **pg_field_prtlen** (resource result, int row_number, string field_name)

pg_field_prtlen() returns the actual printed length (number of characters) of a specific value in a PostgreSQL *result*. Row numbering starts at 0. This function will return -1 on an error.

> **Note:** This function used to be called `pg_field_prtlen()`.

See also **pg_field_size()**.

pg_field_size

(PHP 4 >= 4.2.0)

pg_field_size -- Returns the internal storage size of the named field

Description

int **pg_field_size** (resource result, int field_number)

pg_field_size() returns the internal storage size (in bytes) of the field number in the given PostgreSQL *result*. Field numbering starts at 0. A field size of -1 indicates a variable length field. This function will return **FALSE** on error.

> **Note:** This function used to be called `pg_fieldsize()`.

See also **pg_field_prtlen()** and **pg_field_type()**.

pg_field_type

(PHP 4 >= 4.2.0)

pg_field_type -- Returns the type name for the corresponding field number

Description

string **pg_field_type** (resource result, int field_number)

pg_field_type() returns a string containing the type name of the given *field_number* in the given PostgreSQL *result* resource. Field numbering starts at 0.

> **Note:** This function used to be called `pg_fieldtype()`.

See also **pg_field_len()** and **pg_field_name()**.

pg_free_result

(PHP 4 >= 4.2.0)

pg_free_result -- Free result memory

Description

bool **pg_free_result** (resource result)

pg_free_result() only needs to be called if you are worried about using too much memory while your script is running. All result memory will automatically be freed when the script is finished. But, if you are sure you are not going to need the result data anymore in a script, you may call **pg_free_result()** with the *result* resource as an argument and the associated result memory will be freed. Returns **TRUE** on success or **FALSE** on failure.

> **Note:** This function used to be called `pg_freeresult()`.

See also **pg_query()**.

pg_get_notify

(PHP 4 >= 4.3.0)

pg_get_notify -- Ping database connection

Description

array **pg_get_notify** (resource connection [, int result_type])

pg_get_notify() gets notify message sent by `NOTIFY` SQL command. To recieve nofigy messages, `LISTEN` SQL command must be issued. If there is notify message on the connection, array contains message name and backend PID is returned. If there is no message, **FALSE** is returned.

See also **pg_get_pid()**

Example 1. PostgreSQL NOTIFY message

```
<?php
$conn = pg_pconnect ("dbname=publisher");
if (!$conn) {
    echo "An error occured.\n";
    exit;
}

// Listen 'author_updated' message from other processes
pq_query($conn, 'LISTEN author_updated;');
$notify = pg_get_notify($conn);
if (!$notify)
    print("No messages\n");
else
    print_r($notify);
?>
```

pg_get_pid

(PHP 4 >= 4.3.0)

pg_get_pid -- Ping database connection

Description

int **pg_get_pid** (resource connection)

pg_get_pid() gets backend (database server process) PID. PID is useful to check if `NOTIFY` message is sent from other process or not.

See also **pg_get_notify()**

Example 1. PostgreSQL backend PID

```
<?php
$conn = pg_pconnect ("dbname=publisher");
if (!$conn) {
    echo "An error occured.\n";
    exit;
}

// Backend process PID. Use PID with pg_get_notify()
$pid = pg_get_pid($conn);
?>
```

pg_get_result

(PHP 4 >= 4.2.0)

pg_get_result -- Get asynchronous query result

Description

resource **pg_get_result** ([resource connection])

pg_get_result() get result resource from async query executed by **pg_send_query()**. **pg_send_query()** can send multiple queries to PostgreSQL server and **pg_get_result()** is used to get query result one by one. It returns result resource. If there is no more results, it returns **FALSE**.

pg_host

(PHP 3, PHP 4)

pg_host -- Returns the host name associated with the connection

Description

string **pg_host** (resource connection)

pg_host() returns the host name of the given PostgreSQL *connection* resource is connected to.

See also **pg_connect()** and **pg_pconnect()**.

pg_insert

(PHP 4 >= 4.3.0)

pg_insert -- Insert array into table.

Description

bool **pg_insert** (resource connection, string table_name, array assoc_array [, int options])

pg_insert() inserts `assoc_array` which has `field=>value` into table specified as `table_name`. If `options` is specified, **pg_convert()** is applied to `assoc_array` with specified option.

Example 1. pg_insert

```
<?php
    $db = pg_connect ('dbname=foo');
    // This is safe, since $_POST is converted automatically
    $res = pg_insert($db, 'post_log', $_POST);
    if ($res) {
        echo "POST data is succesfully logged\n";
    }
```

```
    else {
        echo "User must have sent wrong inputs\n";
    }
?>
```

Note: This function is experimental.

See also **pg_convert()**

pg_last_error

(PHP 4 >= 4.2.0)

pg_last_error -- Get the last error message string of a connection

Description

string **pg_last_error** ([resource connection])

pg_last_error() returns the last error message for given *connection*.

Error messages may be overwritten by internal PostgreSQL(libpq) function calls. It may not return appropriate error message, if multiple errors are occured inside a PostgreSQL module function.

Use **pg_result_error()**, **pg_result_status()** and **pg_connection_status()** for better error handling.

Note: This function used to be called `pg_errormessage()`.

See also **pg_result_error()**.

pg_last_notice

(PHP 4 >= 4.0.6)

pg_last_notice -- Returns the last notice message from PostgreSQL server

Description

string **pg_last_notice** (resource connection)

pg_last_notice() returns the last notice message from the PostgreSQL server specified by *connection*. The PostgreSQL server sends notice messages in several cases, e.g. if the transactions can't be continued. With **pg_last_notice()**, you can avoid issuing useless queries, by checking whether the notice is related to the transaction or not.

Warning
This function is EXPERIMENTAL and it is not fully implemented yet. **pg_last_notice()** was added in PHP 4.0.6. However, PHP 4.0.6 has problem with notice message handling. Use of the PostgreSQL module with PHP 4.0.6 is not recommended even if you are not using **pg_last_notice()**. This function is fully implemented in PHP 4.3.0. PHP earlier than PHP 4.3.0 ignores database connection parameter.

Notice message tracking can be set to optional by setting 1 for `pgsql.ignore_notice` ini from PHP 4.3.0.

Notice message logging can be set to optional by setting 0 for `pgsql.log_notice` ini from PHP 4.3.0. Unless `pgsql.ignore_notice` is set to 0, notice message cannot be logged.

See also **pg_query()** and **pg_last_error()**.

pg_last_oid

(PHP 4 >= 4.2.0)

pg_last_oid -- Returns the last object's oid

Description

int **pg_last_oid** (resource result)

pg_last_oid() is used to retrieve the `oid` assigned to an inserted tuple (record) if the result resource is used from the last command sent via **pg_query()** and was an SQL INSERT. Returns a positive integer if there was a valid `oid`. It returns **FALSE** if an error occurs or the last command sent via **pg_query()** was not an INSERT or INSERT is failed.

OID field became an optional field from PostgreSQL 7.2. When OID field is not defined in a table, programmer must use **pg_result_status()** to check if record is is inserted successfully or not.

> **Note:** This function used to be called `pg_getlastoid()`.

See also **pg_query()** and **pg_result_status()**

pg_lo_close

(PHP 4 >= 4.2.0)

pg_lo_close -- Close a large object

Description

bool **pg_lo_close** (resource large_object)

pg_lo_close() closes a Large Object. *`large_object`* is a resource for the large object from **pg_lo_open()**.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

> **Note:** This function used to be called `pg_loclose()`.

See also **pg_lo_open()**, **pg_lo_create()** and **pg_lo_import()**.

pg_lo_create

(PHP 4 >= 4.2.0)

pg_lo_create -- Create a large object

Description

int **pg_lo_create** (resource connection)

pg_lo_create() creates a Large Object and returns the `oid` of the large object. *`connection`* specifies a valid database connection opened by **pg_connect()** or **pg_pconnect()**. PostgreSQL access modes INV_READ, INV_WRITE, and INV_ARCHIVE are not supported, the object is created always with both read and write access. INV_ARCHIVE has been removed from PostgreSQL itself (version 6.3 and above). It returns large object oid, otherwise it returns **FALSE** if an error occurred.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

> **Note:** This function used to be called `pg_locreate()`.

pg_lo_export

(PHP 4 >= 4.2.0)

pg_lo_export -- Export a large object to file

Description

bool **pg_lo_export** (int oid, string pathname [, resource connection])

The *oid* argument specifies oid of the large object to export and the *pathname* argument specifies the pathname of the file. It returns **FALSE** if an error occurred, **TRUE** otherwise.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

> **Note:** This function used to be called `pg_loexport()`.

See also **pg_lo_import()**.

pg_lo_import

(PHP 4 >= 4.2.0)

pg_lo_import -- Import a large object from file

Description

int **pg_lo_import** ([resource connection, string pathname])

In versions before PHP 4.2.0 the syntax of this function was different, see the following definition:

int **pg_lo_import** (string pathname [, resource connection])

The *pathname* argument specifies the pathname of the file to be imported as a large object. It returns **FALSE** if an error occurred, oid of the just created large object otherwise.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

> **Note:** When safe mode is enabled, PHP checks whether the files or directories you are about to operate on have the same UID (owner) as the script that is being executed.

> **Note:** This function used to be called `pg_loimport()`.

See also **pg_lo_export()** and **pg_lo_open()**.

pg_lo_open

(PHP 4 >= 4.2.0)

pg_lo_open -- Open a large object

Description

resource **pg_lo_open** (resource connection, int oid, string mode)

pg_lo_open() opens a Large Object and returns large object resource. The resource encapsulates information about the connection. *oid* specifies a valid large object oid and *mode* can be either `"r"`, `"w"`, or `"rw"`. It returns **FALSE** if there is an error.

Warning
Do not close the database connection before closing the large object resource.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

> **Note:** This function used to be called `pg_loopen()`.

See also **pg_lo_close()** and **pg_lo_create()**.

pg_lo_read_all

(PHP 4 >= 4.2.0)

pg_lo_read_all -- Read a entire large object and send straight to browser

Description

int **pg_lo_read_all** (resource large_object)

pg_lo_read_all() reads a large object and passes it straight through to the browser after sending all pending headers. Mainly intended for sending binary data like images or sound. It returns number of bytes read. It returns **FALSE**, if an error occurred.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

> **Note:** This function used to be called `pg_loreadall()`.

See also **pg_lo_read()**.

pg_lo_read

(PHP 4 >= 4.2.0)

pg_lo_read -- Read a large object

Description

string **pg_lo_read** (resource large_object, int len)

pg_lo_read() reads at most *len* bytes from a large object and returns it as a string. *large_object* specifies a valid large object resource and *len* specifies the maximum allowable size of the large object segment. It returns **FALSE** if there is an error.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

> **Note:** This function used to be called `pg_loread()`.

See also **pg_lo_read_all()**.

pg_lo_seek

(PHP 4 >= 4.2.0)

pg_lo_seek -- Seeks position of large object

Description

bool **pg_lo_seek** (resource large_object, int offset [, int whence])

pg_lo_seek() seeks position of large object resource. *whence* is PGSQL_SEEK_SET, PGSQL_SEEK_CUR or PGSQL_SEEK_END.

See also **pg_lo_tell()**.

pg_lo_tell

(PHP 4 >= 4.2.0)

pg_lo_tell -- Returns current position of large object

Description

int **pg_lo_tell** (resource large_object)

pg_lo_tell() returns current position (offset from the beginning of large object).

See also **pg_lo_seek()**.

pg_lo_unlink

(PHP 4 >= 4.2.0)

pg_lo_unlink -- Delete a large object

Description

bool **pg_lo_unlink** (resource connection, int oid)

pg_lo_unlink() deletes a large object with the *oid*. Returns **TRUE** on success or **FALSE** on failure.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

> **Note:** This function used to be called `pg_lo_unlink()`.

See also **pg_lo_create()** and **pg_lo_import()**.

pg_lo_write

(PHP 4 >= 4.2.0)

pg_lo_write -- Write a large object

Description

int **pg_lo_write** (resource large_object, string data)

pg_lo_write() writes at most to a large object from a variable *data* and returns the number of bytes actually written, or **FALSE** in the case of an error. *large_object* is a large object resource from **pg_lo_open()**.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

> **Note:** This function used to be called `pg_lo_write()`.

See also **pg_lo_create()** and **pg_lo_open()**.

pg_meta_data

(PHP 4 >= 4.3.0)

pg_meta_data -- Get meta data for table.

Description

array **pg_meta_data** (resource connection, string table_name)

pg_meta_data() returns table definition for `table_name` as array. If there is error, it returns **FALSE**

> **Note:** This function is experimental.

See also **pg_convert()**

pg_num_fields

(PHP 4 >= 4.2.0)

pg_num_fields -- Returns the number of fields

Description

int **pg_num_fields** (resource result)

pg_num_fields() returns the number of fields (columns) in a PostgreSQL *result*. The argument is a result resource returned by

pg_query(). This function will return -1 on error.

> **Note:** This function used to be called `pg_numfields()`.

See also **pg_num_rows()** and **pg_affected_rows()**.

pg_num_rows

(PHP 4 >= 4.2.0)

pg_num_rows -- Returns the number of rows

Description

int **pg_num_rows** (resource result)

pg_num_rows() will return the number of rows in a PostgreSQL *`result`* resource. *`result`* is a query result resource returned by **pg_query()**. This function will return -1 on error.

> **Note:** Use **pg_affected_rows()** to get number of rows affected by INSERT, UPDATE and DELETE query.

> **Note:** This function used to be called `pg_numrows()`.

See also **pg_num_fields()** and **pg_affected_rows()**.

pg_options

(PHP 3, PHP 4)

pg_options -- Get the options associated with the connection

Description

string **pg_options** (resource connection)

pg_options() will return a string containing the options specified on the given PostgreSQL *`connection`* resource.

pg_pconnect

(PHP 3, PHP 4)

pg_pconnect -- Open a persistent PostgreSQL connection

Description

resource **pg_pconnect** (string connection_string)

pg_pconnect() opens a connection to a PostgreSQL database. It returns a connection resource that is needed by other PostgreSQL functions.

For a description of the *`connection_string`* parameter, see **pg_connect()**.

To enable persistent connection, the pgsql.allow_persistent `php.ini` directive must be set to "On" (which is the default). The maximum number of persistent connection can be defined with the pgsql.max_persistent `php.ini` directive (defaults to -1 for no limit). The total number of connections can be set with the pgsql.max_links `php.ini` directive.

pg_close() will not close persistent links generated by **pg_pconnect()**.

See also **pg_connect()**, and the section Persistent Database Connections for more information.

pg_ping

(PHP 4 >= 4.3.0)

pg_ping -- Ping database connection

Description

array **pg_ping** (resource connection)

pg_ping() ping database connection, try to reconnect if it is broken. It returns **TRUE** if connection is alive, otherwise **FALSE**.

See also **pg_connection_status()** and **pg_connection_reset()**.

Example 1. PostgreSQL fetch array

```
<?php
$conn = pg_pconnect ("dbname=publisher");
if (!$conn) {
    echo "An error occured.\n";
    exit;
}

if (!pg_ping($conn))
    die("Connection is broken\n");
?>
```

pg_port

(PHP 3, PHP 4)

pg_port -- Return the port number associated with the connection

Description

int **pg_port** (resource connection)

pg_port() returns the port number that the given PostgreSQL *connection* resource is connected to.

pg_put_line

(PHP 4 >= 4.0.3)

pg_put_line -- Send a NULL-terminated string to PostgreSQL backend

Description

bool **pg_put_line** ([resource connection, string data])

pg_put_line() sends a NULL-terminated string to the PostgreSQL backend server. This is useful for example for very high-speed inserting of data into a table, initiated by starting a PostgreSQL copy-operation. That final NULL-character is added automatically. Returns **TRUE** on success or **FALSE** on failure.

> **Note:** The application must explicitly send the two characters "\." on the last line to indicate to the backend that it has finished sending its data.

See also **pg_end_copy()**.

Example 1. High-speed insertion of data into a table

```
<?php
    $conn = pg_pconnect ("dbname=foo");
    pg_query($conn, "create table bar (a int4, b char(16), d float8)");
    pg_query($conn, "copy bar from stdin");
    pg_put_line($conn, "3\thello world\t4.5\n");
    pg_put_line($conn, "4\tgoodbye world\t7.11\n");
    pg_put_line($conn, "\\.\n");
    pg_end_copy($conn);
?>
```

pg_query

(PHP 4 >= 4.2.0)

pg_query -- Execute a query

Description

resource **pg_query** (resource connection, string query)

pg_query() returns a query result resource if query could be executed. It returns **FALSE** on failure or if connection is not a valid connection. Details about the error can be retrieved using the **pg_last_error()** function if connection is valid. **pg_last_error()** sends an SQL statement to the PostgreSQL database specified by the *connection* resource. The *connection* must be a valid connection that was returned by **pg_connect()** or **pg_pconnect()**. The return value of this function is an query result resource to be used to access the results from other PostgreSQL functions such as **pg_fetch_array()**.

> **Note:** *connection* is a optional parameter for **pg_query()**. If *connection* is not set, default connection is used. Default connection is the last connection made by **pg_connect()** or **pg_pconnect()**.
>
> Although *connection* can be omitted, it is not recommended, since it could be a cause of hard to find bug in script.
>
> **Note:** This function used to be called `pg_exec()`. `pg_exec()` is still available for compatibility reasons but users are encouraged to use the newer name.

See also **pg_connect()**, **pg_pconnect()**, **pg_fetch_array()**, **pg_fetch_object()**, **pg_num_rows()**, and **pg_affected_rows()**.

pg_result_error

(PHP 4 >= 4.2.0)

pg_result_error -- Get error message associated with result

Description

string **pg_result_error** (resource result)

pg_result_error() returns error message associated with *result* resource. Therefore, user has better chance to get better error message than **pg_last_error()**.

See also **pg_query()**, **pg_send_query()**, **pg_get_result()**, **pg_last_error()** and **pg_last_notice()**

pg_result_seek

(PHP 4 >= 4.3.0)

pg_result_seek -- Set internal row offset in result resource

Description

array **pg_result_seek** (resource result, int offset)

pg_result_seek() set internal row offset in reuslt resource. It returns **FALSE**, if there is error.

See also **pg_fetch_row()**, **pg_fetch_assoc()**, **pg_fetch_array()**, **pg_fetch_object()** and **pg_fetch_result()**.

pg_result_status

(PHP 4 >= 4.2.0)

pg_result_status -- Get status of query result

Description

int **pg_result_status** (resource result)

pg_result_status() returns status of result resource. Possible retun values are PGSQL_EMPTY_QUERY, PGSQL_COMMAND_OK, PGSQL_TUPLES_OK, PGSQL_COPY_TO, PGSQL_COPY_FROM, PGSQL_BAD_RESPONSE, PGSQL_NONFATAL_ERROR and PGSQL_FATAL_ERROR.

See also **pg_connection_status()**.

pg_select

(PHP 4 >= 4.3.0)

pg_select -- Select records.

Description

array **pg_select** (resource connection, string table_name, array assoc_array [, int options])

pg_select() selects records specified by `assoc_array` which has `field=>value`. For successful query, it returns array contains all records and fields that match the condition specified by `assoc_array`. If `options` is specified, **pg_convert()** is applied to `assoc_array` with specified option.

Example 1. pg_select

```
<?php
    $db = pg_connect ('dbname=foo');
    // This is safe, since $_POST is converted automatically
    $rec = pg_select($db, 'post_log', $_POST);
    if ($rec) {
        echo "Records selected\n";
        var_dump($rec);
    }
    else {
        echo "User must have sent wrong inputs\n";
    }
?>
```

Note: This function is experimental.

See also **pg_convert()**

pg_send_query

(PHP 4 >= 4.2.0)

pg_send_query -- Send asynchronous query

Description

bool **pg_send_query** (resource connection, string query)

bool **pg_send_query** (string query)

pg_send_query() send asynchronous query to the *`connection`*. Unlike **pg_query()**, it can send multiple query to PostgreSQL and get the result one by one using **pg_get_result()**. Script execution is not blocked while query is executing. Use **pg_connection_busy()** to check connection is busy (i.e. query is executing). Query may be canceled by calling **pg_cancel_query()**.

Although user can send multiple query at once, user cannot send multiple query over busy connection. If query is sent while connection is busy, it waits until last query is finished and discards all result.

See also **pg_query()**, **pg_cancel_query()**, **pg_get_result()** and **pg_connection_busy()**

pg_set_client_encoding

(PHP 3 CVS only, PHP 4 >= 4.0.3)

pg_set_client_encoding -- Set the client encoding

Description

int **pg_set_client_encoding** ([resource connection, string encoding])

pg_set_client_encoding() sets the client encoding and returns 0 if success or -1 if error.

encoding is the client encoding and can be either : SQL_ASCII, EUC_JP, EUC_CN, EUC_KR, EUC_TW, UNICODE, MULE_INTERNAL, LATINX (X=1...9), KOI8, WIN, ALT, SJIS, BIG5, WIN1250. Available encoding depends on your PostgreSQL and libpq version. Refer to PostgreSQL manual for supported encodings for your PostgreSQL.

> **Note:** This function requires PHP-4.0.3 or higher and PostgreSQL-7.0 or higher. Supported encoding depends on PostgreSQL version. Refer to PostgreSQL manual for details.
>
> The function used to be called **pg_setclientencoding()**.

See also **pg_client_encoding()**.

pg_trace

(PHP 4 >= 4.0.1)

pg_trace -- Enable tracing a PostgreSQL connection

Description

bool **pg_trace** (string pathname [, string mode [, resource connection]])

pg_trace() enables tracing of the PostgreSQL frontend/backend communication to a debugging file specified as *pathname*. To fully understand the results, one needs to be familiar with the internals of PostgreSQL communication protocol. For those who are not, it can still be useful for tracing errors in queries sent to the server, you could do for example **grep '^To backend' trace.log** and see what query actually were sent to the PostgreSQL server. For more information, refer to PostgreSQL manual.

pathname and *mode* are the same as in **fopen()** (*mode* defaults to 'w'), *connection* specifies the connection to trace and defaults to the last one opened.

It returns **TRUE** if *pathname* could be opened for logging, **FALSE** otherwise.

See also **fopen()** and **pg_untrace()**.

pg_tty

(PHP 3, PHP 4)

pg_tty -- Return the tty name associated with the connection

Description

string **pg_tty** (resource connection)

pg_tty() returns the tty name that server side debugging output is sent to on the given PostgreSQL *connection* resource.

pg_unescape_bytea

(PHP 4 >= 4.3.0)

pg_unescape_bytea -- Escape binary for bytea type

Description

string **pg_unescape_bytea** (string data)

pg_unescape_bytea() unescapes string from bytea datatype. It returns unescaped string (binary).

> **Note:** When you SELECT bytea type, PostgreSQL returns octal byte value prefixed by \ (e.g. \032). Users are supposed to convert back to binary format by yourself.
>
> This function requires PostgreSQL 7.2 or later. With PostgreSQL 7.2.0 and 7.2.1, bytea type must be casted when you enable multi-byte support. i.e. `INSERT INTO test_table (image) VALUES ('$image_escaped'::bytea);` PostgreSQL 7.2.2 or later does not need cast. Exception is when client and backend character encoding does not match, there may be multi-byte stream error. User must cast to bytea to avoid this error.

See also **pg_escape_bytea()** and **pg_escape_string()**

pg_untrace

(PHP 4 >= 4.0.1)

pg_untrace -- Disable tracing of a PostgreSQL connection

Description

bool **pg_untrace** ([resource connection])

Stop tracing started by **pg_trace()**. *`connection`* specifies the connection that was traced and defaults to the last one opened.

Returns always **TRUE**.

See also **pg_trace()**.

pg_update

(PHP 4 >= 4.3.0)

pg_update -- Update table.

Description

long **pg_update** (resource connection, string table_name, array condition, array data [, int options])

pg_update() updates records that matches `condition` with `data`. If `options` is specified, **pg_convert()** is applied to `data` with specified options.

Example 1. pg_update

```
<?php
    $db = pg_connect ('dbname=foo');
    $data = array('field1'=>'AA', 'field2'=>'BB');
    // This is safe, since $_POST is converted automatically
    $res = pg_update($db, 'post_log', $_POST, $data);
    if ($res) {
        echo "Data is updated: $res\n";
    }
    else {
        echo "User must have sent wrong inputs\n";
    }
?>
```

> **Note:** This function is experimental.

See also **pg_convert()**

LXXXII. Process Control Functions

Introduction

Process Control support in PHP implements the Unix style of process creation, program execution, signal handling and process termination. Process Control should not be enabled within a webserver environment and unexpected results may happen if any Process Control functions are used within a webserver environment.

This documentation is intended to explain the general usage of each of the Process Control functions. For detailed information about Unix process control you are encouraged to consult your systems documentation including fork(2), waitpid(2) and signal(2) or a comprehensive reference such as Advanced Programming in the UNIX Environment by W. Richard Stevens (Addison-Wesley).

PCNTL now uses ticks as the signal handle callback mechanism, which is much faster than the previous mechanism. This change follows the same semantics as using "user ticks". You use the **declare()** statement to specify the locations in your program where callbacks are allowed to occur. This allows you to minimize the overhead of handling asynchronous events. In the past, compiling PHP with pcntl enabled would always incur this overhead, whether or not your script actually used pcntl.

There is one adjustment that all pcntl scripts prior to PHP 4.3.0 must make for them to work which is to either to use **declare()** on a section where you wish to allow callbacks or to just enable it across the entire script using the new global syntax of **declare()**.

> **Note:** This extension is not available on Windows platforms.

Requirements

No external libraries are needed to build this extension.

Installation

Process Control support in PHP is not enabled by default. You have to compile the CGI or CLI version of PHP with `--enable-pcntl` configuration option when compiling PHP to enable Process Control support.

> **Note:** Currently, this module will not function on non-Unix platforms (Windows).

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension has no resource types defined.

Predefined Constants

The following list of signals are supported by the Process Control functions. Please see your systems signal(7) man page for details of the default behavior of these signals.

`WNOHANG` (integer)

`WUNTRACED` (integer)

`SIG_IGN` (integer)

`SIG_DFL` (integer)

`SIG_ERR` (integer)

`SIGHUP` (integer)

SIGINT (integer)

SIGQUIT (integer)

SIGILL (integer)

SIGTRAP (integer)

SIGABRT (integer)

SIGIOT (integer)

SIGBUS (integer)

SIGFPE (integer)

SIGKILL (integer)

SIGUSR1 (integer)

SIGSEGV (integer)

SIGUSR2 (integer)

SIGPIPE (integer)

SIGALRM (integer)

SIGTERM (integer)

SIGSTKFLT (integer)

SIGCLD (integer)

SIGCHLD (integer)

SIGCONT (integer)

SIGSTOP (integer)

SIGTSTP (integer)

SIGTTIN (integer)

SIGTTOU (integer)

SIGURG (integer)

SIGXCPU (integer)

SIGXFSZ (integer)

SIGVTALRM (integer)

SIGPROF (integer)

SIGWINCH (integer)

SIGPOLL (integer)

SIGIO (integer)

SIGPWR (integer)

SIGSYS (integer)

SIGBABY (integer)

Examples

This example forks off a daemon process with a signal handler.

Example 1. Process Control Example

```
<?php
declare(ticks=1);

$pid = pcntl_fork();
if ($pid == -1) {
     die("could not fork");
} else if ($pid) {
     exit(); // we are the parent
} else {
     // we are the child
}

// detatch from the controlling terminal
if (!posix_setsid()) {
    die("could not detach from terminal");
}

// setup signal handlers
pcntl_signal(SIGTERM, "sig_handler");
pcntl_signal(SIGHUP, "sig_handler");

// loop forever performing tasks
while(1) {

    // do something interesting here

}

function sig_handler($signo) {

     switch($signo) {
         case SIGTERM:
             // handle shutdown tasks
             exit;
             break;
         case SIGHUP:
             // handle restart tasks
             break;
         default:
             // handle all other signals
     }

}

?>
```

See Also

A look at the section about POSIX functions may be useful.

Table of Contents

pcntl_exec

(PHP 4 >= 4.2.0)

pcntl_exec -- Executes specified program in current process space

Description

bool **pcntl_exec** (string path [, array args [, array envs]])

Warning
This function is currently not documented; only the argument list is available.

pcntl_fork

(PHP 4 >= 4.1.0)

pcntl_fork -- Forks the currently running process

Description

int **pcntl_fork** (void)

The **pcntl_fork()** function creates a child process that differs from the parent process only in it's PID and PPID. Please see your system's fork(2) man page for specific details as to how fork works on your system.

On success, the PID of the child process is returned in the parent's thread of execution, and a 0 is returned in the child's thread of execution. On failure, a -1 will be returned in the parent's context, no child process will be created, and a PHP error is raised.

Example 1. pcntl_fork() Example

```
<?php

$pid = pcntl_fork();
if ($pid == -1) {
     die("could not fork");
} else if ($pid) {
     // we are the parent
} else {
     // we are the child
}

?>
```

See also **pcntl_waitpid()** and **pcntl_signal()**.

pcntl_signal

(PHP 4 >= 4.1.0)

pcntl_signal -- Installs a signal handler

Description

bool **pcntl_signal** (int signo, mixed handle [, bool restart_syscalls])

The **pcntl_signal()** function installs a new signal handler for the signal indicated by *signo*. The signal handler is set to *handler* which may be the name of a user created function, or either of the two global constants SIG_IGN or SIG_DFL. The optional *restart_syscalls* specifies whether system call restarting should be used when this signal arrives and defaults to **TRUE**.

Returns **TRUE** on success or **FALSE** on failure.

> **Note:** The optional *restart_syscalls* parameter became available in PHP 4.3.0.
>
> **Note:** The ability to use an object method as a callback became available in PHP 4.3.0. Note that when you set a handler to an object method, that object's reference count is increased which makes it persist until you either change the handler to something else, or your script ends.

Example 1. pcntl_signal() Example

```
<?php
// tick use required as of PHP 4.3.0
declare (ticks = 1);

// signal handler function
function sig_handler($signo) {

     switch($signo) {
         case SIGTERM:
             // handle shutdown tasks
             exit;
             break;
         case SIGHUP:
             // handle restart tasks
```

```
            break;
        case SIGUSR1:
            print "Caught SIGUSR1...\n";
            break;
        default:
            // handle all other signals
    }

}

print "Installing signal handler...\n";

// setup signal handlers
pcntl_signal(SIGTERM, "sig_handler");
pcntl_signal(SIGHUP,  "sig_handler");
pcntl_signal(SIGUSR1, "sig_handler");

// or use an object, available as of PHP 4.3.0
// pcntl_signal(SIGUSR1, array($obj, "do_something");

print "Generating signal SIGTERM to self...\n";

// send SIGUSR1 to current process id
posix_kill(posix_getpid(), SIGUSR1);

print "Done\n"

?>
```

See also **pcntl_fork()** and **pcntl_waitpid()**.

pcntl_waitpid

(PHP 4 >= 4.1.0)

pcntl_waitpid -- Waits on or returns the status of a forked child

Description

int **pcntl_waitpid** (int pid, int status, int options)

The **pcntl_waitpid()** function suspends execution of the current process until a child as specified by the *pid* argument has exited, or until a signal is delivered whose action is to terminate the current process or to call a signal handling function. If a child as requested by *pid* has already exited by the time of the call (a so-called "zombie" process), the function returns immediately. Any system resources used by the child are freed. Please see your system's waitpid(2) man page for specific details as to how waitpid works on your system.

pcntl_waitpid() returns the process ID of the child which exited, -1 on error or zero if WNOHANG was used and no child was available

The value of *pid* can be one of the following:

Table 1. possible values for *pid*

< -1	wait for any child process whose process group ID is equal to the absolute value of *pid*.
-1	wait for any child process; this is the same behaviour that the wait function exhibits.
0	wait for any child process whose process group ID is equal to that of the calling process.
> 0	wait for the child whose process ID is equal to the value of *pid*.

pcntl_waitpid() will store status information in the *status* parameter which can be evaluated using the following functions: **pcntl_wifexited()**, **pcntl_wifstopped()**, **pcntl_wifsignaled()**, **pcntl_wexitstatus()**, **pcntl_wtermsig()** and **pcntl_wstopsig()**.

The value of *options* is the value of zero or more of the following two global constants OR'ed together:

Table 2. possible values for *options*

WNOHANG	return immediately if no child has exited.
WUNTRACED	return for children which are stopped, and whose status has not been reported.

See also **pcntl_fork()**, **pcntl_signal()**, **pcntl_wifexited()**, **pcntl_wifstopped()**, **pcntl_wifsignaled()**, **pcntl_wexitstatus()**, **pcntl_wtermsig()** and **pcntl_wstopsig()**.

pcntl_wexitstatus

(PHP 4 >= 4.1.0)

pcntl_wexitstatus -- Returns the return code of a terminated child

Description

int **pcntl_wexitstatus** (int status)

Returns the return code of a terminated child. This function is only useful if **pcntl_wifexited()** returned **TRUE**.

The parameter *status* is the status parameter supplied to a successfull call to **pcntl_waitpid()**.

See also **pcntl_waitpid()** and **pcntl_wifexited()**.

pcntl_wifexited

(PHP 4 >= 4.1.0)

pcntl_wifexited -- Returns **TRUE** if status code represents a successful exit

Description

int **pcntl_wifexited** (int status)

Returns **TRUE** if the child status code represents a successful exit.

The parameter *status* is the status parameter supplied to a successfull call to **pcntl_waitpid()**.

See also **pcntl_waitpid()** and **pcntl_wexitstatus()**.

pcntl_wifsignaled

(PHP 4 >= 4.1.0)

pcntl_wifsignaled -- Returns **TRUE** if status code represents a termination due to a signal

Description

int **pcntl_wifsignaled** (int status)

Returns **TRUE** if the child process exited because of a signal which was not caught.

The parameter *status* is the status parameter supplied to a successfull call to **pcntl_waitpid()**.

See also **pcntl_waitpid()** and **pcntl_signal()**.

pcntl_wifstopped

(PHP 4 >= 4.1.0)

pcntl_wifstopped -- Returns **TRUE** if child process is currently stopped

Description

int **pcntl_wifstopped** (int status)

Returns **TRUE** if the child process which caused the return is currently stopped; this is only possible if the call to **pcntl_waitpid()** was done using the option WUNTRACED.

The parameter *status* is the status parameter supplied to a successfull call to **pcntl_waitpid()**.

See also **pcntl_waitpid()**.

pcntl_wstopsig

(PHP 4 >= 4.1.0)

pcntl_wstopsig -- Returns the signal which caused the child to stop

Description

int **pcntl_wstopsig** (int status)

Returns the number of the signal which caused the child to stop. This function is only useful if **pcntl_wifstopped()** returned **TRUE**.

The parameter *status* is the status parameter supplied to a successfull call to **pcntl_waitpid()**.

See also **pcntl_waitpid()** and **pcntl_wifstopped()**.

pcntl_wtermsig

(PHP 4 >= 4.1.0)

pcntl_wtermsig -- Returns the signal which caused the child to terminate

Description

int **pcntl_wtermsig** (int status)

Returns the number of the signal that caused the child process to terminate. This function is only useful if **pcntl_wifsignaled()** returned **TRUE**.

The parameter *status* is the status parameter supplied to a successfull call to **pcntl_waitpid()**.

See also **pcntl_waitpid()**, **pcntl_signal()** and **pcntl_wifsignaled()**.

LXXXIII. Program Execution functions

Introduction

Those functions provides means to executes commands on the system itself, and means secure such commands.

Requirements

No external libraries are needed to build this extension.

Installation

There is no installation needed to use these functions; they are part of the PHP core.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

See Also

These functions are also closely related to the backtick operator. Also, while in Safe Mode you must consider the safe_mode_exec_dir directive.

Table of Contents

escapeshellarg

(PHP 4 >= 4.0.3)

escapeshellarg -- escape a string to be used as a shell argument

Description

string **escapeshellarg** (string arg)

escapeshellarg() adds single quotes around a string and quotes/escapes any existing single quotes allowing you to pass a string directly to a shell function and having it be treated as a single safe argument. This function should be used to escape individual arguments to shell functions coming from user input. The shell functions include **exec()**, **system()** and the backtick operator. A standard use would be:

```
system("ls ".escapeshellarg($dir));
```

See also **exec()**, **popen()**, **system()**, and the backtick operator.

escapeshellcmd

(PHP 3, PHP 4)

escapeshellcmd -- escape shell metacharacters

Description

string **escapeshellcmd** (string command)

escapeshellcmd() escapes any characters in a string that might be used to trick a shell command into executing arbitrary commands. This function should be used to make sure that any data coming from user input is escaped before this data is passed to the **exec()** or **system()** functions, or to the backtick operator. A standard use would be:

```
$e = escapeshellcmd($userinput);
```

```
system("echo $e"); // here we don't care if $e has spaces
$f = escapeshellcmd($filename);
system("touch \"/tmp/$f\"; ls -l \"/tmp/$f\""); // and here we do, so we use quotes
```

See also **escapeshellarg()**, **exec()**, **popen()**, **system()**, and the backtick operator.

exec

(PHP 3, PHP 4)

exec -- Execute an external program

Description

string **exec** (string command [, array output [, int return_var]])

exec() executes the given *command*, however it does not output anything. It simply returns the last line from the result of the command. If you need to execute a command and have all the data from the command passed directly back without any interference, use the **passthru()** function.

If the *array* argument is present, then the specified array will be filled with every line of output from the command. Line endings, such as `\n`, are not included in this array. Note that if the array already contains some elements, **exec()** will append to the end of the array. If you do not want the function to append elements, call **unset()** on the array before passing it to **exec()**.

If the *return_var* argument is present along with the *array* argument, then the return status of the executed command will be written to this variable.

Warning
If you are going to allow data coming from user input to be passed to this function, then you should be using **escapeshellarg()** or **escapeshellcmd()** to make sure that users cannot trick the system into executing arbitrary commands.

> **Note:** If you start a program using this function and want to leave it running in the background, you have to make sure that the output of that program is redirected to a file or some other output stream or else PHP will hang until the execution of the program ends.

See also **system()**, **passthru()**, **popen()**, **escapeshellcmd()**, and the backtick operator.

passthru

(PHP 3, PHP 4)

passthru -- Execute an external program and display raw output

Description

void **passthru** (string command [, int return_var])

The **passthru()** function is similar to the **exec()** function in that it executes a *command*. If the *return_var* argument is present, the return status of the Unix command will be placed here. This function should be used in place of **exec()** or **system()** when the output from the Unix command is binary data which needs to be passed directly back to the browser. A common use for this is to execute something like the pbmplus utilities that can output an image stream directly. By setting the Content-type to *image/gif* and then calling a pbmplus program to output a gif, you can create PHP scripts that output images directly.

Warning
If you are going to allow data coming from user input to be passed to this function, then you should be using **escapeshellarg()** or **escapeshellcmd()** to make sure that users cannot trick the system into executing arbitrary commands.

> **Note:** If you start a program using this function and want to leave it running in the background, you have to make sure that the output of that program is redirected to a file or some other output stream or else PHP will hang until the execution of the program ends.

See also **exec()**, **system()**, **popen()**, **escapeshellcmd()**, and the backtick operator.

proc_close

(PHP 4 >= 4.3.0)

proc_close -- Close a process opened by proc_open and return the exit code of that process.

Description

int **proc_close** (resource process)

proc_close() is similar to **pclose()** except that it only works on processes opened by **proc_open()**. **proc_close()** waits for the process to terminate, and returns it's exit code. If you have open pipes to that process, you should **fclose()** them prior to calling this function in order to avoid a deadlock - the child process may not be able to exit while the pipes are open.

proc_open

(PHP 4 >= 4.3.0)

proc_open -- Execute a command and open file pointers for input/output

Description

resource **proc_open** (string cmd, array descriptorspec, array pipes)

proc_open() is similar to **popen()** but provides a much greater degree of control over the program execution. *cmd* is the command to be executed by the shell. *descriptorspec* is an indexed array where the key represents the descriptor number and the value represents how PHP will pass that descriptor to the child process. *pipes* will be set to an indexed array of file pointers that correspond to PHP's end of any pipes that are created. The return value is a resource representing the process; you should free it using **proc_close()** when you are finished with it.

```
$descriptorspec = array(
   0 => array("pipe", "r"),  // stdin is a pipe that the child will read from
   1 => array("pipe", "w"),  // stdout is a pipe that the child will write to
   2 => array("file", "/tmp/error-output.txt", "a"), // stderr is a file to write to
);
$process = proc_open("php", $descriptorspec, $pipes);
if (is_resource($process)) {
    // $pipes now looks like this:
    // 0 => writeable handle connected to child stdin
    // 1 => readable handle connected to child stdout
    // Any error output will be appended to /tmp/error-output.txt

    fwrite($pipes[0], "<?php echo \"Hello World!\"; ?>");
    fclose($pipes[0]);

    while(!feof($pipes[1])) {
        echo fgets($pipes[1], 1024);
    }
    fclose($pipes[1]);
    // It is important that you close any pipes before calling
    // proc_close in order to avoid a deadlock
    $return_value = proc_close($process);

    echo "command returned $return_value\n";
}
```

The file descriptor numbers in *descriptorspec* are not limited to 0, 1 and 2 - you may specify any valid file descriptor number and it will be passed to the child process. This allows your script to interoperate with other scripts that run as "co-processes". In particular, this is useful for passing passphrases to programs like PGP, GPG and openssl in a more secure manner. It is also useful for reading status information provided by those programs on auxillary file descriptors.

> **Note:** Windows compatibility: Descriptors beyond 2 (stderr) are made available to the child process as inheritable handles, but since the Windows architecture does not associate file descriptor numbers with low-level handles, the child process does not (yet) have a means of accessing those handles. Stdin, stdout and stderr work as expected.
>
> **Note:** This function was introduced in PHP 4.3.0.
>
> **Note:** If you only need a uni-directional (one-way) process pipe, use **popen()** instead, as it is much easier to use.

See also **exec()**, **system()**, **passthru()**, **popen()**, **escapeshellcmd()**, and the backtick operator.

shell_exec

(PHP 4)

shell_exec -- Execute command via shell and return complete output as string

Description

string **shell_exec** (string cmd)

This function is identical to the backtick operator.

system

(PHP 3, PHP 4)

system -- Execute an external program and display output

Description

string **system** (string command [, int return_var])

system() is just like the C version of the function in that it executes the given *command* and outputs the result. If a variable is provided as the second argument, then the return status code of the executed command will be written to this variable.

Warning
If you are going to allow data coming from user input to be passed to this function, then you should be using **escapeshellarg()** or **escapeshellcmd()** to make sure that users cannot trick the system into executing arbitrary commands.

> **Note:** If you start a program using this function and want to leave it running in the background, you have to make sure that the output of that program is redirected to a file or some other output stream or else PHP will hang until the execution of the program ends.

The **system()** call also tries to automatically flush the web server's output buffer after each line of output if PHP is running as a server module.

Returns the last line of the command output on success, and **FALSE** on failure.

If you need to execute a command and have all the data from the command passed directly back without any interference, use the **passthru()** function.

See also **exec()**, **passthru()**, **popen()**, **escapeshellcmd()**, and the backtick operator.

LXXXIV. Printer functions

Introduction

These functions are only available under Windows 9.x, ME, NT4 and 2000. They have been added in PHP 4.0.4.

Installation

Add the line `extension=php_printer.dll` to your `php.ini` file.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. Printer configuration options

Name	Default	Changeable
printer.default_printer	""	PHP_INI_ALL

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Table of Contents

printer_abort

(no version information, might be only in CVS)

printer_abort -- Deletes the printer's spool file

Description

void **printer_abort** (resource handle)

This function deletes the printers spool file.

handle must be a valid handle to a printer.

Example 1. printer_abort() example

```
$handle = printer_open();
printer_abort($handle);
printer_close($handle);
```

printer_close

(no version information, might be only in CVS)

printer_close -- Close an open printer connection

Description

void **printer_close** (resource handle)

This function closes the printer connection. **printer_close()** also closes the active device context.

handle must be a valid handle to a printer.

Example 1. printer_close() example

```
$handle = printer_open();
printer_close($handle);
```

printer_create_brush

(no version information, might be only in CVS)

printer_create_brush -- Create a new brush

Description

mixed **printer_create_brush** (int style, string color)

The function creates a new brush and returns a handle to it. A brush is used to fill shapes. For an example see **printer_select_brush()**. *color* must be a color in RGB hex format, i.e. "oooooo" for black, *style* must be one of the following constants:

- *PRINTER_BRUSH_SOLID*: creates a brush with a solid color.
- *PRINTER_BRUSH_DIAGONAL*: creates a brush with a 45-degree upward left-to-right hatch (/).
- *PRINTER_BRUSH_CROSS*: creates a brush with a cross hatch (+).
- *PRINTER_BRUSH_DIAGCROSS*: creates a brush with a 45 cross hatch (x).
- *PRINTER_BRUSH_FDIAGONAL*: creates a brush with a 45-degree downward left-to-right hatch (\ ).
- *PRINTER_BRUSH_HORIZONTAL*: creates a brush with a horizontal hatch (-).
- *PRINTER_BRUSH_VERTICAL*: creates a brush with a vertical hatch (|).
- *PRINTER_BRUSH_CUSTOM*: creates a custom brush from an BMP file. The second parameter is used to specify the BMP instead of the RGB color code.

printer_create_dc

(no version information, might be only in CVS)

printer_create_dc -- Create a new device context

Description

void **printer_create_dc** (resource handle)

The function creates a new device context. A device context is used to customize the graphic objects of the document. *handle* must be a valid handle to a printer.

Example 1. printer_create_dc() example

```
$handle = printer_open();
printer_start_doc($handle);
printer_start_page($handle);

printer_create_dc($handle);
/* do some stuff with the dc */
printer_set_option($handle, PRINTER_TEXT_COLOR, "333333");
printer_draw_text($handle, 1, 1, "text");
printer_delete_dc($handle);
```

```
/* create another dc */
printer_create_dc($handle);
printer_set_option($handle, PRINTER_TEXT_COLOR, "000000");
printer_draw_text($handle, 1, 1, "text");
/* do some stuff with the dc */

printer_delete_dc($handle);

printer_endpage($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_create_font

(no version information, might be only in CVS)

printer_create_font -- Create a new font

Description

mixed **printer_create_font** (string face, int height, int width, int font_weight, bool italic, bool underline, bool strikeout, int orientaton)

The function creates a new font and returns a handle to it. A font is used to draw text. For an example see **printer_select_font()**. *face* must be a string specifying the font face. *height* specifies the font height, and *width* the font width. The *font_weight* specifies the font weight (400 is normal), and can be one of the following predefined constants.

- *PRINTER_FW_THIN*: sets the font weight to thin (100).
- *PRINTER_FW_ULTRALIGHT*: sets the font weight to ultra light (200).
- *PRINTER_FW_LIGHT*: sets the font weight to light (300).
- *PRINTER_FW_NORMAL*: sets the font weight to normal (400).
- *PRINTER_FW_MEDIUM*: sets the font weight to medium (500).
- *PRINTER_FW_BOLD*: sets the font weight to bold (700).
- *PRINTER_FW_ULTRABOLD*: sets the font weight to ultra bold (800).
- *PRINTER_FW_HEAVY*: sets the font weight to heavy (900).

italic can be **TRUE** or **FALSE**, and sets whether the font should be italic.

underline can be **TRUE** or **FALSE**, and sets whether the font should be underlined.

strikeout can be **TRUE** or **FALSE**, and sets whether the font should be striked out.

orientation specifies a rotation. For an example see **printer_select_font()**.

printer_create_pen

(no version information, might be only in CVS)

printer_create_pen -- Create a new pen

Description

mixed **printer_create_pen** (int style, int width, string color)

The function creates a new pen and returns a handle to it. A pen is used to draw lines and curves. For an example see **printer_select_pen()**. *color* must be a color in RGB hex format, i.e. "oooooo" for black, *width* specifies the width of the pen whereas *style* must be one of the following constants:

- *PRINTER_PEN_SOLID*: creates a solid pen.
- *PRINTER_PEN_DASH*: creates a dashed pen.

- *PRINTER_PEN_DOT*: creates a dotted pen.
- *PRINTER_PEN_DASHDOT*: creates a pen with dashes and dots.
- *PRINTER_PEN_DASHDOTDOT*: creates a pen with dashes and double dots.
- *PRINTER_PEN_INVISIBLE*: creates an invisible pen.

printer_delete_brush

(no version information, might be only in CVS)

printer_delete_brush -- Delete a brush

Description

bool **printer_delete_brush** (resource handle)

The function deletes the selected brush. For an example see **printer_select_brush()**. It returns **TRUE** on success, or **FALSE** otherwise. *handle* must be a valid handle to a brush.

printer_delete_dc

(no version information, might be only in CVS)

printer_delete_dc -- Delete a device context

Description

bool **printer_delete_dc** (resource handle)

The function deletes the device context and returns **TRUE** on success, or **FALSE** if an error occurred. For an example see **printer_create_dc()**. *handle* must be a valid handle to a printer.

printer_delete_font

(no version information, might be only in CVS)

printer_delete_font -- Delete a font

Description

bool **printer_delete_font** (resource handle)

The function deletes the selected font. For an example see **printer_select_font()**. It returns **TRUE** on success, or **FALSE** otherwise. *handle* must be a valid handle to a font.

printer_delete_pen

(no version information, might be only in CVS)

printer_delete_pen -- Delete a pen

Description

bool **printer_delete_pen** (resource handle)

The function deletes the selected pen. For an example see **printer_select_pen()**. It returns **TRUE** on success, or **FALSE** otherwise. *handle* must be a valid handle to a pen.

printer_draw_bmp

(no version information, might be only in CVS)

printer_draw_bmp -- Draw a bmp

Description

void **printer_draw_bmp** (resource handle, string filename, int x, int y)

The function simply draws an bmp the bitmap *filename* at position *x*, *y*. *handle* must be a valid handle to a printer.

The function returns **TRUE** on success, or otherwise **FALSE**.

Example 1. printer_draw_bmp() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

printer_draw_bmp($handle, "c:\\image.bmp", 1, 1);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_chord

(no version information, might be only in CVS)

printer_draw_chord -- Draw a chord

Description

void **printer_draw_chord** (resource handle, int rec_x, int rec_y, int rec_x1, int rec_y1, int rad_x, int rad_y, int rad_x1, int rad_y1)

The function simply draws an chord. *handle* must be a valid handle to a printer.

rec_x is the upper left x coordinate of the bounding rectangle.

rec_y is the upper left y coordinate of the bounding rectangle.

rec_x1 is the lower right x coordinate of the bounding rectangle.

rec_y1 is the lower right y coordinate of the bounding rectangle.

rad_x is x coordinate of the radial defining the beginning of the chord.

rad_y is y coordinate of the radial defining the beginning of the chord.

rad_x1 is x coordinate of the radial defining the end of the chord.

rad_y1 is y coordinate of the radial defining the end of the chord.

Example 1. printer_draw_chord() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_chord($handle, 1, 1, 500, 500, 1, 1, 500, 1);

printer_delete_brush($brush);
printer_delete_pen($pen);
```

```
printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_elipse

(no version information, might be only in CVS)

printer_draw_elipse -- Draw an ellipse

Description

void **printer_draw_elipse** (resource handle, int ul_x, int ul_y, int lr_x, int lr_y)

The function simply draws an ellipse. *handle* must be a valid handle to a printer.

ul_x is the upper left x coordinate of the ellipse.

ul_y is the upper left y coordinate of the ellipse.

lr_x is the lower right x coordinate of the ellipse.

lr_y is the lower right y coordinate of the ellipse.

Example 1. printer_draw_elipse() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_elipse($handle, 1, 1, 500, 500);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_line

(no version information, might be only in CVS)

printer_draw_line -- Draw a line

Description

void **printer_draw_line** (resource printer_handle, int from_x, int from_y, int to_x, int to_y)

The function simply draws a line from position *from_x*, *from_y* to position *to_x*, *to_y* using the selected pen. *printer_handle* must be a valid handle to a printer.

Example 1. printer_draw_line() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 30, "000000");
printer_select_pen($handle, $pen);

printer_draw_line($handle, 1, 10, 1000, 10);
printer_draw_line($handle, 1, 60, 500, 60);

printer_delete_pen($pen);
```

```
printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_pie

(no version information, might be only in CVS)

printer_draw_pie -- Draw a pie

Description

void **printer_draw_pie** (resource handle, int rec_x, int rec_y, int rec_x1, int rec_y1, int rad1_x, int rad1_y, int rad2_x, int rad2_y)

The function simply draws an pie. *handle* must be a valid handle to a printer.

rec_x is the upper left x coordinate of the bounding rectangle.

rec_y is the upper left y coordinate of the bounding rectangle.

rec_x1 is the lower right x coordinate of the bounding rectangle.

rec_y1 is the lower right y coordinate of the bounding rectangle.

rad1_x is x coordinate of the first radial's ending.

rad1_y is y coordinate of the first radial's ending.

rad2_x is x coordinate of the second radial's ending.

rad2_y is y coordinate of the second radial's ending.

Example 1. printer_draw_pie() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_pie($handle, 1, 1, 500, 500, 1, 1, 500, 1);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_rectangle

(no version information, might be only in CVS)

printer_draw_rectangle -- Draw a rectangle

Description

void **printer_draw_rectangle** (resource handle, int ul_x, int ul_y, int lr_x, int lr_y)

The function simply draws a rectangle.

handle must be a valid handle to a printer.

ul_x is the upper left x coordinate of the rectangle.

ul_y is the upper left y coordinate of the rectangle.

lr_x is the lower right x coordinate of the rectangle.

lr_y is the lower right y coordinate of the rectangle.

Example 1. printer_draw_rectangle() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_rectangle($handle, 1, 1, 500, 500);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_roundrect

(no version information, might be only in CVS)

printer_draw_roundrect -- Draw a rectangle with rounded corners

Description

void **printer_draw_roundrect** (resource handle, int ul_x, int ul_y, int lr_x, int lr_y, int width, int height)

The function simply draws a rectangle with rounded corners.

handle must be a valid handle to a printer.

ul_x is the upper left x coordinate of the rectangle.

ul_y is the upper left y coordinate of the rectangle.

lr_x is the lower right x coordinate of the rectangle.

lr_y is the lower right y coordinate of the rectangle.

width is the width of the ellipse.

height is the height of the ellipse.

Example 1. printer_draw_roundrect() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_roundrect($handle, 1, 1, 500, 500, 200, 200);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_text

(no version information, might be only in CVS)

printer_draw_text -- Draw text

Description

void **printer_draw_text** (resource printer_handle, string text, int x, int y)

The function simply draws *text* at position *x*, *y* using the selected font. *printer_handle* must be a valid handle to a printer.

Example 1. printer_draw_text() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$font = printer_create_font("Arial",72,48,400,false,false,false,0);
printer_select_font($handle, $font);
printer_draw_text($handle, "test", 10, 10);
printer_delete_font($font);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_end_doc

(no version information, might be only in CVS)

printer_end_doc -- Close document

Description

bool **printer_end_doc** (resource handle)

Closes a new document in the printer spooler. The document is now ready for printing. For an example see **printer_start_doc()**. *handle* must be a valid handle to a printer.

printer_end_page

(no version information, might be only in CVS)

printer_end_page -- Close active page

Description

bool **printer_end_page** (resource handle)

The function closes the active page in the active document. For an example see **printer_start_doc()**. *handle* must be a valid handle to a printer.

printer_get_option

(no version information, might be only in CVS)

printer_get_option -- Retrieve printer configuration data

Description

mixed **printer_get_option** (resource handle, string option)

The function retrieves the configuration setting of *option*. *handle* must be a valid handle to a printer. Take a look at **printer_set_option()** for the settings that can be retrieved, additionally the following settings can be retrieved:

- *PRINTER_DEVICENAME* returns the devicename of the printer.
- *PRINTER_DRIVERVERSION* returns the printer driver version.

Example 1. printer_get_option() example

```
$handle = printer_open();
print printer_get_option($handle, PRINTER_DRIVERVERSION);
printer_close($handle);
```

printer_list

(no version information, might be only in CVS)

printer_list -- Return an array of printers attached to the server

Description

array **printer_list** (int enumtype [, string name [, int level]])

The function enumerates available printers and their capabilities. *level* sets the level of information request. Can be 1,2,4 or 5. *enumtype* must be one of the following predefined constants:

- *PRINTER_ENUM_LOCAL*: enumerates the locally installed printers.
- *PRINTER_ENUM_NAME*: enumerates the printer of *name*, can be a server, domain or print provider.
- *PRINTER_ENUM_SHARED*: this parameter can't be used alone, it has to be OR'ed with other parameters, i.e. PRINTER_ENUM_LOCAL to detect the locally shared printers.
- *PRINTER_ENUM_DEFAULT*: (Wing.x only) enumerates the default printer.
- *PRINTER_ENUM_CONNECTIONS*: (WinNT/2000 only) enumerates the printers to which the user has made connections.
- *PRINTER_ENUM_NETWORK*: (WinNT/2000 only) enumerates network printers in the computer's domain. Only valid if *level* is 1.
- *PRINTER_ENUM_REMOTE*: (WinNT/2000 only) enumerates network printers and print servers in the computer's domain. Only valid if *level* is 1.

Example 1. printer_list() example

```
/* detect locally shared printer */
var_dump( printer_list(PRINTER_ENUM_LOCAL | PRINTER_ENUM_SHARED) );
```

printer_logical_fontheight

(no version information, might be only in CVS)

printer_logical_fontheight -- Get logical font height

Description

int **printer_logical_fontheight** (resource handle, int height)

The function calculates the logical font height of *height*. *handle* must be a valid handle to a printer.

Example 1. printer_logical_fontheight() example

```
$handle = printer_open();
print printer_logical_fontheight($handle, 72);
printer_close($handle);
```

printer_open

(no version information, might be only in CVS)

printer_open -- Open connection to a printer

Description

mixed **printer_open** ([string devicename])

This function tries to open a connection to the printer *devicename*, and returns a handle on success or **FALSE** on failure.

If no parameter was given it tries to open a connection to the default printer (if not specified in `php.ini` as `printer.default_printer`, php tries to detect it).

printer_open() also starts a device context.

Example 1. printer_open() example

```
$handle = printer_open("HP Deskjet 930c");
$handle = printer_open();
```

printer_select_brush

(no version information, might be only in CVS)

printer_select_brush -- Select a brush

Description

void **printer_select_brush** (resource printer_handle, resource brush_handle)

The function selects a brush as the active drawing object of the actual device context. A brush is used to fill shapes. If you draw an rectangle the brush is used to draw the shapes, while the pen is used to draw the border. If you haven't selected a brush before drawing shapes, the shape won't be filled. *printer_handle* must be a valid handle to a printer. *brush_handle* must be a valid handle to a brush.

Example 1. printer_select_brush() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);
$brush = printer_create_brush(PRINTER_BRUSH_CUSTOM, "c:\\brush.bmp");
printer_select_brush($handle, $brush);

printer_draw_rectangle($handle, 1,1,500,500);

printer_delete_brush($brush);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "000000");
printer_select_brush($handle, $brush);
printer_draw_rectangle($handle, 1,501,500,1001);
printer_delete_brush($brush);

printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_select_font

(no version information, might be only in CVS)

printer_select_font -- Select a font

Description

void **printer_select_font** (resource printer_handle, resource font_handle)

The function selects a font to draw text. *printer_handle* must be a valid handle to a printer. *font_handle* must be a valid handle to a font.

Example 1. printer_select_font() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$font = printer_create_font("Arial", 148, 76, PRINTER_FW_MEDIUM, false, false, false, -50);
printer_select_font($handle, $font);
printer_draw_text($handle, "PHP is simply cool", 40, 40);
printer_delete_font($font);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_select_pen

(no version information, might be only in CVS)

printer_select_pen -- Select a pen

Description

void **printer_select_pen** (resource printer_handle, resource pen_handle)

The function selects a pen as the active drawing object of the actual device context. A pen is used to draw lines and curves. I.e. if you draw a single line the pen is used. If you draw an rectangle the pen is used to draw the borders, while the brush is used to fill the shape. If you haven't selected a pen before drawing shapes, the shape won't be outlined. *printer_handle* must be a valid handle to a printer. *pen_handle* must be a valid handle to a pen.

Example 1. printer_select_pen() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 30, "2222FF");
printer_select_pen($handle, $pen);

printer_draw_line($handle, 1, 60, 500, 60);

printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_set_option

(no version information, might be only in CVS)

printer_set_option -- Configure the printer connection

Description

bool **printer_set_option** (resource handle, int option, mixed value)

The function sets the following options for the current connection. *handle* must be a valid handle to a printer. For *option* can be one of the following constants:

- *PRINTER_COPIES*: sets how many copies should be printed, *value* must be an integer.
- *PRINTER_MODE*: specifies the type of data (text, raw or emf), *value* must be a string.

- *PRINTER_TITLE*: specifies the name of the document, *value* must be a string.
- *PRINTER_ORIENTATION*: specifies the orientation of the paper, *value* can be either PRINTER_ORIENTATION_PORTRAIT or PRINTER_ORIENTATION_LANDSCAPE
- *PRINTER_RESOLUTION_Y*: specifies the y-resolution in DPI, *value* must be an integer.
- *PRINTER_RESOLUTION_X*: specifies the x-resolution in DPI, *value* must be an integer.
- *PRINTER_PAPER_FORMAT*: specifies the a predefined paper format, set *value* to PRINTER_FORMAT_CUSTOM if you want to specify a custom format with PRINTER_PAPER_WIDTH and PRINTER_PAPER_LENGTH. *value* can be one of the following constants.
  - *PRINTER_FORMAT_CUSTOM*: let's you specify a custom paper format.
  - *PRINTER_FORMAT_LETTER*: specifies standard letter format (8 1/2- by 11-inches).
  - *PRINTER_FORMAT_LETTER*: specifies standard legal format (8 1/2- by 14-inches).
  - *PRINTER_FORMAT_A3*: specifies standard A3 format (297- by 420-millimeters).
  - *PRINTER_FORMAT_A4*: specifies standard A4 format (210- by 297-millimeters).
  - *PRINTER_FORMAT_A5*: specifies standard A5 format (148- by 210-millimeters).
  - *PRINTER_FORMAT_B4*: specifies standard B4 format (250- by 354-millimeters).
  - *PRINTER_FORMAT_B5*: specifies standard B5 format (182- by 257-millimeter).
  - *PRINTER_FORMAT_FOLIO*: specifies standard FOLIO format (8 1/2- by 13-inch).
- *PRINTER_PAPER_LENGTH*: if PRINTER_PAPER_FORMAT is set to PRINTER_FORMAT_CUSTOM, PRINTER_PAPER_LENGTH specifies a custom paper length in mm, *value* must be an integer.
- *PRINTER_PAPER_WIDTH*: if PRINTER_PAPER_FORMAT is set to PRINTER_FORMAT_CUSTOM, PRINTER_PAPER_WIDTH specifies a custom paper width in mm, *value* must be an integer.
- *PRINTER_SCALE*: specifies the factor by which the printed output is to be scaled. the page size is scaled from the physical page size by a factor of scale/100. for example if you set the scale to 50, the output would be half of it's original size. *value* must be an integer.
- *PRINTER_BACKGROUND_COLOR*: specifies the background color for the actual device context, *value* must be a string containing the rgb information in hex format i.e. "005533".
- *PRINTER_TEXT_COLOR*: specifies the text color for the actual device context, *value* must be a string containing the rgb information in hex format i.e. "005533".
- *PRINTER_TEXT_ALIGN*: specifies the text alignment for the actual device context, *value* can be combined through OR'ing the following constants:
  - *PRINTER_TA_BASELINE*: text will be aligned at the base line.
  - *PRINTER_TA_BOTTOM*: text will be aligned at the bottom.
  - *PRINTER_TA_TOP*: text will be aligned at the top.
  - *PRINTER_TA_CENTER*: text will be aligned at the center.
  - *PRINTER_TA_LEFT*: text will be aligned at the left.
  - *PRINTER_TA_RIGHT*: text will be aligned at the right.

Example 1. printer_set_option() example

```
$handle = printer_open();
printer_set_option($handle, PRINTER_SCALE, 75);
printer_set_option($handle, PRINTER_TEXT_ALIGN, PRINTER_TA_LEFT);
printer_close($handle);
```

printer_start_doc

(no version information, might be only in CVS)

printer_start_doc -- Start a new document

Description

bool **printer_start_doc** (resource handle [, string document])

The function creates a new document in the printer spooler. A document can contain multiple pages, it's used to schedule the print job in the spooler. *handle* must be a valid handle to a printer. The optional parameter *document* can be used to set an alternative document name.

Example 1. printer_start_doc() example

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_start_page

(no version information, might be only in CVS)

printer_start_page -- Start a new page

Description

bool **printer_start_page** (resource handle)

The function creates a new page in the active document. For an example see **printer_start_doc()**. *handle* must be a valid handle to a printer.

printer_write

(no version information, might be only in CVS)

printer_write -- Write data to the printer

Description

bool **printer_write** (resource handle, string content)

Writes *content* directly to the printer, and returns **TRUE** on success or **FALSE** if it failed.

handle must be a valid handle to a printer.

Example 1. printer_write() example

```
$handle = printer_open();
printer_write($handle, "Text to print");
printer_close($handle);
```

LXXXV. Pspell Functions

Introduction

These functions allow you to check the spelling of a word and offer suggestions.

> **Note:** This extension is not available on Windows platforms.

Requirements

To compile PHP with pspell support, you need the aspell and pspell libraries, available from http://aspell.sourceforge.net/ and http://aspell.net/ respectively.

Installation

If you have the libraries needed add the `--with-pspell[=dir]` option when compiling PHP.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

PSPELL_FAST (integer)

PSPELL_NORMAL (integer)

PSPELL_BAD_SPELLERS (integer)

PSPELL_RUN_TOGETHER (integer)

Table of Contents

pspell_add_to_personal

(PHP 4 >= 4.0.2)

pspell_add_to_personal -- Add the word to a personal wordlist

Description

int **pspell_add_to_personal** (int dictionary_link, string word)

pspell_add_to_personal() adds a word to the personal wordlist. If you used **pspell_new_config()** with **pspell_config_personal()** to open the dictionary, you can save the wordlist later with **pspell_save_wordlist()**. Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Example 1. pspell_add_to_personal()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
$pspell_link = pspell_new_config ($pspell_config);

pspell_add_to_personal ($pspell_link, "Vlad");
pspell_save_wordlist ($pspell_link);
```

pspell_add_to_session

(PHP 4 >= 4.0.2)

pspell_add_to_session -- Add the word to the wordlist in the current session

Description

int **pspell_add_to_session** (int dictionary_link, string word)

pspell_add_to_session() adds a word to the wordlist associated with the current session. It is very similar to **pspell_add_to_personal()**

pspell_check

(PHP 4 >= 4.0.2)

pspell_check -- Check a word

Description

bool **pspell_check** (int dictionary_link, string word)

pspell_check() checks the spelling of a word and returns **TRUE** if the spelling is correct, **FALSE** if not.

Example 1. pspell_check()

```
$pspell_link = pspell_new ("en");

if (pspell_check ($pspell_link, "testt")) {
    echo "This is a valid spelling";
} else {
    echo "Sorry, wrong spelling";
}
```

pspell_clear_session

(PHP 4 >= 4.0.2)

pspell_clear_session -- Clear the current session

Description

int **pspell_clear_session** (int dictionary_link)

pspell_clear_session() clears the current session. The current wordlist becomes blank, and, for example, if you try to save it with **pspell_save_wordlist()**, nothing happens.

Example 1. **pspell_add_to_personal()**

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
$pspell_link = pspell_new_config ($pspell_config);

pspell_add_to_personal ($pspell_link, "Vlad");
pspell_clear_session ($pspell_link);
pspell_save_wordlist ($pspell_link);    //"Vlad" will not be saved
```

pspell_config_create

(PHP 4 >= 4.0.2)

pspell_config_create -- Create a config used to open a dictionary

Description

int **pspell_config_create** (string language [, string spelling [, string jargon [, string encoding]]])

pspell_config_create() has a very similar syntax to **pspell_new()**. In fact, using **pspell_config_create()** immediatelly followed by **pspell_new_config()** will produce the exact same result. However, after creating a new config, you can also use **pspell_config_*()** functions before calling **pspell_new_config()** to take advantage of some advanced functionality.

The language parameter is the language code which consists of the two letter ISO 639 language code and an optional two letter ISO 3166 country code after a dash or underscore.

The spelling parameter is the requested spelling for languages with more than one spelling such as English. Known values are 'american', 'british', and 'canadian'.

The jargon parameter contains extra information to distinguish two different words lists that have the same language and spelling parameters.

The encoding parameter is the encoding that words are expected to be in. Valid values are 'utf-8', 'iso8859-*', 'koi8-r', 'viscii', 'cp1252', 'machine unsigned 16', 'machine unsigned 32'. This parameter is largely untested, so be careful when using.

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- **PSPELL_FAST** - Fast mode (least number of suggestions)
- **PSPELL_NORMAL** - Normal mode (more suggestions)
- **PSPELL_BAD_SPELLERS** - Slow mode (a lot of suggestions)

For more information and examples, check out inline manual pspell website:http://aspell.net/.

Example 1. **pspell_config_create()**

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl ($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_personal ($pspell_config, "en");
```

pspell_config_ignore

(PHP 4 >= 4.0.2)

pspell_config_ignore -- Ignore words less than N characters long

Description

int **pspell_config_ignore** (int dictionary_link, int n)

pspell_config_ignore() should be used on a config before calling **pspell_new_config()**. This function allows short words to be

skipped by the spellchecker. Words less then n characters will be skipped.

Example 1. pspell_config_ignore()

```
$pspell_config = pspell_config_create ("en");
pspell_config_ignore($pspell_config, 5);
$pspell_link = pspell_new_config($pspell_config);
pspell_check($pspell_link, "abcd");    //will not result in an error
```

pspell_config_mode

(PHP 4 >= 4.0.2)

pspell_config_mode -- Change the mode number of suggestions returned

Description

int **pspell_config_mode** (int dictionary_link, int mode)

pspell_config_mode() should be used on a config before calling **pspell_new_config()**. This function determines how many suggestions will be returned by **pspell_suggest()**.

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- **PSPELL_FAST** - Fast mode (least number of suggestions)
- **PSPELL_NORMAL** - Normal mode (more suggestions)
- **PSPELL_BAD_SPELLERS** - Slow mode (a lot of suggestions)

Example 1. pspell_config_mode()

```
$pspell_config = pspell_config_create ("en");
pspell_config_mode($pspell_config, PSPELL_FAST);
$pspell_link = pspell_new_config($pspell_config);
pspell_check($pspell_link, "thecat");
```

pspell_config_personal

(PHP 4 >= 4.0.2)

pspell_config_personal -- Set a file that contains personal wordlist

Description

int **pspell_config_personal** (int dictionary_link, string file)

pspell_config_personal() should be used on a config before calling **pspell_new_config()**. The personal wordlist will be loaded and used in addition to the standard one after you call **pspell_new_config()**. If the file does not exist, it will be created. The file is also the file where **pspell_save_wordlist()** will save personal wordlist to. The file should be writable by whoever php runs as (e.g. nobody). Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Example 1. pspell_config_personal()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
$pspell_link = pspell_new_config ($pspell_config);
pspell_check ($pspell_link, "thecat");
```

pspell_config_repl

(PHP 4 >= 4.0.2)

pspell_config_repl -- Set a file that contains replacement pairs

Description

int **pspell_config_repl** (int dictionary_link, string file)

pspell_config_repl() should be used on a config before calling **pspell_new_config()**. The replacement pairs improve the quality of the spellchecker. When a word is misspelled, and a proper suggestion was not found in the list, **pspell_store_replacement()** can be used to store a replacement pair and then **pspell_save_wordlist()** to save the wordlist along with the replacement pairs. The file should be writable by whoever php runs as (e.g. nobody). Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Example 1. pspell_config_repl()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl ($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_config ($pspell_config);
pspell_check ($pspell_link, "thecat");
```

pspell_config_runtogether

(PHP 4 >= 4.0.2)

pspell_config_runtogether -- Consider run-together words as valid compounds

Description

int **pspell_config_runtogether** (int dictionary_link, bool flag)

pspell_config_runtogether() should be used on a config before calling **pspell_new_config()**. This function determines whether run-together words will be treated as legal compounds. That is, "thecat" will be a legal compound, athough there should be a space between the two words. Changing this setting only affects the results returned by **pspell_check()**; **pspell_suggest()** will still return suggestions.

Example 1. pspell_config_runtogether()

```
$pspell_config = pspell_config_create ("en");
pspell_config_runtogether ($pspell_config, true);
$pspell_link = pspell_new_config ($pspell_config);
pspell_check ($pspell_link, "thecat");
```

pspell_config_save_repl

(PHP 4 >= 4.0.2)

pspell_config_save_repl -- Determine whether to save a replacement pairs list along with the wordlist

Description

int **pspell_config_save_repl** (int dictionary_link, bool flag)

pspell_config_save_repl() should be used on a config before calling **pspell_new_config()**. It determines whether **pspell_save_wordlist()** will save the replacement pairs along with the wordlist. Usually there is no need to use this function because if **pspell_config_repl()** is used, the replacement pairs will be saved by **pspell_save_wordlist()** anyway, and if it is not, the replacement pairs will not be saved. Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

pspell_new_config

(PHP 4 >= 4.0.2)

pspell_new_config -- Load a new dictionary with settings based on a given config

Description

int **pspell_new_config** (int config)

pspell_new_config() opens up a new dictionary with settings specified in a config, created with with **pspell_config_create()** and modified with **pspell_config_*()** functions. This method provides you with the most flexibility and has all the functionality provided by **pspell_new()** and **pspell_new_personal()**.

The config parameter is the one returned by **pspell_config_create()** when the config was created.

Example 1. pspell_new_config()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl ($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_config ($pspell_config);
```

pspell_new_personal

(PHP 4 >= 4.0.2)

pspell_new_personal -- Load a new dictionary with personal wordlist

Description

int **pspell_new_personal** (string personal, string language [, string spelling [, string jargon [, string encoding [, int mode]]]])

pspell_new_personal() opens up a new dictionary with a personal wordlist and returns the dictionary link identifier for use in other pspell functions. The wordlist can be modified and saved with **pspell_save_wordlist()**, if desired. However, the replacement pairs are not saved. In order to save replacement pairs, you should create a config using **pspell_config_create()**, set the personal wordlist file with **pspell_config_personal()**, set the file for replacement pairs with **pspell_config_repl()**, and open a new dictionary with **pspell_new_config()**.

The personal parameter specifies the file where words added to the personal list will be stored. It should be an absolute filename beginning with '/' because otherwise it will be relative to $HOME, which is "/root" for most systems, and is probably not what you want.

The language parameter is the language code which consists of the two letter ISO 639 language code and an optional two letter ISO 3166 country code after a dash or underscore.

The spelling parameter is the requested spelling for languages with more than one spelling such as English. Known values are 'american', 'british', and 'canadian'.

The jargon parameter contains extra information to distinguish two different words lists that have the same language and spelling parameters.

The encoding parameter is the encoding that words are expected to be in. Valid values are 'utf-8', 'iso8859-*', 'koi8-r', 'viscii', 'cp1252', 'machine unsigned 16', 'machine unsigned 32'. This parameter is largely untested, so be careful when using.

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- **PSPELL_FAST** - Fast mode (least number of suggestions)
- **PSPELL_NORMAL** - Normal mode (more suggestions)
- **PSPELL_BAD_SPELLERS** - Slow mode (a lot of suggestions)
- **PSPELL_RUN_TOGETHER** - Consider run-together words as legal compounds. That is, "thecat" will be a legal compound, athough there should be a space between the two words. Changing this setting only affects the results returned by **pspell_check()**; **pspell_suggest()** will still return suggestions.

Mode is a bitmask constructed from different constants listed above. However, **PSPELL_FAST**, **PSPELL_NORMAL** and **PSPELL_BAD_SPELLERS** are mutually exclusive, so you should select only one of them.

For more information and examples, check out inline manual pspell website:http://aspell.net/.

Example 1. pspell_new_personal()

```
$pspell_link = pspell_new_personal ("/var/dictionaries/custom.pws",
        "en", "", "", "", PSPELL_FAST|PSPELL_RUN_TOGETHER));
```

pspell_new

(PHP 4 >= 4.0.2)

pspell_new -- Load a new dictionary

Description

int **pspell_new** (string language [, string spelling [, string jargon [, string encoding [, int mode]]]])

pspell_new() opens up a new dictionary and returns the dictionary link identifier for use in other pspell functions.

The language parameter is the language code which consists of the two letter ISO 639 language code and an optional two letter ISO 3166 country code after a dash or underscore.

The spelling parameter is the requested spelling for languages with more than one spelling such as English. Known values are 'american', 'british', and 'canadian'.

The jargon parameter contains extra information to distinguish two different words lists that have the same language and spelling parameters.

The encoding parameter is the encoding that words are expected to be in. Valid values are 'utf-8', 'iso8859-*', 'koi8-r', 'viscii', 'cp1252', 'machine unsigned 16', 'machine unsigned 32'. This parameter is largely untested, so be careful when using.

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- **PSPELL_FAST** - Fast mode (least number of suggestions)
- **PSPELL_NORMAL** - Normal mode (more suggestions)
- **PSPELL_BAD_SPELLERS** - Slow mode (a lot of suggestions)
- **PSPELL_RUN_TOGETHER** - Consider run-together words as legal compounds. That is, "thecat" will be a legal compound, athough there should be a space between the two words. Changing this setting only affects the results returned by pspell_check(); pspell_suggest() will still return suggestions.

Mode is a bitmask constructed from different constants listed above. However, **PSPELL_FAST**, **PSPELL_NORMAL** and **PSPELL_BAD_SPELLERS** are mutually exclusive, so you should select only one of them.

For more information and examples, check out inline manual pspell website:http://aspell.net/.

Example 1. pspell_new()

```
$pspell_link = pspell_new ("en", "", "", "",
                           (PSPELL_FAST|PSPELL_RUN_TOGETHER));
```

pspell_save_wordlist

(PHP 4 >= 4.0.2)

pspell_save_wordlist -- Save the personal wordlist to a file

Description

int **pspell_save_wordlist** (int dictionary_link)

pspell_save_wordlist() saves the personal wordlist from the current session. The dictionary has to be open with pspell_new_personal(), and the location of files to be saved specified with pspell_config_personal() and (optionally) pspell_config_repl(). Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Example 1. pspell_add_to_personal()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/tmp/dicts/newdict");
$pspell_link = pspell_new_config ($pspell_config);

pspell_add_to_personal ($pspell_link, "Vlad");
pspell_save_wordlist ($pspell_link);
```

pspell_store_replacement

(PHP 4 >= 4.0.2)

pspell_store_replacement -- Store a replacement pair for a word

Description

int **pspell_store_replacement** (int dictionary_link, string misspelled, string correct)

pspell_store_replacement() stores a replacement pair for a word, so that replacement can be returned by **pspell_suggest()** later. In order to be able to take advantage of this function, you have to use **pspell_new_personal()** to open the dictionary. In order to permanently save the replacement pair, you have to use **pspell_config_personal()** and **pspell_config_repl()** to set the path where to save your custom wordlists, and then use **pspell_save_wordlist()** for the changes to be written to disk. Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Example 1. pspell_store_replacement()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl ($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_config ($pspell_config);

pspell_store_replacement ($pspell_link, $misspelled, $correct);
pspell_save_wordlist ($pspell_link);
```

pspell_suggest

(PHP 4 >= 4.0.2)

pspell_suggest -- Suggest spellings of a word

Description

array **pspell_suggest** (int dictionary_link, string word)

pspell_suggest() returns an array of possible spellings for the given word.

Example 1. pspell_suggest()

```
$pspell_link = pspell_new ("en");

if (!pspell_check ($pspell_link, "testt")) {
    $suggestions = pspell_suggest ($pspell_link, "testt");

    foreach ($suggestions as $suggestion) {
        echo "Possible spelling: $suggestion<br>";
    }
}
```

LXXXVI. GNU Readline

Introduction

The **readline()** functions implement an interface to the GNU Readline library. These are functions that provide editable command lines. An example being the way Bash allows you to use the arrow keys to insert characters or scroll through command history. Because of the interactive nature of this library, it will be of little use for writing Web applications, but may be useful when writing scripts meant using PHP from the command line.

Note: This extension is not available on Windows platforms.

Requirements

To use the readline functions, you need to install libreadline. You can find libreadline on the home page of the GNU Readline project, at http://cnswww.cns.cwru.edu/~chet/readline/rltop.html. It's maintained by Chet Ramey, who's also the author of Bash.

You can also use this functions with the libedit library, a non-GPL replacement for the readline library. The libedit library is BSD licensend and available for download from http://sourceforge.net/projects/libedit/.

Installation

To use this functions you must compile the CGI or CLI version of PHP with readline support. You need to configure PHP `--with-readline[=DIR]`. In order you want to use the libedit readline replacement, configure PHP `--with-libedit[=DIR]`.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Table of Contents

readline_add_history

(PHP 4)

readline_add_history -- Adds a line to the history

Description

void **readline_add_history** (string line)

This function adds a line to the command line history.

readline_clear_history

(PHP 4)

readline_clear_history -- Clears the history

Description

bool **readline_clear_history** (void)

This function clears the entire command line history.

readline_completion_function

(PHP 4)

readline_completion_function -- Registers a completion function

Description

bool **readline_completion_function** (string line)

This function registers a completion function. You must supply the name of an existing function which accepts a partial command line and returns an array of possible matches. This is the same kind of functionality you'd get if you hit your tab key while using Bash.

readline_info

(PHP 4)

readline_info -- Gets/sets various internal readline variables

Description

mixed **readline_info** ([string varname [, string newvalue]])

If called with no parameters, this function returns an array of values for all the setting readline uses. The elements will be indexed by the following values: done, end, erase_empty_line, library_version, line_buffer, mark, pending_input, point, prompt, readline_name, and terminal_name.

If called with one parameter, the value of that setting is returned. If called with two parameters, the setting will be changed to the given value.

readline_list_history

(PHP 4)

readline_list_history -- Lists the history

Description

array **readline_list_history** (void)

This function returns an array of the entire command line history. The elements are indexed by integers starting at zero.

readline_read_history

(PHP 4)

readline_read_history -- Reads the history

Description

bool **readline_read_history** (string filename)

This function reads a command history from a file.

readline_write_history

(PHP 4)

readline_write_history -- Writes the history

Description

bool **readline_write_history** (string filename)

This function writes the command history to a file.

readline

(PHP 4)

readline -- Reads a line

Description

string **readline** ([string prompt])

This function returns a single string from the user. You may specify a string with which to prompt the user. The line returned has the ending newline removed. You must add this line to the history yourself using **readline_add_history()**.

Example 1. readline()

```
//get 3 commands from user
for ($i=0; $i < 3; $i++) {
        $line = readline ("Command: ");
        readline_add_history ($line);
}

//dump history
print_r (readline_list_history());

//dump variables
print_r (readline_info());
```

LXXXVII. GNU Recode functions

Introduction

This module contains an interface to the GNU Recode library, version 3.5. The GNU Recode library converts files between various coded character sets and surface encodings. When this cannot be achieved exactly, it may get rid of the offending characters or fall back on approximations. The library recognises or produces nearly 150 different character sets and is able to convert files between almost any pair. Most RFC 1345 character sets are supported.

> **Note:** This extension is not available on Windows platforms.

Requirements

You must have GNU Recode 3.5 or higher installed on your system. You can download the package from here.

Installation

To be able to use the functions defined in this module you must compile your PHP interpreter using the `--with-recode[=DIR]` option.

Warning
Crashes and startup problems of PHP may be encountered when loading the recode as extension **after** loading any extension of mysql or imap. Loading the recode before those extension has proven to fix the problem. This is due a technical problem that both the c-client library used by imap and recode have their own `hash_lookup()` function and both mysql and recode have their own `hash_insert` function.

Warning
The IMAP extension cannot be used in conjuction with the recode or YAZ extensions. This is due to the fact that they both share the same internal symbol.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Table of Contents

recode_file

(PHP 3>= 3.0.13, PHP 4)

recode_file -- Recode from file to file according to recode request

Description

bool **recode_file** (string request, resource input, resource output)

Recode the file referenced by file handle *input* into the file referenced by file handle *output* according to the recode *request*. Returns **FALSE**, if unable to comply, **TRUE** otherwise.

This function does not currently process filehandles referencing remote files (URLs). Both filehandles must refer to local files.

Example 1. Basic recode_file() example

```
$input = fopen ('input.txt', 'r');
$output = fopen ('output.txt', 'w');
recode_file ("us..flat", $input, $output);
```

recode_string

(PHP 3>= 3.0.13, PHP 4)

recode_string -- Recode a string according to a recode request

Description

string **recode_string** (string request, string string)

Recode the string *string* according to the recode request *request*. Returns the recoded string or **FALSE**, if unable to perform the recode request.

A simple recode request may be "lat1..iso646-de". See also the GNU Recode documentation of your installation for detailed instructions about recode requests.

Example 1. Basic recode_string() example:

```
print recode_string ("us..flat", "The following character has a diacritical mark: &aacute;");
```

recode

(PHP 4)

recode -- Recode a string according to a recode request

Description

string **recode** (string request, string string)

> **Note:** This is an alias for **recode_string()**. It has been added in PHP 4.

LXXXVIII. Regular Expression Functions (Perl-Compatible)

Introduction

The syntax for patterns used in these functions closely resembles Perl. The expression should be enclosed in the delimiters, a forward slash (/), for example. Any character can be used for delimiter as long as it's not alphanumeric or backslash (\). If the delimiter character has to be used in the expression itself, it needs to be escaped by backslash. Since PHP 4.0.4, you can also use Perl-style (), {}, [], and <> matching delimiters.

The ending delimiter may be followed by various modifiers that affect the matching. See Pattern Modifiers.

PHP also supports regular expressions using a POSIX-extended syntax using the POSIX-extended regex functions..

Requirements

Regular expression support is provided by the PCRE library package, which is open source software, written by Philip Hazel, and copyright by the University of Cambridge, England. It is available at ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/.

Installation

Beginning with PHP 4.2.0 these functions are enabled by default. You can disable the pcre functions with `--without-pcre-regex`. Use `--with-pcre-regex=DIR` to specify DIR where PCRE's include and library files are located, if not using bundled library. For older versions you have to configure and compile PHP with `--with-pcre-regex[=DIR]` in order to use these functions.

The windows version of PHP has built in support for this extension. You do not need to load any additional extension in order to

use these functions.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

Table 1. PREG constants

constant	description
PREG_PATTERN_ORDER	Orders results so that $matches[0] is an array of full pattern matches, $matches[1] is an array of strings matched by the first parenthesized subpattern, and so on. This flag is only used with **preg_match_all()**.
PREG_SET_ORDER	Orders results so that $matches[0] is an array of first set of matches, $matches[1] is an array of second set of matches, and so on. This flag is only used with **preg_match_all()**.
PREG_OFFSET_CAPTURE	See the description of **PREG_SPLIT_OFFSET_CAPTURE**. This flag is available since PHP 4.3.0 .
PREG_SPLIT_NO_EMPTY	This flag tells **preg_split()** to only return only non-empty pieces.
PREG_SPLIT_DELIM_CAPTURE	This flag tells **preg_split()** to capture parenthesized expression in the delimiter pattern as well. This flag is available since PHP 4.0.5 .
PREG_SPLIT_OFFSET_CAPTURE	If this flag is set, for every occuring match the appendant string offset will also be returned. Note that this changes the return value in an array where very element is an array consisting of the matched string at offset 0 and it's string offset into subject at offset 1. This flag is available since PHP 4.3.0 and is only used for **preg_split()**.

Examples

Example 1. Examples of valid patterns

- `/<\/\w+>/`
- `|(\d{3})-\d+|Sm`
- `/^(?i)php[34]/`
- `{^\s+(\s+)?$}`

Example 2. Examples of invalid patterns

- `/href='(.*)'` - missing ending delimiter
- `/\w+\s*\w+/J` - unknown modifier 'J'
- `1-\d3-\d3-\d4|` - missing starting delimiter

Table of Contents

preg_grep -- Return array entries that match the pattern
preg_match_all -- Perform a global regular expression match
preg_match -- Perform a regular expression match
preg_quote -- Quote regular expression characters
preg_replace_callback -- Perform a regular expression search and replace using a callback
preg_replace -- Perform a regular expression search and replace
preg_split -- Split string by a regular expression

Pattern Modifiers

Pattern Modifiers -- Describes possible modifiers in regex patterns

Description

The current possible PCRE modifiers are listed below. The names in parentheses refer to internal PCRE names for these modifiers.

i (PCRE_CASELESS)

If this modifier is set, letters in the pattern match both upper and lower case letters.

m (PCRE_MULTILINE)

By default, PCRE treats the subject string as consisting of a single "line" of characters (even if it actually contains several newlines). The "start of line" metacharacter (^) matches only at the start of the string, while the "end of line" metacharacter ($) matches only at the end of the string, or before a terminating newline (unless *D* modifier is set). This is the same as Perl.

When this modifier is set, the "start of line" and "end of line" constructs match immediately following or immediately before any newline in the subject string, respectively, as well as at the very start and end. This is equivalent to Perl's /m modifier. If there are no "\n" characters in a subject string, or no occurrences of ^ or $ in a pattern, setting this modifier has no effect.

s (PCRE_DOTALL)

If this modifier is set, a dot metacharacter in the pattern matches all characters, including newlines. Without it, newlines are excluded. This modifier is equivalent to Perl's /s modifier. A negative class such as [^a] always matches a newline character, independent of the setting of this modifier.

x (PCRE_EXTENDED)

If this modifier is set, whitespace data characters in the pattern are totally ignored except when escaped or inside a character class, and characters between an unescaped # outside a character class and the next newline character, inclusive, are also ignored. This is equivalent to Perl's /x modifier, and makes it possible to include comments inside complicated patterns. Note, however, that this applies only to data characters. Whitespace characters may never appear within special character sequences in a pattern, for example within the sequence (?(which introduces a conditional subpattern.

e

If this modifier is set, **preg_replace()** does normal substitution of backreferences in the replacement string, evaluates it as PHP code, and uses the result for replacing the search string.

Only **preg_replace()** uses this modifier; it is ignored by other PCRE functions.

Note: This modifier was not available in PHP3.

A (PCRE_ANCHORED)

If this modifier is set, the pattern is forced to be "anchored", that is, it is constrained to match only at the start of the string which is being searched (the "subject string"). This effect can also be achieved by appropriate constructs in the pattern itself, which is the only way to do it in Perl.

D (PCRE_DOLLAR_ENDONLY)

If this modifier is set, a dollar metacharacter in the pattern matches only at the end of the subject string. Without this modifier, a dollar also matches immediately before the final character if it is a newline (but not before any other newlines). This modifier is ignored if *m* modifier is set. There is no equivalent to this modifier in Perl.

S

When a pattern is going to be used several times, it is worth spending more time analyzing it in order to speed up the time taken for matching. If this modifier is set, then this extra analysis is performed. At present, studying a pattern is useful only for non-anchored patterns that do not have a single fixed starting character.

U (PCRE_UNGREEDY)

This modifier inverts the "greediness" of the quantifiers so that they are not greedy by default, but become greedy if followed by "?". It is not compatible with Perl. It can also be set by a (?U) modifier setting within the pattern.

X (PCRE_EXTRA)

This modifier turns on additional functionality of PCRE that is incompatible with Perl. Any backslash in a pattern that is followed by a letter that has no special meaning causes an error, thus reserving these combinations for future expansion. By default, as in Perl, a backslash followed by a letter with no special meaning is treated as a literal. There are at present no other features controlled by this modifier.

u (PCRE_UTF8)

This modifier turns on additional functionality of PCRE that is incompatible with Perl. Pattern strings are treated as UTF-8. This modifier is available from PHP 4.1.0 or greater on Unix and from PHP 4.2.3 on win32.

Pattern Syntax

Pattern Syntax -- Describes PCRE regex syntax

Description

The PCRE library is a set of functions that implement regular expression pattern matching using the same syntax and semantics as Perl 5, with just a few differences (see below). The current implementation corresponds to Perl 5.005.

Differences From Perl

The differences described here are with respect to Perl 5.005.

1. By default, a whitespace character is any character that the C library function isspace() recognizes, though it is possible to compile PCRE with alternative character type tables. Normally isspace() matches space, formfeed, newline, carriage return, horizontal tab, and vertical tab. Perl 5 no longer includes vertical tab in its set of whitespace characters. The \v escape that was in the Perl documentation for a long time was never in fact recognized. However, the character itself was treated as whitespace at least up to 5.002. In 5.004 and 5.005 it does not match \s.

2. PCRE does not allow repeat quantifiers on lookahead assertions. Perl permits them, but they do not mean what you might think. For example, (?!a){3} does not assert that the next three characters are not "a". It just asserts that the next character is not "a" three times.

3. Capturing subpatterns that occur inside negative looka- head assertions are counted, but their entries in the offsets vector are never set. Perl sets its numerical vari- ables from any such patterns that are matched before the assertion fails to match something (thereby succeeding), but only if the negative lookahead assertion contains just one branch.

4. Though binary zero characters are supported in the subject string, they are not allowed in a pattern string because it is passed as a normal C string, terminated by zero. The escape sequence "\\x00" can be used in the pattern to represent a binary zero.

5. The following Perl escape sequences are not supported: \l, \u, \L, \U, \E, \Q. In fact these are implemented by Perl's general string-handling and are not part of its pat- tern matching engine.

6. The Perl \G assertion is not supported as it is not relevant to single pattern matches.

7. Fairly obviously, PCRE does not support the (?{code}) construction.

8. There are at the time of writing some oddities in Perl 5.005_02 concerned with the settings of captured strings when part of a pattern is repeated. For example, matching "aba" against the pattern /^(a(b)?)+$/ sets $2 to the value "b", but matching "aabbaa" against /^(aa(bb)?)+$/ leaves $2 unset. However, if the pattern is changed to /^(aa(b(b))?)+$/ then $2 (and $3) get set. In Perl 5.004 $2 is set in both cases, and that is also **TRUE** of PCRE. If in the future Perl changes to a consistent state that is different, PCRE may change to follow.

9. Another as yet unresolved discrepancy is that in Perl 5.005_02 the pattern /^(a)?(?(1)a|b)+$/ matches the string "a", whereas in PCRE it does not. However, in both Perl and PCRE /^(a)?a/ matched against "a" leaves $1 unset.

10. PCRE provides some extensions to the Perl regular expression facilities:

a. Although lookbehind assertions must match fixed length strings, each alternative branch of a lookbehind assertion can match a different length of string. Perl 5.005 requires them all to have the same length.

b. If PCRE_DOLLAR_ENDONLY is set and PCRE_MULTILINE is not set, the $ meta- character matches only at the very end of the string.

c. If PCRE_EXTRA is set, a backslash followed by a letter with no special meaning is faulted.

d. If PCRE_UNGREEDY is set, the greediness of the repeti- tion quantifiers is inverted, that is, by default they are not greedy, but if followed by a question mark they are.

Regular Expression Details

Introduction

The syntax and semantics of the regular expressions sup- ported by PCRE are described below. Regular expressions are also described in the Perl documentation and in a number of other books, some of which have copious examples. Jeffrey Friedl's "Mastering Regular Expressions", published by O'Reilly (ISBN 1-56592-257-3), covers them in great detail. The description here is intended as reference documentation. A regular expression is a pattern that is matched against a subject string from left to right. Most characters stand for themselves in a pattern, and match the corresponding charac- ters in the subject. As a trivial example, the pattern `The quick brown fox` matches a portion of a subject string that is identical to itself.

Meta-characters

The power of regular expressions comes from the ability to include alternatives and repetitions in the pat- tern. These are encoded in the pattern by the use of *meta- characters*, which do not stand for themselves but instead are interpreted in some special way.

There are two different sets of meta-characters: those that are recognized anywhere in the pattern except within square brackets, and those that are recognized in square brackets. Outside square brackets, the meta-characters are as follows:

\

general escape character with several uses

^

assert start of subject (or line, in multiline mode)

$

assert end of subject (or line, in multiline mode)

.

match any character except newline (by default)

[

start character class definition

]

end character class definition

|

start of alternative branch

(

start subpattern

)

end subpattern

?

extends the meaning of (, also 0 or 1 quantifier, also quantifier minimizer

*

o or more quantifier

+

1 or more quantifier

{

start min/max quantifier

}

end min/max quantifier

Part of a pattern that is in square brackets is called a "character class". In a character class the only meta- characters are:

\

general escape character

^

negate the class, but only if the first character

-

indicates character range

]

terminates the character class

The following sections describe the use of each of the meta-characters.

backslash

The backslash character has several uses. Firstly, if it is followed by a non-alphanumeric character, it takes away any special meaning that character may have. This use of backslash as an escape character applies both inside and outside character classes.

For example, if you want to match a "*" character, you write "*" in the pattern. This applies whether or not the follow- ing character would otherwise be interpreted as a meta- character, so it is always safe to precede a non-alphanumeric with "\" to specify that it stands for itself. In particu- lar, if you want to match a backslash, you write "\\".

If a pattern is compiled with the PCRE_EXTENDED option, whi- tespace in the pattern (other than in a character class) and characters between a "#" outside a character class and the next newline character are ignored. An escaping backslash can be used to include a whitespace or "#" character as part of the pattern.

A second use of backslash provides a way of encoding non- printing characters in patterns in a visible manner. There is no restriction on the appearance of non-printing characters, apart from the binary zero that terminates a pattern, but when a pattern is being prepared by text editing, it is usually easier to use one of the following escape sequences than the binary character it represents:

\a

alarm, that is, the BEL character (hex 07)

\cx

"control-x", where x is any character

\e

escape (hex 1B)

\f

formfeed (hex 0C)

\n

newline (hex 0A)

\r

carriage return (hex 0D)

\t

tab (hex 09)

\xhh

character with hex code hh

\ddd

character with octal code ddd, or backreference

The precise effect of "`\cx`" is as follows: if "`x`" is a lower case letter, it is converted to upper case. Then bit 6 of the character (hex 40) is inverted. Thus "`\cz`" becomes hex 1A, but "`\c{`" becomes hex 3B, while "`\c;`" becomes hex 7B.

After "`\x`", up to two hexadecimal digits are read (letters can be in upper or lower case).

After "`\0`" up to two further octal digits are read. In both cases, if there are fewer than two digits, just those that are present are used. Thus the sequence "`\0\x\07`" specifies two binary zeros followed by a BEL character. Make sure you supply two digits after the initial zero if the character that follows is itself an octal digit.

The handling of a backslash followed by a digit other than 0 is complicated. Outside a character class, PCRE reads it and any following digits as a decimal number. If the number is less than 10, or if there have been at least that many previous capturing left parentheses in the expression, the entire sequence is taken as a *back reference*. A description of how this works is given later, following the discussion of parenthesized subpatterns.

Inside a character class, or if the decimal number is greater than 9 and there have not been that many capturing subpatterns, PCRE re-reads up to three octal digits following the backslash, and generates a single byte from the least significant 8 bits of the value. Any subsequent digits stand for themselves. For example:

\040

is another way of writing a space

\40

is the same, provided there are fewer than 40 previous capturing subpatterns

\7

is always a back reference

\11

might be a back reference, or another way of writing a tab

\011

is always a tab

\0113

is a tab followed by the character "3"

\113

is the character with octal code 113 (since there can be no more than 99 back references)

\377

is a byte consisting entirely of 1 bits

\81

is either a back reference, or a binary zero followed by the two characters "8" and "1"

Note that octal values of 100 or greater must not be intro- duced by a leading zero, because no more than three octal digits are ever read.

All the sequences that define a single byte value can be used both inside and outside character classes. In addition, inside a character class, the sequence "`\b`" is interpreted as the backspace character (hex 08). Outside a character class it has a different meaning (see below).

The third use of backslash is for specifying generic charac- ter types:

\d

any decimal digit

\D

any character that is not a decimal digit

\s

any whitespace character

\S

any character that is not a whitespace character

\w

any "word" character

\W

any "non-word" character

Each pair of escape sequences partitions the complete set of characters into two disjoint sets. Any given character matches one, and only one, of each pair.

A "word" character is any letter or digit or the underscore character, that is, any character which can be part of a Perl "`word`". The definition of letters and digits is controlled by PCRE's character tables, and may vary if locale-specific matching is taking place (see "Locale support" above). For example, in the "fr" (French) locale, some char- acter codes greater than 128 are used for accented letters, and these are matched by `\w`.

These character type sequences can appear both inside and outside character classes. They each match one character of the appropriate type. If the current matching point is at the end of the subject string, all of them fail, since there is no character to match.

The fourth use of backslash is for certain simple asser- tions. An assertion specifies a condition that has to be met at a particular point in a match, without consuming any characters from the subject string. The use of subpatterns for more complicated assertions is described below. The backslashed assertions are

\b

word boundary

\B

not a word boundary

\A

start of subject (independent of multiline mode)

\Z

end of subject or newline at end (independent of multiline mode)

\z

end of subject(independent of multiline mode)

These assertions may not appear in character classes (but note that "`\b`" has a different meaning, namely the backspace character, inside a character class).

A word boundary is a position in the subject string where the current character and the previous character do not both match `\w` or `\W` (i.e. one matches `\w` and the other matches `\W`), or the start or end of the string if the first or last character matches \w, respectively.

The `\A`, `\Z`, and `\z` assertions differ from the traditional circumflex and dollar (described below) in that they only ever match at the very start and end of the subject string, whatever options are set. They are not affected by the PCRE_NOTBOL or PCRE_NOTEOL options. The difference between `\Z` and `\z` is that `\Z` matches before a newline that is the last character of the string as well as at the end of the string, whereas `\z` matches only at the end.

Circumflex and dollar

Outside a character class, in the default matching mode, the circumflex character is an assertion which is true only if the current matching point is at the start of the subject string. Inside a character class, circumflex has an entirely different meaning (see below).

Circumflex need not be the first character of the pattern if a number of alternatives are involved, but it should be the first thing in each alternative in which it appears if the pattern is ever to match that branch. If all possible alternatives start with a circumflex, that is, if the pattern is constrained to match only at the start of the subject, it is said to be an "anchored" pattern. (There are also other constructs that can cause a pattern to be anchored.)

A dollar character is an assertion which is **TRUE** only if the current matching point is at the end of the subject string, or immediately before a newline character that is the last character in the string (by default). Dollar need not be the last character of the pattern if a number of alternatives are involved, but it should be the last item in any branch in which it appears. Dollar has no special meaning in a character class.

The meaning of dollar can be changed so that it matches only at the very end of the string, by setting the PCRE_DOLLAR_ENDONLY option at compile or matching time. This does not affect the \Z assertion.

The meanings of the circumflex and dollar characters are changed if the PCRE_MULTILINE option is set. When this is the case, they match immediately after and immediately before an internal "\n" character, respectively, in addition to matching at the start and end of the subject string. For example, the pattern /^abc$/ matches the subject string "def\nabc" in multiline mode, but not otherwise. Consequently, patterns that are anchored in single line mode because all branches start with "^" are not anchored in multiline mode. The PCRE_DOLLAR_ENDONLY option is ignored if PCRE_MULTILINE is set.

Note that the sequences \A, \Z, and \z can be used to match the start and end of the subject in both modes, and if all branches of a pattern start with \A is it always anchored, whether PCRE_MULTILINE is set or not.

FULL STOP

Outside a character class, a dot in the pattern matches any one character in the subject, including a non-printing character, but not (by default) newline. If the PCRE_DOTALL option is set, then dots match newlines as well. The handling of dot is entirely independent of the handling of circumflex and dollar, the only relationship being that they both involve newline characters. Dot has no special meaning in a character class.

Square brackets

An opening square bracket introduces a character class, terminated by a closing square bracket. A closing square bracket on its own is not special. If a closing square bracket is required as a member of the class, it should be the first data character in the class (after an initial cir-

cumflex, if present) or escaped with a backslash.

A character class matches a single character in the subject; the character must be in the set of characters defined by the class, unless the first character in the class is a circumflex, in which case the subject character must not be in the set defined by the class. If a circumflex is actually required as a member of the class, ensure it is not the first character, or escape it with a backslash.

For example, the character class [aeiou] matches any lower case vowel, while [^aeiou] matches any character that is not a lower case vowel. Note that a circumflex is just a convenient notation for specifying the characters which are in the class by enumerating those that are not. It is not an assertion: it still consumes a character from the subject string, and fails if the current pointer is at the end of the string.

When caseless matching is set, any letters in a class represent both their upper case and lower case versions, so for example, a caseless [aeiou] matches "A" as well as "a", and a caseless [^aeiou] does not match "A", whereas a caseful version would.

The newline character is never treated in any special way in character classes, whatever the setting of the PCRE_DOTALL or PCRE_MULTILINE options is. A class such as [^a] will always match a newline.

The minus (hyphen) character can be used to specify a range of characters in a character class. For example, [d-m] matches any letter between d and m, inclusive. If a minus character is required in a class, it must be escaped with a backslash or appear in a position where it cannot be interpreted as indicating a range, typically as the first or last character in the class.

It is not possible to have the literal character "]" as the end character of a range. A pattern such as [W-]46] is interpreted as a class of two characters ("W" and "-") followed by a literal string "46]", so it would match "W46]" or "-46]". However, if the "]" is escaped with a backslash it is interpreted as the end of range, so [W-\]46] is interpreted as a single class containing a range followed by two separate characters. The octal or hexadecimal representation of "]" can also be used to end a range.

Ranges operate in ASCII collating sequence. They can also be used for characters specified numerically, for example [\000-\037]. If a range that includes letters is used when caseless matching is set, it matches the letters in either case. For example, [W-c] is equivalent to [][\^_`wxyzabc], matched caselessly, and if character tables for the "fr" locale are in use, [\xc8-\xcb] matches accented E characters in both cases.

The character types \d, \D, \s, \S, \w, and \W may also appear in a character class, and add the characters that they match to the class. For example, [\dABCDEF] matches any hexadecimal digit. A circumflex can conveniently be used with the upper case character types to specify a more restricted set of characters than the matching lower case type. For example, the class [^\W_] matches any letter or digit, but not underscore.

All non-alphanumeric characters other than \, -, ^ (at the start) and the terminating] are non-special in character classes, but it does no harm if they are escaped.

Vertical bar

Vertical bar characters are used to separate alternative patterns. For example, the pattern

```
gilbert|sullivan
```

matches either "gilbert" or "sullivan". Any number of alternatives may appear, and an empty alternative is permitted (matching the empty string). The matching process tries each alternative in turn, from left to right, and the first one that succeeds is used. If the alternatives are within a subpattern (defined below), "succeeds" means matching the rest of the main pattern as well as the alternative in the subpattern.

Internal option setting

The settings of PCRE_CASELESS , PCRE_MULTILINE , PCRE_DOTALL , and PCRE_EXTENDED can be changed from within the pattern by a sequence of Perl option letters enclosed between "(?" and ")". The option letters are

```
i  for PCRE_CASELESS
m  for PCRE_MULTILINE
s  for PCRE_DOTALL
x  for PCRE_EXTENDED
```

For example, (?im) sets caseless, multiline matching. It is also possible to unset these options by preceding the letter with a hyphen, and a combined setting and unsetting such as (?im-sx), which sets PCRE_CASELESS and PCRE_MULTILINE while unsetting PCRE_DOTALL and PCRE_EXTENDED , is also permitted. If a letter appears both before and after the hyphen, the option is unset.

The scope of these option changes depends on where in the pattern the setting occurs. For settings that are outside any subpattern (defined below), the effect is the same as if the options were set or unset at the start of matching. The following patterns all behave in exactly the same way:

```
(?i)abc
a(?i)bc
ab(?i)c
abc(?i)
```

which in turn is the same as compiling the pattern abc with PCRE_CASELESS set.
In other words, such "top level" settings apply to the whole pattern (unless there are other changes inside subpatterns). If there is more than one setting of the same option at top level, the rightmost setting is used.

If an option change occurs inside a subpattern, the effect is different. This is a change of behaviour in Perl 5.005. An option change inside a subpattern affects only that part of the subpattern that follows it, so

```
(a(?i)b)c
```

matches abc and aBc and no other strings (assuming PCRE_CASELESS is not used). By this means, options can be made to have different settings in different parts of the pattern. Any changes made in one alternative do carry on

into subsequent branches within the same subpattern. For example,

(a(?i)b|c)

matches "ab", "aB", "c", and "C", even though when matching "C" the first branch is abandoned before the option setting. This is because the effects of option settings happen at compile time. There would be some very weird behaviour otherwise.

The PCRE-specific options PCRE_UNGREEDY and PCRE_EXTRA can be changed in the same way as the Perl-compatible options by using the characters U and X respectively. The (?X) flag setting is special in that it must always occur earlier in the pattern than any of the additional features it turns on, even when it is at top level. It is best put at the start.

subpatterns

Subpatterns are delimited by parentheses (round brackets), which can be nested. Marking part of a pattern as a subpattern does two things:

1. It localizes a set of alternatives. For example, the pattern

cat(aract|erpillar|)

matches one of the words "cat", "cataract", or "caterpillar". Without the parentheses, it would match "cataract", "erpillar" or the empty string.

2. It sets up the subpattern as a capturing subpattern (as defined above). When the whole pattern matches, that portion of the subject string that matched the subpattern is passed back to the caller via the *ovector* argument of **pcre_exec()**. Opening parentheses are counted from left to right (starting from 1) to obtain the numbers of the capturing subpatterns.

For example, if the string "the red king" is matched against the pattern

the ((red|white) (king|queen))

the captured substrings are "red king", "red", and "king", and are numbered 1, 2, and 3.

The fact that plain parentheses fulfil two functions is not always helpful. There are often times when a grouping subpattern is required without a capturing requirement. If an opening parenthesis is followed by "?:", the subpattern does not do any capturing, and is not counted when computing the number of any subsequent capturing subpatterns. For example, if the string "the white queen" is matched against the pattern

the ((?:red|white) (king|queen))

the captured substrings are "white queen" and "queen", and are numbered 1 and 2. The maximum number of captured substrings is 99, and the maximum number of all subpatterns, both capturing and non-capturing, is 200.

As a convenient shorthand, if any option settings are required at the start of a non-capturing subpattern, the

option letters may appear between the "?" and the ":". Thus the two patterns

```
(?i:saturday|sunday)
(?:(?i)saturday|sunday)
```

match exactly the same set of strings. Because alternative branches are tried from left to right, and options are not reset until the end of the subpattern is reached, an option setting in one branch does affect subsequent branches, so the above patterns match "SUNDAY" as well as "Saturday".

Repetition

Repetition is specified by quantifiers, which can follow any of the following items:

- a single character, possibly escaped
- the . metacharacter
- a character class
- a back reference (see next section)
- a parenthesized subpattern (unless it is an assertion - see below)

The general repetition quantifier specifies a minimum and maximum number of permitted matches, by giving the two numbers in curly brackets (braces), separated by a comma. The numbers must be less than 65536, and the first must be less than or equal to the second. For example:

```
z{2,4}
```

matches "zz", "zzz", or "zzzz". A closing brace on its own is not a special character. If the second number is omitted, but the comma is present, there is no upper limit; if the second number and the comma are both omitted, the quantifier specifies an exact number of required matches. Thus

```
[aeiou]{3,}
```

matches at least 3 successive vowels, but may match many more, while

```
\d{8}
```

matches exactly 8 digits. An opening curly bracket that appears in a position where a quantifier is not allowed, or one that does not match the syntax of a quantifier, is taken as a literal character. For example, {,6} is not a quantifier, but a literal string of four characters.

The quantifier {0} is permitted, causing the expression to behave as if the previous item and the quantifier were not present.

For convenience (and historical compatibility) the three most common quantifiers have single-character abbreviations:

- `*` is equivalent to {0,}
- `+` is equivalent to {1,}
- `?` is equivalent to {0,1}

It is possible to construct infinite loops by following a subpattern that can match no characters with a quantifier that has no upper limit, for example:

```
(a?)*
```

Earlier versions of Perl and PCRE used to give an error at compile time for such patterns. However, because there are cases where this can be useful, such patterns are now accepted, but if any repetition of the subpattern does in fact match no characters, the loop is forcibly broken.

By default, the quantifiers are "greedy", that is, they match as much as possible (up to the maximum number of permitted times), without causing the rest of the pattern to fail. The classic example of where this gives problems is in trying to match comments in C programs. These appear between the sequences /* and */ and within the sequence, individual * and / characters may appear. An attempt to match C comments by applying the pattern

```
/\*.*\*/
```

to the string

```
/* first command */  not comment  /* second comment */
```

fails, because it matches the entire string due to the greediness of the .* item.

However, if a quantifier is followed by a question mark, then it ceases to be greedy, and instead matches the minimum number of times possible, so the pattern

```
/\*.*?\*/
```

does the right thing with the C comments. The meaning of the various quantifiers is not otherwise changed, just the preferred number of matches. Do not confuse this use of ques-tion mark with its use as a quantifier in its own right. Because it has two uses, it can sometimes appear doubled, as in

```
\d??\d
```

which matches one digit by preference, but can match two if that is the only way the rest of the pattern matches.

If the PCRE_UNGREEDY option is set (an option which is not available in Perl) then the quantifiers are not greedy by default, but individual ones can be made greedy by following them with a question mark. In other words, it inverts the default behaviour.

When a parenthesized subpattern is quantified with a minimum repeat count that is greater than 1 or with a limited maximum, more store is required for the compiled pattern, in proportion to the size of the minimum or maximum.

If a pattern starts with .* or .{0,} and the PCRE_DOTALL option (equivalent to Perl's /s) is set, thus allowing the . to match newlines, then the pattern is implicitly anchored, because whatever follows will be tried against every character position in the subject string, so there is no point in retrying the overall match at any position after the first. PCRE treats such a pattern as though it were preceded by \A. In cases where it is known that the subject string contains no newlines, it is worth setting PCRE_DOTALL when the pattern begins with .* in order to obtain this optimization, or alternatively using ^ to indicate anchoring explicitly.

When a capturing subpattern is repeated, the value captured is the substring that matched the final iteration. For example, after

```
(tweedle[dume]{3}\s*)+
```

has matched "tweedledum tweedledee" the value of the captured substring is "tweedledee". However, if there are nested capturing subpatterns, the corresponding captured values may have been set in previous iterations. For example, after

/(a|(b))+/

matches "aba" the value of the second captured substring is "b".

BACK REFERENCES

Outside a character class, a backslash followed by a digit greater than 0 (and possibly further digits) is a back reference to a capturing subpattern earlier (i.e. to its left) in the pattern, provided there have been that many previous capturing left parentheses.

However, if the decimal number following the backslash is less than 10, it is always taken as a back reference, and causes an error only if there are not that many capturing left parentheses in the entire pattern. In other words, the parentheses that are referenced need not be to the left of the reference for numbers less than 10. See the section entitled "Backslash" above for further details of the handling of digits following a backslash.

A back reference matches whatever actually matched the capturing subpattern in the current subject string, rather than anything matching the subpattern itself. So the pattern

(sens|respons)e and \1ibility

matches "sense and sensibility" and "response and responsibility", but not "sense and responsibility". If caseful matching is in force at the time of the back reference, then the case of letters is relevant. For example,

((?i)rah)\s+\1

matches "rah rah" and "RAH RAH", but not "RAH rah", even though the original capturing subpattern is matched caselessly.

There may be more than one back reference to the same subpattern. If a subpattern has not actually been used in a particular match, then any back references to it always fail. For example, the pattern

(a|(bc))\2

always fails if it starts to match "a" rather than "bc". Because there may be up to 99 back references, all digits following the backslash are taken as part of a potential back reference number. If the pattern continues with a digit character, then some delimiter must be used to terminate the back reference. If the PCRE_EXTENDED option is set, this can be whitespace. Otherwise an empty comment can be used.

A back reference that occurs inside the parentheses to which it refers fails when the subpattern is first used, so, for example, (a\1) never matches. However, such references can be useful inside repeated subpatterns. For example, the pattern

(a|b\1)+

matches any number of "a"s and also "aba", "ababaa" etc. At each iteration of the subpattern, the back reference matches

the character string corresponding to the previous iteration. In order for this to work, the pattern must be such that the first iteration does not need to match the back reference. This can be done using alternation, as in the example above, or by a quantifier with a minimum of zero.

Assertions

An assertion is a test on the characters following or preceding the current matching point that does not actually consume any characters. The simple assertions coded as \b, \B, \A, \Z, \z, ^ and $ are described above. More complicated assertions are coded as subpatterns. There are two kinds: those that look ahead of the current position in the subject string, and those that look behind it.

An assertion subpattern is matched in the normal way, except that it does not cause the current matching position to be changed. Lookahead assertions start with (?= for positive assertions and (?! for negative assertions. For example,

```
\w+(?=;)
```

matches a word followed by a semicolon, but does not include the semicolon in the match, and

```
foo(?!bar)
```

matches any occurrence of "foo" that is not followed by "bar". Note that the apparently similar pattern

```
(?!foo)bar
```

does not find an occurrence of "bar" that is preceded by something other than "foo"; it finds any occurrence of "bar" whatsoever, because the assertion (?!foo) is always **TRUE** when the next three characters are "bar". A lookbehind assertion is needed to achieve this effect.

Lookbehind assertions start with (?<= for positive assertions and (?<! for negative assertions. For example,

```
(?<!foo)bar
```

does find an occurrence of "bar" that is not preceded by "foo". The contents of a lookbehind assertion are restricted such that all the strings it matches must have a fixed length. However, if there are several alternatives, they do not all have to have the same fixed length. Thus

```
(?<=bullock|donkey)
```

is permitted, but

```
(?<!dogs?|cats?)
```

causes an error at compile time. Branches that match different length strings are permitted only at the top level of a lookbehind assertion. This is an extension compared with Perl 5.005, which requires all branches to match the same length of string. An assertion such as

```
(?<=ab(c|de))
```

is not permitted, because its single top-level branch can match two different lengths, but it is acceptable if rewritten to use two top-level branches:

```
(?<=abc|abde)
```

The implementation of lookbehind assertions is, for each alternative, to temporarily move the current position back by the fixed width and then try to match. If there are insufficient characters before the current position, the match is deemed to fail. Lookbehinds in conjunction with once-only subpatterns can be particularly useful for matching at the ends of strings; an example is given at the end of the section on once-only subpatterns.

Several assertions (of any sort) may occur in succession. For example,

```
(?<=\d{3})(?<!999)foo
```

matches "foo" preceded by three digits that are not "999". Notice that each of the assertions is applied independently at the same point in the subject string. First there is a check that the previous three characters are all digits, then there is a check that the same three characters are not "999". This pattern does not match "foo" preceded by six characters, the first of which are digits and the last three of which are not "999". For example, it doesn't match "123abcfoo". A pattern to do that is

```
(?<=\d{3}...)(?<!999)foo
```

This time the first assertion looks at the preceding six characters, checking that the first three are digits, and then the second assertion checks that the preceding three characters are not "999".

Assertions can be nested in any combination. For example,

```
(?<=(?<!foo)bar)baz
```

matches an occurrence of "baz" that is preceded by "bar" which in turn is not preceded by "foo", while

```
(?<=\d{3}(?!999)...)foo
```

is another pattern which matches "foo" preceded by three digits and any three characters that are not "999".

Assertion subpatterns are not capturing subpatterns, and may not be repeated, because it makes no sense to assert the same thing several times. If any kind of assertion contains capturing subpatterns within it, these are counted for the purposes of numbering the capturing subpatterns in the whole pattern. However, substring capturing is carried out only for positive assertions, because it does not make sense for negative assertions.

Assertions count towards the maximum of 200 parenthesized subpatterns.

Once-only subpatterns

With both maximizing and minimizing repetition, failure of what follows normally causes the repeated item to be re-evaluated to see if a different number of repeats allows the rest of the pattern to match. Sometimes it is useful to prevent this, either to change the nature of the match, or to cause it fail earlier than it otherwise might, when the author of the pattern knows there is no point in carrying on.

Consider, for example, the pattern \d+foo when applied to the subject line

```
123456bar
```

After matching all 6 digits and then failing to match "foo", the normal action of the matcher is to try again with only 5 digits matching the \d+ item, and then with 4, and so on, before ultimately failing. Once-only subpatterns provide the means for specifying that once a portion of the pattern has matched, it is not to be re-evaluated in this way, so the matcher would give up immediately on failing to match "foo" the first time. The notation is another kind of special parenthesis, starting with (?> as in this example:

```
(?>\d+)bar
```

This kind of parenthesis "locks up" the part of the pattern it contains once it has matched, and a failure further into the pattern is prevented from backtracking into it. Backtracking past it to previous items, however, works as normal.

An alternative description is that a subpattern of this type matches the string of characters that an identical standalone pattern would match, if anchored at the current point in the subject string.

Once-only subpatterns are not capturing subpatterns. Simple cases such as the above example can be thought of as a maximizing repeat that must swallow everything it can. So, while both \d+ and \d+? are prepared to adjust the number of digits they match in order to make the rest of the pattern match, (?>\d+) can only match an entire sequence of digits.

This construction can of course contain arbitrarily complicated subpatterns, and it can be nested.

Once-only subpatterns can be used in conjunction with lookbehind assertions to specify efficient matching at the end of the subject string. Consider a simple pattern such as

```
abcd$
```

when applied to a long string which does not match. Because matching proceeds from left to right, PCRE will look for each "a" in the subject and then see if what follows matches the rest of the pattern. If the pattern is specified as

```
^.*abcd$
```

then the initial .* matches the entire string at first, but when this fails (because there is no following "a"), it backtracks to match all but the last character, then all but the last two characters, and so on. Once again the search for "a" covers the entire string, from right to left, so we are no better off. However, if the pattern is written as

```
^(?>.*)(?<=abcd)
```

then there can be no backtracking for the .* item; it can match only the entire string. The subsequent lookbehind assertion does a single test on the last four characters. If it fails, the match fails immediately. For long strings, this approach makes a significant difference to the processing time.

When a pattern contains an unlimited repeat inside a subpattern that can itself be repeated an unlimited number of times, the use of a once-only subpattern is the only way to avoid some failing matches taking a very long time indeed. The pattern

```
(\D+|<\d+>)*[!?]
```

matches an unlimited number of substrings that either consist of non-digits, or digits enclosed in <>, followed by either ! or ?. When it matches, it runs quickly. However, if it is applied to

```
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

it takes a long time before reporting failure. This is because the string can be divided between the two repeats in a large number of ways, and all have to be tried. (The example used [!?] rather than a single character at the end, because both PCRE and Perl have an optimization that allows for fast failure when a single character is used. They remember the last single character that is required for a match, and fail early if it is not present in the string.) If the pattern is changed to

```
((?>\D+)|<\d+>)*[!?]
```

sequences of non-digits cannot be broken, and failure happens quickly.

Conditional subpatterns

It is possible to cause the matching process to obey a subpattern conditionally or to choose between two alternative subpatterns, depending on the result of an assertion, or whether a previous capturing subpattern matched or not. The two possible forms of conditional subpattern are

```
(?(condition)yes-pattern)
(?(condition)yes-pattern|no-pattern)
```

If the condition is satisfied, the yes-pattern is used; otherwise the no-pattern (if present) is used. If there are more than two alternatives in the subpattern, a compile-time error occurs.

There are two kinds of condition. If the text between the parentheses consists of a sequence of digits, then the condition is satisfied if the capturing subpattern of that number has previously matched. Consider the following pattern, which contains non-significant white space to make it more readable (assume the PCRE_EXTENDED option) and to divide it into three parts for ease of discussion:

```
( \( )?    [^()]+    (?(1) \) )
```

The first part matches an optional opening parenthesis, and if that character is present, sets it as the first captured substring. The second part matches one or more characters that are not parentheses. The third part is a conditional subpattern that tests whether the first set of parentheses matched or not. If they did, that is, if subject started with an opening parenthesis, the condition is **TRUE**, and so the yes-pattern is executed and a closing parenthesis is required. Otherwise, since no-pattern is not present, the subpattern matches nothing. In other words, this pattern matches a sequence of non-parentheses, optionally enclosed in parentheses.

If the condition is not a sequence of digits, it must be an assertion. This may be a positive or negative lookahead or lookbehind assertion. Consider this pattern, again containing non-significant white space, and with the two alternatives on the second line:

```
(?(?=[^a-z]*[a-z])
\d{2}-[a-z]{3}-\d{2} | \d{2}-\d{2}-\d{2} )
```

The condition is a positive lookahead assertion that matches an optional sequence of non-letters followed by a letter. In other words, it tests for the presence of at least one letter in the subject. If a letter is found, the subject is matched against the first alternative; otherwise it is matched against the second. This pattern matches strings in one of the two forms dd-aaa-dd or dd-dd-dd, where aaa are letters and dd are digits.

Comments

The sequence (?# marks the start of a comment which continues up to the next closing parenthesis. Nested parentheses are not permitted. The characters that make up a comment play no part in the pattern matching at all.

If the PCRE_EXTENDED option is set, an unescaped # character outside a character class introduces a comment that continues up to the next newline character in the pattern.

Recursive patterns

Consider the problem of matching a string in parentheses, allowing for unlimited nested parentheses. Without the use of recursion, the best that can be done is to use a pattern that matches up to some fixed depth of nesting. It is not possible to handle an arbitrary nesting depth. Perl 5.6 has provided an experimental facility that allows regular expressions to recurse (amongst other things). The special item (?R) is provided for the specific case of recursion. This PCRE pattern solves the parentheses problem (assume the PCRE_EXTENDED option is set so that white space is ignored):

```
\( ( (?>[^()]+) | (?R) )* \)
```

First it matches an opening parenthesis. Then it matches any number of substrings which can either be a sequence of non-parentheses, or a recursive match of the pattern itself (i.e. a correctly parenthesized substring). Finally there is a closing parenthesis.

This particular example pattern contains nested unlimited repeats, and so the use of a once-only subpattern for matching strings of non-parentheses is important when applying the pattern to strings that do not match. For example, when it is applied to

```
(aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa()
```

it yields "no match" quickly. However, if a once-only subpattern is not used, the match runs for a very long time indeed because there are so many different ways the + and * repeats can carve up the subject, and all have to be tested before failure can be reported.

The values set for any capturing subpatterns are those from the outermost level of the recursion at which the subpattern value is set. If the pattern above is matched against

```
(ab(cd)ef)
```

the value for the capturing parentheses is "ef", which is the last value taken on at the top level. If additional parentheses are added, giving

```
\( ( ( (?>[^()]+) | (?R) )* ) \)
  ^                   ^
  ^                   ^ then the string they capture
```

is "ab(cd)ef", the contents of the top level parentheses. If there are more than 15 capturing parentheses in a pattern, PCRE has to obtain extra memory to store data during a recursion, which it does by using pcre_malloc, freeing it via pcre_free afterwards. If no memory can be obtained, it saves data for the first 15 capturing parentheses only, as there is no way to give an out-of-memory error from within a recursion.

Performances

Certain items that may appear in patterns are more efficient than others. It is more efficient to use a character class like [aeiou] than a set of alternatives such as (a|e|i|o|u). In general, the simplest construction that provides the required behaviour is usually the most efficient. Jeffrey Friedl's book contains a lot of discussion about optimizing regular expressions for efficient performance.

When a pattern begins with .* and the PCRE_DOTALL option is set, the pattern is implicitly anchored by PCRE, since it can match only at the start of a subject string. However, if PCRE_DOTALL is not set, PCRE cannot make this optimization, because the . metacharacter does not then match a newline, and if the subject string contains newlines, the pattern may match from the character immediately following one of them instead of from the very start. For example, the pattern

```
(.*) second
```

matches the subject "first\nand second" (where \n stands for a newline character) with the first captured substring being "and". In order to do this, PCRE has to retry the match starting after every newline in the subject.

If you are using such a pattern with subject strings that do not contain newlines, the best performance is obtained by setting PCRE_DOTALL , or starting the pattern with ^.* to indicate explicit anchoring. That saves PCRE from having to scan along the subject looking for a newline to restart at.

Beware of patterns that contain nested indefinite repeats. These can take a long time to run when applied to a string that does not match. Consider the pattern fragment

```
(a+)*
```

This can match "aaaa" in 33 different ways, and this number increases very rapidly as the string gets longer. (The * repeat can match 0, 1, 2, 3, or 4 times, and for each of those cases other than 0, the + repeats can match different numbers of times.) When the remainder of the pattern is such that the entire match is going to fail, PCRE has in principle to try every possible variation, and this can take an extremely long time.

An optimization catches some of the more simple cases such as

```
(a+)*b
```

where a literal character follows. Before embarking on the standard matching procedure, PCRE checks that there is a "b" later in the subject string, and if there is not, it fails the match immediately. However, when there is no following literal this optimization cannot be used. You can see the difference by comparing the behaviour of

(a+)*\d

with the pattern above. The former gives a failure almost instantly when applied to a whole line of "a" characters, whereas the latter takes an appreciable time with strings longer than about 20 characters.

preg_grep

(PHP 4)

preg_grep -- Return array entries that match the pattern

Description

array **preg_grep** (string pattern, array input)

preg_grep() returns the array consisting of the elements of the *input* array that match the given *pattern*.

Since PHP 4.0.4, the results returned by **preg_grep()** are indexed using the keys from the input array. If this behavior is undesirable, use **array_values()** on the array returned by **preg_grep()** to reindex the values.

Example 1. preg_grep() example

```
// return all array elements
// containing floating point numbers
$fl_array = preg_grep ("/^(\d+)?\.\d+$/", $array);
```

preg_match_all

(PHP 3>= 3.0.9, PHP 4)

preg_match_all -- Perform a global regular expression match

Description

int **preg_match_all** (string pattern, string subject, array matches [, int flags])

Searches *subject* for all matches to the regular expression given in *pattern* and puts them in *matches* in the order specified by *flags*.

After the first match is found, the subsequent searches are continued on from end of the last match.

flags can be a combination of the following flags (note that it doesn't make sense to use **PREG_PATTERN_ORDER** together with **PREG_SET_ORDER**):

PREG_PATTERN_ORDER

Orders results so that $matches[0] is an array of full pattern matches, $matches[1] is an array of strings matched by the first parenthesized subpattern, and so on.

```
<?php
preg_match_all ("|<[^>]+>(.*)</[^>]+>|U",
    "<b>example: </b><div align=left>this is a test</div>",
    $out, PREG_PATTERN_ORDER);
print $out[0][0].", ".$out[0][1]."\n";
print $out[1][0].", ".$out[1][1]."\n";
?>
```

This example will produce:

```
<b>example: </b>, <div align=left>this is a test</div>
example: , this is a test
```

So, $out[0] contains array of strings that matched full pattern, and $out[1] contains array of strings enclosed by tags.

PREG_SET_ORDER

Orders results so that $matches[0] is an array of first set of matches, $matches[1] is an array of second set of matches, and so on.

```
<?php
preg_match_all ("|<[^>]+>(.*)</[^>]+>|U",
    "<b>example: </b><div align=left>this is a test</div>",
    $out, PREG_SET_ORDER);
print $out[0][0].", ".$out[0][1]."\n";
print $out[1][0].", ".$out[1][1]."\n";
?>
```

This example will produce:

```
<b>example: </b>, example:
<div align=left>this is a test</div>, this is a test
```

In this case, $matches[0] is the first set of matches, and $matches[0][0] has text matched by full pattern, $matches[0][1] has text matched by first subpattern and so on. Similarly, $matches[1] is the second set of matches, etc.

PREG_OFFSET_CAPTURE

If this flag is set, for every occuring match the appendant string offset will also be returned. Note that this changes the return value in an array where every element is an array consisting of the matched string at offset `0` and it's string offset into *`subject`* at offset `1`. This flag is available since PHP 4.3.0 .

If no order flag is given, **PREG_PATTERN_ORDER** is assumed.

Returns the number of full pattern matches (which might be zero), or **FALSE** if an error occurred.

Example 1. Getting all phone numbers out of some text.

```
<?php
preg_match_all ("/\(?  (\d{3})?  \)?  (?(1)  [\-\s] ) \d{3}-\d{4}/x",
                "Call 555-1212 or 1-800-555-1212", $phones);
?>
```

Example 2. Find matching HTML tags (greedy)

```
<?php
// The \\2 is an example of backreferencing. This tells pcre that
// it must match the second set of parentheses in the regular expression
// itself, which would be the ([\w]+) in this case. The extra backslash is
// required because the string is in double quotes.
$html = "<b>bold text</b><a href=howdy.html>click me</a>";

preg_match_all ("/(<([\w]+)[^>]*>)(.*)(<\/\\2>)/", $html, $matches);

for ($i=0; $i< count($matches[0]); $i++) {
  echo "matched: ".$matches[0][$i]."\n";
  echo "part 1: ".$matches[1][$i]."\n";
  echo "part 2: ".$matches[3][$i]."\n";
  echo "part 3: ".$matches[4][$i]."\n\n";
}
?>
```

This example will produce:

```
matched: <b>bold text</b>
part 1: <b>
part 2: bold text
part 3: </b>

matched: <a href=howdy.html>click me</a>
part 1: <a href=howdy.html>
part 2: click me
part 3: </a>
```

See also **preg_match()**, **preg_replace()**, and **preg_split()**.

preg_match

(PHP 3>= 3.0.9, PHP 4)

preg_match -- Perform a regular expression match

Description

int **preg_match** (string pattern, string subject [, array matches [, int flags]])

Searches *subject* for a match to the regular expression given in *pattern*.

If *matches* is provided, then it is filled with the results of search. $matches[0] will contain the text that matched the full pattern, $matches[1] will have the text that matched the first captured parenthesized subpattern, and so on.

flags can be the following flag:

PREG_OFFSET_CAPTURE

> If this flag is set, for every occuring match the appendant string offset will also be returned. Note that this changes the return value in an array where every element is an array consisting of the matched string at offset 0 and it's string offset into *subject* at offset 1. This flag is available since PHP 4.3.0 .

The *flags* parameter is available since PHP 4.3.0 .

preg_match() returns the number of times *pattern* matches. That will be either 0 times (no match) or 1 time because **preg_match()** will stop searching after the first match. **preg_match_all()** on the contrary will continue until it reaches the end of *subject*. **preg_match()** returns **FALSE** if an error occured.

Example 1. Find the string of text "php"

```
// the "i" after the pattern delimiter indicates a case-insensitive search
if (preg_match ("/php/i", "PHP is the web scripting language of choice.")) {
    print "A match was found.";
} else {
    print "A match was not found.";
}
```

Example 2. find the word "web"

```
// the \b in the pattern indicates a word boundary, so only the distinct
// word "web" is matched, and not a word partial like "webbing" or "cobweb"
if (preg_match ("/\bweb\b/i", "PHP is the web scripting language of choice.")) {
    print "A match was found.";
} else {
    print "A match was not found.";
}
if (preg_match ("/\bweb\b/i", "PHP is the website scripting language of choice.")) {
    print "A match was found.";
} else {
    print "A match was not found.";
}
```

Example 3. Getting the domain name out of a URL

```
// get host name from URL
preg_match("/^(http:\/\/)?([^\/]+)/i",
"http://www.php.net/index.html", $matches);
$host = $matches[2];
// get last two segments of host name
preg_match("/[^\.\/]+\.[^\.\/]+$/",$host,$matches);
echo "domain name is: ".$matches[0]."\n";
```

This example will produce:

```
domain name is: php.net
```

See also **preg_match_all()**, **preg_replace()**, and **preg_split()**.

preg_quote

(PHP 3>= 3.0.9, PHP 4)

preg_quote -- Quote regular expression characters

Description

string **preg_quote** (string str [, string delimiter])

preg_quote() takes *str* and puts a backslash in front of every character that is part of the regular expression syntax. This is useful if you have a run-time string that you need to match in some text and the string may contain special regex characters.

If the optional *delimiter* is specified, it will also be escaped. This is useful for escaping the delimiter that is required by the PCRE functions. The / is the most commonly used delimiter.

The special regular expression characters are:

```
. \\ + * ? [ ^ ] $ ( ) { } = ! < > | :
```

Example 1.

```
$keywords = "$40 for a g3/400";
$keywords = preg_quote ($keywords, "/");
echo $keywords; // returns \$40 for a g3\/400
```

Example 2. Italicizing a word within some text

```
// In this example, preg_quote($word) is used to keep the
// asterisks from having special meaning to the regular
// expression.

$textbody = "This book is *very* difficult to find.";
$word = "*very*";
$textbody = preg_replace ("/".preg_quote($word)."/",
                          "<i>".$word."</i>",
                          $textbody);
```

preg_replace_callback

(PHP 4 >= 4.0.5)

preg_replace_callback -- Perform a regular expression search and replace using a callback

Description

mixed **preg_replace_callback** (mixed pattern, callback callback, mixed subject [, int limit])

The behavior of this function is almost identical to **preg_replace()**, except for the fact that instead of *replacement* parameter, one should specify a *callback* that will be called and passed an array of matched elements in the subject string. The callback should return the replacement string.

See also **preg_replace()**.

preg_replace

(PHP 3>= 3.0.9, PHP 4)

preg_replace -- Perform a regular expression search and replace

Description

mixed **preg_replace** (mixed pattern, mixed replacement, mixed subject [, int limit])

Searches *subject* for matches to *pattern* and replaces them with *replacement*. If *limit* is specified, then only *limit* matches will be replaced; if *limit* is omitted or is -1, then all matches are replaced.

Replacement may contain references of the form *\\n* or (since PHP 4.0.4) *$n*, with the latter form being the preferred one. Every such reference will be replaced by the text captured by the *n*'th parenthesized pattern. *n* can be from 0 to 99, and *\\0* or *$0* refers to the text matched by the whole pattern. Opening parentheses are counted from left to right (starting from 1) to obtain the number of the capturing subpattern.

> **Note:** When working with a replacement pattern where a backreference is immediately followed by another number

(i.e.: placing a literal number immediately after a matched pattern), you cannot use the familiar `\\1` notation for your backreference. `\\11`, for example, would confuse **preg_replace()** since it does not know whether you want the `\\1` backreference followed by a literal `1`, or the `\\11` backreference followed by nothing. In this case the solution is to use `\${1}1`. This creates an isolated `$1` backreference, leaving the `1` as a literal.

Example 1. Using backreferences followed by numeric literals.

```
<?php
$str = "April 15, 2003";
$pattern = "/(\w+) (\d+), (\d+)/i";
$replacement = "\${1}1,\$3";
print preg_replace($pattern, $replacement, $string);

/* Output
   ======

April1,2003

 */
?>
```

If matches are found, the new *subject* will be returned, otherwise *subject* will be returned unchanged.

Every parameter to **preg_replace()** (except *limit*) can be an array.

> **Note:** When using arrays with *pattern* and *replacement*, the keys are processed in the order they appear in the array. This is *not necessarily* the same as the numerical index order. If you use indexes to identify which *pattern* should be replaced by which *replacement*, you should perform a **ksort()** on each array prior to calling **preg_replace()**.

Example 2. Using indexed arrays with preg_replace()

```
<?php
$string = "The quick brown fox jumped over the lazy dog.";

$patterns[0] = "/quick/";
$patterns[1] = "/brown/";
$patterns[2] = "/fox/";

$replacements[2] = "bear";
$replacements[1] = "black";
$replacements[0] = "slow";

print preg_replace($patterns, $replacements, $string);

/* Output
   ======

The bear black slow jumped over the lazy dog.

*/

/* By ksorting patterns and replacements,
   we should get what we wanted. */

ksort($patterns);
ksort($replacements);

print preg_replace($patterns, $replacements, $string);

/* Output
   ======

The slow black bear jumped over the lazy dog.

*/

?>
```

If *subject* is an array, then the search and replace is performed on every entry of *subject*, and the return value is an array as well.

If *pattern* and *replacement* are arrays, then **preg_replace()** takes a value from each array and uses them to do search and replace on *subject*. If *replacement* has fewer values than *pattern*, then empty string is used for the rest of replacement values. If *pattern* is an array and *replacement* is a string, then this replacement string is used for every value of *pattern*. The converse would not make sense, though.

`/e` modifier makes **preg_replace()** treat the *replacement* parameter as PHP code after the appropriate references substitution is done. Tip: make sure that *replacement* constitutes a valid PHP code string, otherwise PHP will complain about a parse error at the line containing **preg_replace()**.

Example 3. Replacing several values

```
$patterns = array ("/(19|20)(\d{2})-(\d{1,2})-(\d{1,2})/",
                   "/^\s*{(\w+)}\s*=/");
$replace = array ("\\3/\\4/\\1\\2", "$\\1 =");
print preg_replace ($patterns, $replace, "{startDate} = 1999-5-27");
```

This example will produce:

```
$startDate = 5/27/1999
```

Example 4. Using /e modifier

```
preg_replace ("/(<\/?)(\w+)([^>]*>)/e",
              "'\\1'.strtoupper('\\2').'\\3'",
              $html_body);
```

This would capitalize all HTML tags in the input text.

Example 5. Convert HTML to text

```
// $document should contain an HTML document.
// This will remove HTML tags, javascript sections
// and white space. It will also convert some
// common HTML entities to their text equivalent.

$search = array ("'<script[^>]*?>.*?</script>'si",  // Strip out javascript
                 "'<[\/\!]*?[^<>]*?>'si",           // Strip out html tags
                 "'([\r\n])[\s]+'",                 // Strip out white space
                 "'&(quot|#34);'i",                 // Replace html entities
                 "'&(amp|#38);'i",
                 "'&(lt|#60);'i",
                 "'&(gt|#62);'i",
                 "'&(nbsp|#160);'i",
                 "'&(iexcl|#161);'i",
                 "'&(cent|#162);'i",
                 "'&(pound|#163);'i",
                 "'&(copy|#169);'i",
                 "'&#(\d+);'e");                    // evaluate as php

$replace = array ("",
                  "",
                  "\\1",
                  "\"",
                  "&",
                  "<",
                  ">",
                  " ",
                  chr(161),
                  chr(162),
                  chr(163),
                  chr(169),
                  "chr(\\1)");

$text = preg_replace ($search, $replace, $document);
```

> **Note:** Parameter *limit* was added after PHP 4.0.1pl2.

See also **preg_match()**, **preg_match_all()**, and **preg_split()**.

preg_split

(PHP 3>= 3.0.9, PHP 4)

preg_split -- Split string by a regular expression

Description

array **preg_split** (string pattern, string subject [, int limit [, int flags]])

> **Note:** Parameter *flags* was added in PHP 4 Beta 3.

Returns an array containing substrings of *subject* split along boundaries matched by *pattern*.

If *limit* is specified, then only substrings up to *limit* are returned, and if *limit* is -1, it actually means "no limit", which is useful for specifying the *flags*.

flags can be any combination of the following flags (combined with bitwise | operator):

PREG_SPLIT_NO_EMPTY

If this flag is set, only non-empty pieces will be returned by **preg_split()**.

PREG_SPLIT_DELIM_CAPTURE

If this flag is set, parenthesized expression in the delimiter pattern will be captured and returned as well. This flag was added for 4.0.5.

PREG_SPLIT_OFFSET_CAPTURE

If this flag is set, for every occuring match the appendant string offset will also be returned. Note that this changes the return value in an array where every element is an array consisting of the matched string at offset `0` and it's string offset into *`subject`* at offset `1`. This flag is available since PHP 4.3.0 .

Example 1. preg_split() example : Get the parts of a search string.

```
// split the phrase by any number of commas or space characters,
// which include " ", \r, \t, \n and \f
$keywords = preg_split ("/[\s,]+/", "hypertext language, programming");
```

Example 2. Splitting a string into component characters.

```
$str = 'string';
$chars = preg_split('//', $str, -1, PREG_SPLIT_NO_EMPTY);
print_r($chars);
```

Example 3. Splitting a string into matches and their offsets.

```
$str = 'hypertext language programming';
$chars = preg_split('/ /', $str, -1, PREG_SPLIT_OFFSET_CAPTURE);
print_r($chars);
```

will yield

```
Array
(
    [0] => Array
        (
            [0] => hypertext
            [1] => 0
        )

    [1] => Array
        (
            [0] => language
            [1] => 10
        )

    [2] => Array
        (
            [0] => programming
            [1] => 19
        )

)
```

See also **spliti()**, **split()**, **implode()**, **preg_match()**, **preg_match_all()**, and **preg_replace()**.

LXXXIX. qtdom functions

Warning
This extension is *EXPERIMENTAL*. The behaviour of this extension -- including the names of its functions and anything else documented about this extension -- may change without notice in a future release of PHP. Use this extension at your own risk.

Note: This extension is not available on Windows platforms.

Requirements

You need the Qt-library >=2.2.0

Installation

Configure PHP `--with-qtdom` to use these functions.

Table of Contents

qdom_error

(PHP 4 >= 4.0.5)

qdom_error -- Returns the error string from the last QDOM operation or FALSE if no errors occured

Description

string **qdom_error** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

qdom_tree

(PHP 4 >= 4.0.4)

qdom_tree -- creates a tree of an xml string

Description

object **qdom_tree** (string)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

XC. Regular Expression Functions (POSIX Extended)

Introduction

> **Note:** PHP also supports regular expressions using a Perl-compatible syntax using the PCRE functions. Those functions support non-greedy matching, assertions, conditional subpatterns, and a number of other features not supported by the POSIX-extended regular expression syntax.

Warning
These regular expression functions are not binary-safe. The PCRE functions are.

Regular expressions are used for complex string manipulation in PHP. The functions that support regular expressions are:

- **ereg()**
- **ereg_replace()**

- eregi()
- eregi_replace()
- split()
- spliti()

These functions all take a regular expression string as their first argument. PHP uses the POSIX extended regular expressions as defined by POSIX 1003.2. For a full description of POSIX regular expressions see the regex man pages included in the regex directory in the PHP distribution. It's in manpage format, so you'll want to do something along the lines of **man /usr/local/src/regex/regex.7** in order to read it.

Requirements

No external libraries are needed to build this extension.

Installation

To enable regexp support configure PHP `--with-regex[=TYPE]`. TYPE can be one of system, apache, php. The default is to use php.

> **Note:** Do not change the TYPE unless you know what you are doing.

The windows version of PHP has built in support for this extension. You do not need to load any additional extension in order to use these functions.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Examples

Example 1. Regular Expression Examples

```
ereg ("abc", $string);
/* Returns true if"abc"
   is found anywhere in $string. */

ereg ("^abc", $string);
/* Returns true if "abc";
   is found at the beginning of $string. */

ereg ("abc$", $string);
/* Returns true if "abc"
```

```
   is found at the end of $string. */

eregi ("(ozilla.[23]|MSIE.3)", $HTTP_USER_AGENT);
/* Returns true if client browser
   is Netscape 2, 3 or MSIE 3. */

ereg ("([[:alnum:]]+) ([[:alnum:]]+) ([[:alnum:]]+)", $string,$regs);
/* Places three space separated words
   into $regs[1], $regs[2] and $regs[3]. */

$string = ereg_replace ("^", "<br />", $string);
/* Put a <br /> tag at the beginning of $string. */

$string = ereg_replace ("$", "<br />", $string);
/* Put a <br />; tag at the end of $string. */

$string = ereg_replace ("\n", "", $string);
/* Get rid of any newline
   characters in $string. */
```

See Also

For regular expressions in Perl-compatible syntax have a look at the PCRE functions. The simpler shell style wildcard pattern matching is provided by **fnmatch()**.

Table of Contents

ereg_replace

(PHP 3, PHP 4)

ereg_replace -- Replace regular expression

Description

string **ereg_replace** (string pattern, string replacement, string string)

> **Note: preg_replace()**, which uses a Perl-compatible regular expression syntax, is often a faster alternative to **ereg_replace()**.

This function scans *string* for matches to *pattern*, then replaces the matched text with *replacement*.

The modified string is returned. (Which may mean that the original string is returned if there are no matches to be replaced.)

If *pattern* contains parenthesized substrings, *replacement* may contain substrings of the form `\\digit`, which will be replaced by the text matching the digit'th parenthesized substring; `\\0` will produce the entire contents of string. Up to nine substrings may be used. Parentheses may be nested, in which case they are counted by the opening parenthesis.

If no matches are found in *string*, then *string* will be returned unchanged.

For example, the following code snippet prints "This was a test" three times:

Example 1. ereg_replace() Example

```
$string = "This is a test";
echo ereg_replace (" is", " was", $string);
echo ereg_replace ("( )is", "\\1was", $string);
echo ereg_replace ("(( )is)", "\\2was", $string);
```

One thing to take note of is that if you use an integer value as the *replacement* parameter, you may not get the results you expect. This is because **ereg_replace()** will interpret the number as the ordinal value of a character, and apply that. For instance:

Example 2. ereg_replace() Example

```
<?php
/* This will not work as expected. */
$num = 4;
$string = "This string has four words.";
$string = ereg_replace('four', $num, $string);
echo $string;   /* Output: 'This string has   words.' */

/* This will work. */
$num = '4';
$string = "This string has four words.";
$string = ereg_replace('four', $num, $string);
echo $string;   /* Output: 'This string has 4 words.' */
?>
```

Example 3. Replace URLs with links

```
$text = ereg_replace("[[:alpha:]]+://[^<>[:space:]]+[[:alnum:]/]",
                     "<a href=\"\\0\">\\0</a>", $text);
```

See also **ereg()**, **eregi()**, **eregi_replace()**, **str_replace()**, and **preg_match()**.

ereg

(PHP 3, PHP 4)

ereg -- Regular expression match

Description

int **ereg** (string pattern, string string [, array regs])

> **Note:** **preg_match()**, which uses a Perl-compatible regular expression syntax, is often a faster alternative to **ereg()**.

Searches a *string* for matches to the regular expression given in *pattern*.

If matches are found for parenthesized substrings of *pattern* and the function is called with the third argument *regs*, the matches will be stored in the elements of the array *regs*. $regs[1] will contain the substring which starts at the first left parenthesis; $regs[2] will contain the substring starting at the second, and so on. $regs[0] will contain a copy of the complete string matched.

> **Note:** Up to (and including) PHP 4.1.0 `$regs` will be filled with exactly ten elements, even though more or fewer than ten parenthesized substrings may actually have matched. This has no effect on **ereg()**'s ability to match more substrings. If no matches are found, `$regs` will not be altered by **ereg()**.

Searching is case sensitive.

Returns **TRUE** if a match for *pattern* was found in *string*, or **FALSE** if no matches were found or an error occurred.

The following code snippet takes a date in ISO format (YYYY-MM-DD) and prints it in DD.MM.YYYY format:

Example 1. ereg() Example

```
if (ereg ("([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})", $date, $regs)) {
    echo "$regs[3].$regs[2].$regs[1]";
} else {
    echo "Invalid date format: $date";
}
```

See also **eregi()**, **ereg_replace()**, **eregi_replace()**, and **preg_match()**.

eregi_replace

(PHP 3, PHP 4)

eregi_replace -- replace regular expression case insensitive

Description

string **eregi_replace** (string pattern, string replacement, string string)

This function is identical to **ereg_replace()** except that this ignores case distinction when matching alphabetic characters.

See also **ereg()**, **eregi()**, and **ereg_replace()**.

eregi

(PHP 3, PHP 4)

eregi -- case insensitive regular expression match

Description

int **eregi** (string pattern, string string [, array regs])

This function is identical to **ereg()** except that this ignores case distinction when matching alphabetic characters.

Example 1. eregi() example

```
if (eregi("z", $string)) {
    echo "'$string' contains a 'z' or 'Z'!";
}
```

See also **ereg()**, **ereg_replace()**, and **eregi_replace()**.

split

(PHP 3, PHP 4)

split -- split string into array by regular expression

Description

array **split** (string pattern, string string [, int limit])

> **Note:** **preg_split()**, which uses a Perl-compatible regular expression syntax, is often a faster alternative to **split()**.

Returns an array of strings, each of which is a substring of *string* formed by splitting it on boundaries formed by the regular expression *pattern*. If *limit* is set, the returned array will contain a maximum of *limit* elements with the last element containing the whole rest of *string*. If an error occurs, **split()** returns **FALSE**.

To split off the first four fields from a line from `/etc/passwd`:

Example 1. split() Example

```
list($user,$pass,$uid,$gid,$extra)= split (":", $passwd_line, 5);
```

> **Note:** If there are *n* occurrences of *pattern*, the returned array will contain *n*+1 items. For example, if there is no occurrence of *pattern*, an array with only one element will be returned. Of course, this is also true if *string* is empty.

To parse a date which may be delimited with slashes, dots, or hyphens:

Example 2. split() Example

```
$date = "04/30/1973";  // Delimiters may be slash, dot, or hyphen
list ($month, $day, $year) = split ('[/.-]', $date);
echo "Month: $month; Day: $day; Year: $year<br>\n";
```

Note that *pattern* is case-sensitive.

Note that if you don't require the power of regular expressions, it is faster to use **explode()**, which doesn't incur the overhead of the regular expression engine.

For users looking for a way to emulate Perl's **@chars = split('', $str)** behaviour, please see the examples for **preg_split()**.

Please note that *pattern* is a regular expression. If you want to split on any of the characters which are considered special by regular expressions, you'll need to escape them first. If you think **split()** (or any other regex function, for that matter) is doing something weird, please read the file `regex.7`, included in the `regex/` subdirectory of the PHP distribution. It's in manpage

format, so you'll want to do something along the lines of **man /usr/local/src/regex/regex.7** in order to read it.

See also: **preg_split()**, **spliti()**, **explode()**, **implode()**, **chunk_split()**, and **wordwrap()**.

spliti

(PHP 4 >= 4.0.1)

spliti -- Split string into array by regular expression case insensitive

Description

array **spliti** (string pattern, string string [, int limit])

This function is identical to **split()** except that this ignores case distinction when matching alphabetic characters.

See also **preg_spliti()**, **split()**, **explode()**, and **implode()**.

sql_regcase

(PHP 3, PHP 4)

sql_regcase -- Make regular expression for case insensitive match

Description

string **sql_regcase** (string string)

Returns a valid regular expression which will match *string*, ignoring case. This expression is *string* with each character converted to a bracket expression; this bracket expression contains that character's uppercase and lowercase form if applicable, otherwise it contains the original character twice.

Example 1. sql_regcase() Example

```
echo sql_regcase ("Foo bar");
```

prints

```
[Ff][Oo][Oo] [Bb][Aa][Rr]
```

.

This can be used to achieve case insensitive pattern matching in products which support only case sensitive regular expressions.

XCI. Semaphore, Shared Memory and IPC Functions

Introduction

This module provides wrappers for the System V IPC family of functions. It includes semaphores, shared memory and inter-process messaging (IPC).

Semaphores may be used to provide exclusive access to resources on the current machine, or to limit the number of processes that may simultaneously use a resource.

This module provides also shared memory functions using System V shared memory. Shared memory may be used to provide access to global variables. Different httpd-daemons and even other programs (such as Perl, C, ...) are able to access this data to provide a global data-exchange. Remember, that shared memory is NOT safe against simultaneous access. Use semaphores for synchronization.

Table 1. Limits of Shared Memory by the Unix OS

SHMMAX	max size of shared memory, normally 131072 bytes
SHMMIN	minimum size of shared memory, normally 1 byte
SHMMNI	max amount of shared memory segments on a system, normally 100
SHMSEG	max amount of shared memory segments per process, normally 6

The messaging functions may be used to send and receive messages to/from other processes. They provide a simple and effective means of exchanging data between processes, without the need for setting up an alternative using unix domain sockets.

Note: This extension is not available on Windows platforms.

Requirements

No external libraries are needed to build this extension.

Installation

Support for this functions are not enabled by default. To enable System V semaphore support compile PHP with the option `--enable-sysvsem`. To enable the System V shared memory support compile PHP with the option `--enable-sysvshm`. To enable the System V messages support compile PHP with the option `--enable-sysvmsg`.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 2. Semaphore Configuration Options

Name	Default	Changeable
sysvmsg.value	"42"	PHP_INI_ALL
sysvmsg.string	"foobar"	PHP_INI_ALL

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Resource Types

Predefined Constants

This extension has no constants defined.

Table of Contents

shm_get_var -- Returns a variable from shared memory
shm_put_var -- Inserts or updates a variable in shared memory
shm_remove_var -- Removes a variable from shared memory
shm_remove -- Removes shared memory from Unix systems

ftok

(PHP 4 >= 4.2.0)

ftok -- Convert a pathname and a project identifier to a System V IPC key

Description

int **ftok** (string pathname, string proj)

Warning
This function is currently not documented; only the argument list is available.

msg_get_queue

(PHP 4 >= 4.3.0)

msg_get_queue -- Create or attach to a message queue

Description

int **msg_get_queue** (int key [, int perms])

msg_get_queue() returns an id that can be used to access the System V message queue with the given *key*. The first call creates the message queue with the optional *perms* (default: 0666). A second call to **msg_get_queue()** for the same *key* will return a different message queue identifier, but both identifiers access the same underlying message queue. If the message queue already exists, the *perms* will be ignored.

See also: **msg_remove_queue()**, **msg_receive()**, **msg_send()**, **msg_stat_queue()** and **msg_set_queue()**.

This function was introduced in PHP 4.3.0 (not yet released).

msg_receive

(PHP 4 >= 4.3.0)

msg_receive -- Receive a message from a message queue

Description

bool **msg_receive** (int queue, int desiredmsgtype, int msgtype, int maxsize, mixed message [, bool unserialize [, int flags [, int errorcode]]])

msg_receive() will receive the first message from the specified *queue* of the type specified by *desiredmsgtype*. The type of the message that was received will be stored in *msgtype*. The maximum size of message to be accepted is specified by the *maxsize*; if the message in the queue is larger than this size the function will fail (unless you set *flags* as described below). The received message will be stored in *message*, unless there were errors receiving the message, in which case the optional *errorcode* will be set to the value of the system errno variable to help you identify the cause.

If *desiredmsgtype* is 0, the message from the front of the queue is returned. If *desiredmsgtype* is greater than 0, then the first message of that type is returned. If *desiredmsgtype* is less than 0, the first message on the queue with the lowest type less than or equal to the absolute value of *desiredmsgtype* will be read. If no messages match the criteria, your script will wait until a suitable message arrives on the queue. You can prevent the script from blocking by specifying MSG_IPC_NOWAIT in the *flags* parameter.

unserialize defaults to **TRUE**; if it is set to **TRUE**, the message is treated as though it was serialized using the same mechanism as the session module. The message will be unserialized and then returned to your script. This allows you to easily receive arrays or

complex object structures from other PHP scripts, or if you are using the WDDX serializer, from any WDDX compatible source. If *unserialize* is **FALSE**, the message will be returned as a binary-safe string.

The optional *flags* allows you to pass flags to the low-level msgrcv system call. It defaults to 0, but you may specify one or more of the following values (by adding or ORing them together).

Table 1. Flag values for msg_receive

MSG_IPC_NOWAIT	If there are no messages of the *desiredmsgtype*, return immediately and do not wait. The function will fail and return an integer value corresponding to ENOMSG.
MSG_EXCEPT	Using this flag in combination with a *desiredmsgtype* greater than 0 will cause the function to receive the first message that is not equal to *desiredmsgtype*.
MSG_NOERROR	If the message is longer than *maxsize*, setting this flag will truncate the message to *maxsize* and will not signal an error.

Upon successful completion the message queue data structure is updated as follows: *msg_lrpid* is set to the process-ID of the calling process, *msg_qnum* is decremented by 1 and *msg_rtime* is set to the current time.

msg_receive() returns **TRUE** on success or **FALSE** on failure. If the function fails, the optional *errorcode* will be set to the value of the system errno variable.

See also: **msg_remove_queue()**, **msg_send()**, **msg_stat_queue()** and **msg_set_queue()**.

This function was introduced in PHP 4.3.0 (not yet released).

msg_remove_queue

(PHP 4 >= 4.3.0)

msg_remove_queue -- Destroy a message queue

Description

bool **msg_remove_queue** (int queue)

msg_remove_queue() destroys the message queue specified by the *queue*. Only use this function when all processes have finished working with the message queue and you need to release the system resources held by it.

See also: **msg_remove_queue()**, **msg_receive()**, **msg_stat_queue()** and **msg_set_queue()**.

This function was introduced in PHP 4.3.0 (not yet released).

msg_send

(PHP 4 >= 4.3.0)

msg_send -- Send a message to a message queue

Description

bool **msg_send** (int queue, int msgtype, mixed message [, bool serialize [, bool blocking [, int errorcode]]])

msg_send() sends a *message* of type *msgtype* (which MUST be greater than 0) to a the message queue specified by *queue*.

If the message is too large to fit in the queue, your script will wait until another process reads messages from the queue and frees enough space for your message to be sent. This is called blocking; you can prevent blocking by setting the optional *blocking* parameter to **FALSE**, in which case **msg_send()** will immediately return **FALSE** if the message is too big for the queue, and set the optional *errorcode* to EAGAIN, indicating that you should try to send your message again a little later on.

The optional *serialize* controls how the *message* is sent. *serialize* defaults to **TRUE** which means that the *message* is serialized using the same mechanism as the session module before being sent to the queue. This allows complex arrays and objects to be sent to other PHP scripts, or if you are using the WDDX serializer, to any WDDX compatible client.

Upon successful completion the message queue data structure is updated as follows: *msg_lspid* is set to the process-ID of the calling process, *msg_qnum* is incremented by 1 and *msg_stime* is set to the current time.

See also: **msg_remove_queue()**, **msg_receive()**, **msg_stat_queue()** and **msg_set_queue()**.

This function was introduced in PHP 4.3.0 (not yet released).

msg_set_queue

(PHP 4 >= 4.3.0)

msg_set_queue -- Set information in the message queue data structure

Description

bool **msg_set_queue** (int queue, array data)

msg_set_queue() allows you to change the values of the msg_perm.uid, msg_perm.gid, msg_perm.mode and msg_qbytes fields of the underlying message queue data structure. You specify the values you require by setting the value of the keys that you require in the *data* array.

Changing the data structure will require that PHP be running as the same user that created the the queue, owns the queue (as determined by the existing msg_perm.xxx fields), or be running with root privileges. root privileges are required to raise the msg_qbytes values above the system defined limit.

See also: **msg_remove_queue()**, **msg_receive()**, **msg_stat_queue()** and **msg_set_queue()**.

This function was introduced in PHP 4.3.0 (not yet released).

msg_stat_queue

(PHP 4 >= 4.3.0)

msg_stat_queue -- Returns information from the message queue data structure

Description

array **msg_stat_queue** (int queue)

msg_stat_queue() returns the message queue meta data for the message queue specified by the *queue*. This is useful, for example, to determine which process sent the message that was just received.

The return value is an array whose keys and values have the following meanings:

Table 1. Array structure for msg_stat_queue

msg_perm.uid	The uid of the owner of the queue.
msg_perm.gid	The gid of the owner of the queue.
msg_perm.mode	The file access mode of the queue.
msg_stime	The time that the last message was sent to the queue.
msg_rtime	The time that the last message was received from the queue.
msg_ctime	The time that the queue was last changed.
msg_qnum	The number of messages waiting to be read from the queue.
msg_qbytes	The number of bytes of space currently available in the queue to hold sent messages until they are received.
msg_lspid	The pid of the process that sent the last message to the queue.
msg_lrpid	The pid of the process that received the last message from the queue.

See also: **msg_remove_queue()**, **msg_receive()**, **msg_stat_queue()** and **msg_set_queue()**.

This function was introduced in PHP 4.3.0 (not yet released).

sem_acquire

(PHP 3>= 3.0.6, PHP 4)

sem_acquire -- Acquire a semaphore

Description

bool **sem_acquire** (int sem_identifier)

Returns: **TRUE** on success, **FALSE** on error.

sem_acquire() blocks (if necessary) until the semaphore can be acquired. A process attempting to acquire a semaphore which it has already acquired will block forever if acquiring the semaphore would cause its max_acquire value to be exceeded.

After processing a request, any semaphores acquired by the process but not explicitly released will be released automatically and a warning will be generated.

See also: **sem_get()** and **sem_release()**.

sem_get

(PHP 3>= 3.0.6, PHP 4)

sem_get -- Get a semaphore id

Description

int **sem_get** (int key [, int max_acquire [, int perm]])

Returns: A positive semaphore identifier on success, or **FALSE** on error.

sem_get() returns an id that can be used to access the System V semaphore with the given key. The semaphore is created if necessary using the permission bits specified in perm (defaults to 0666). The number of processes that can acquire the semaphore simultaneously is set to max_acquire (defaults to 1). Actually this value is set only if the process finds it is the only process currently attached to the semaphore.

A second call to **sem_get()** for the same key will return a different semaphore identifier, but both identifiers access the same underlying semaphore.

See also: **sem_acquire()**, **sem_release()** and **ftok()**.

> **Note:** This function does not work on Windows systems.

sem_release

(PHP 3>= 3.0.6, PHP 4)

sem_release -- Release a semaphore

Description

bool **sem_release** (int sem_identifier)

Returns: **TRUE** on success, **FALSE** on error.

sem_release() releases the semaphore if it is currently acquired by the calling process, otherwise a warning is generated.

After releasing the semaphore, **sem_acquire()** may be called to re-acquire it.

See also: **sem_get()** and **sem_acquire()**.

> **Note:** This function does not work on Windows systems.

sem_remove

(PHP 4 >= 4.1.0)

sem_remove -- Remove a semaphore

Description

bool **sem_remove** (int sem_identifier)

Returns: **TRUE** on success, **FALSE** on error.

sem_remove() removes the semaphore *sem_identifier* if it has been created by **sem_get()**, otherwise generates a warning.

After removing the semaphore, it is no more accessible.

See also: **sem_get()**, **sem_release()** and **sem_acquire()**.

> **Note:** This function does not work on Windows systems. It was added on PHP 4.1.0.

shm_attach

(PHP 3>= 3.0.6, PHP 4)

shm_attach -- Creates or open a shared memory segment

Description

int **shm_attach** (int key [, int memsize [, int perm]])

shm_attach() returns an id that that can be used to access the System V shared memory with the given key, the first call creates the shared memory segment with mem_size (default: sysvshm.init_mem in the configuration file, otherwise 10000 bytes) and the optional perm-bits (default: 0666).

A second call to **shm_attach()** for the same *key* will return a different shared memory identifier, but both identifiers access the same underlying shared memory. *Memsize* and *perm* will be ignored.

See also: **ftok()**.

> **Note:** This function does not work on Windows systems.

shm_detach

(PHP 3>= 3.0.6, PHP 4)

shm_detach -- Disconnects from shared memory segment

Description

bool **shm_detach** (int shm_identifier)

shm_detach() disconnects from the shared memory given by the *shm_identifier* created by **shm_attach()**. Remember, that shared memory still exist in the Unix system and the data is still present.

shm_detach() always returns **TRUE**.

shm_get_var

(PHP 3>= 3.0.6, PHP 4)

shm_get_var -- Returns a variable from shared memory

Description

mixed **shm_get_var** (int id, int variable_key)

shm_get_var() returns the variable with a given *variable_key*. The variable is still present in the shared memory.

Note: This function does not work on Windows systems.

shm_put_var

(PHP 3>= 3.0.6, PHP 4)

shm_put_var -- Inserts or updates a variable in shared memory

Description

int **shm_put_var** (int shm_identifier, int variable_key, mixed variable)

Inserts or updates a *variable* with a given *variable_key*. All variable-types are supported.

Note: This function does not work on Windows systems.

shm_remove_var

(PHP 3>= 3.0.6, PHP 4)

shm_remove_var -- Removes a variable from shared memory

Description

int **shm_remove_var** (int id, int variable_key)

Removes a variable with a given *variable_key* and frees the occupied memory.

Note: This function does not work on Windows systems.

shm_remove

(PHP 3>= 3.0.6, PHP 4)

shm_remove -- Removes shared memory from Unix systems

Description

int **shm_remove** (int shm_identifier)

Removes shared memory from Unix systems. All data will be destroyed.

Note: This function does not work on Windows systems.

XCII. SESAM database functions

Introduction

SESAM/SQL-Server is a mainframe database system, developed by Fujitsu Siemens Computers, Germany. It runs on high-end mainframe servers using the operating system BS2000/OSD.

In numerous productive BS2000 installations, SESAM/SQL-Server has proven

- the ease of use of Java-, Web- and client/server connectivity,
- the capability to work with an availability of more than 99.99%,
- the ability to manage tens and even hundreds of thousands of users.

There is a PHP3 SESAM interface available which allows database operations via PHP-scripts.

Note: Access to SESAM is only available with the latest CVS-Version of PHP3. PHP 4 does not support the SESAM database.

Runtime Configuration

The behaviour of these functions is affected by settings in php.ini.

sesam_oml **string**

Name of BS2000 PLAM library containing the loadable SESAM driver modules. Required for using SESAM functions. The BS2000 PLAM library must be set ACCESS=READ,SHARE=YES because it must be readable by the apache server's user id.

sesam_configfile **string**

Name of SESAM application configuration file. Required for using SESAM functions. The BS2000 file must be readable by the apache server's user id.

The application configuration file will usually contain a configuration like (see SESAM reference manual):

```
CNF=B
NAM=K
NOTYPE
```

sesam_messagecatalog **string**

Name of SESAM message catalog file. In most cases, this directive is not neccessary. Only if the SESAM message file is not installed in the system's BS2000 message file table, it can be set with this directive.

The message catalog must be set ACCESS=READ,SHARE=YES because it must be readable by the apache server's user id.

Configuration notes

There is no standalone support for the PHP SESAM interface, it works only as an integrated Apache module. In the Apache PHP module, this SESAM interface is configured using Apache directives.

Table 1. SESAM Configuration directives

Directive	Meaning
php3_sesam_oml	Name of BS2000 PLAM library containing the loadable SESAM driver modules. Required for using SESAM functions. Example: `php3_sesam_oml $.SYSLNK.SESAM-SQL.030`

php3_sesam_configfile	Name of SESAM application configuration file. Required for using SESAM functions.
	
	Example:
	
	`php3_sesam_configfile $SESAM.SESAM.CONF.AW`
	
	It will usually contain a configuration like (see SESAM reference manual):
	
	`CNF=B`
	`NAM=K`
	`NOTYPE`

In addition to the configuration of the PHP/SESAM interface, you have to configure the SESAM-Database server itself on your mainframe as usual. That means:

- starting the SESAM database handler (DBH), and
- connecting the databases with the SESAM database handler

To get a connection between a PHP script and the database handler, the CNF and NAM parameters of the selected SESAM configuration file must match the id of the started database handler.

In case of distributed databases you have to start a SESAM/SQL-DCN agent with the distribution table including the host and database names.

The communication between PHP (running in the POSIX subsystem) and the database handler (running outside the POSIX subsystem) is realized by a special driver module called SQLSCI and SESAM connection modules using common memory. Because of the common memory access, and because PHP is a static part of the web server, database accesses are very fast, as they do not require remote accesses via ODBC, JDBC or UTM.

Only a small stub loader (SESMOD) is linked with PHP, and the SESAM connection modules are pulled in from SESAM's OML PLAM library. In the configuration, you must tell PHP the name of this PLAM library, and the file link to use for the SESAM configuration file (As of SESAM V3.0, SQLSCI is available in the SESAM Tool Library, which is part of the standard distribution).

Because the SQL command quoting for single quotes uses duplicated single quotes (as opposed to a single quote preceded by a backslash, used in some other databases), it is advisable to set the PHP configuration directives php3_magic_quotes_gpc and php3_magic_quotes_sybase to On for all PHP scripts using the SESAM interface.

Runtime considerations

Because of limitations of the BS2000 process model, the driver can be loaded only after the Apache server has forked off its server child processes. This will slightly slow down the initial SESAM request of each child, but subsequent accesses will respond at full speed.

When explicitly defining a Message Catalog for SESAM, that catalog will be loaded each time the driver is loaded (i.e., at the initial SESAM request). The BS2000 operating system prints a message after successful load of the message catalog, which will be sent to Apache's error_log file. BS2000 currently does not allow suppression of this message, it will slowly fill up the log.

Make sure that the SESAM OML PLAM library and SESAM configuration file are readable by the user id running the web server. Otherwise, the server will be unable to load the driver, and will not allow to call any SESAM functions. Also, access to the database must be granted to the user id under which the Apache server is running. Otherwise, connections to the SESAM database handler will fail.

Cursor Types

The result cursors which are allocated for SQL "select type" queries can be either "sequential" or "scrollable". Because of the larger memory overhead needed by "scrollable" cursors, the default is "sequential".

When using "scrollable" cursors, the cursor can be freely positioned on the result set. For each "scrollable" query, there are global default values for the scrolling type (initialized to: SESAM_SEEK_NEXT) and the scrolling offset which can either be set once by **sesam_seek_row()** or each time when fetching a row using **sesam_fetch_row()**. When fetching a row using a "scrollable" cursor, the following post-processing is done for the global default values for the scrolling type and scrolling offset:

Table 2. Scrolled Cursor Post-Processing

Scroll Type	Action
SESAM_SEEK_NEXT	none
SESAM_SEEK_PRIOR	none
SESAM_SEEK_FIRST	set scroll type to SESAM_SEEK_NEXT
SESAM_SEEK_LAST	set scroll type to SESAM_SEEK_PRIOR
SESAM_SEEK_ABSOLUTE	Auto-Increment internal offset value
SESAM_SEEK_RELATIVE	none. (maintain global default *offset* value, which allows for, e.g., fetching each 10th row backwards)

Porting note

Because in the PHP world it is natural to start indexes at zero (rather than 1), some adaptions have been made to the SESAM interface: whenever an indexed array is starting with index 1 in the native SESAM interface, the PHP interface uses index 0 as a starting point. E.g., when retrieving columns with **sesam_fetch_row()**, the first column has the index 0, and the subsequent

columns have indexes up to (but not including) the column count ($array["count"]). When porting SESAM applications from other high level languages to PHP, be aware of this changed interface. Where appropriate, the description of the respective php sesam functions include a note that the index is zero based.

Security concerns

When allowing access to the SESAM databases, the web server user should only have as little privileges as possible. For most databases, only read access privilege should be granted. Depending on your usage scenario, add more access rights as you see fit. Never allow full control to any database for any user from the 'net! Restrict access to php scripts which must administer the database by using password control and/or SSL security.

Migration from other SQL databases

No two SQL dialects are ever 100% compatible. When porting SQL applications from other database interfaces to SESAM, some adaption may be required. The following typical differences should be noted:

- Vendor specific data types

  Some vendor specific data types may have to be replaced by standard SQL data types (e.g., `TEXT` could be replaced by `VARCHAR(max. size)`).

- Keywords as SQL identifiers

  In SESAM (as in standard SQL), such identifiers must be enclosed in double quotes (or renamed).

- Display length in data types

  SESAM data types have a precision, not a display length. Instead of `int(4)` (intended use: integers up to '9999'), SESAM requires simply `int` for an implied size of 31 bits. Also, the only datetime data types available in SESAM are: `DATE`, `TIME(3)` and `TIMESTAMP(3)`.

- SQL types with vendor-specific `unsigned`, `zerofill`, or `auto_increment` attributes

  `Unsigned` and `zerofill` are not supported. `Auto_increment` is automatic (use `"INSERT ... VALUES(*, ...)"` instead of `"... VALUES(0, ...)"` to take advantage of SESAM-implied auto-increment.

- **int ... DEFAULT 'oooo'**

  Numeric variables must not be initialized with string constants. Use **DEFAULT o** instead. To initialize variables of the datetime SQL data types, the initialization string must be prefixed with the respective type keyword, as in: `CREATE TABLE exmpl ( xtime timestamp(3) DEFAULT TIMESTAMP '1970-01-01 00:00:00.000' NOT NULL );`

- **$count = xxxx_num_rows();**

  Some databases promise to guess/estimate the number of the rows in a query result, even though the returned value is grossly incorrect. SESAM does not know the number of rows in a query result before actually fetching them. If you REALLY need the count, try **SELECT COUNT(...) WHERE ...**, it will tell you the number of hits. A second query will (hopefully) return the results.

- **DROP TABLE thename;**

  In SESAM, in the **DROP TABLE** command, the table name must be either followed by the keyword `RESTRICT` or `CASCADE`. When specifying `RESTRICT`, an error is returned if there are dependent objects (e.g., VIEWs), while with `CASCADE`, dependent objects will be deleted along with the specified table.

Notes on the use of various SQL types

SESAM does not currently support the BLOB type. A future version of SESAM will have support for BLOB.

At the PHP interface, the following type conversions are automatically applied when retrieving SQL fields:

Table 3. SQL to PHP Type Conversions

SQL Type	PHP Type
SMALLINT, INTEGER	**integer**
NUMERIC, DECIMAL, FLOAT, REAL, DOUBLE	**float**
DATE, TIME, TIMESTAMP	**string**
VARCHAR, CHARACTER	**string**

When retrieving a complete row, the result is returned as an array. Empty fields are not filled in, so you will have to check for the existence of the individual fields yourself (use **isset()** or **empty()** to test for empty fields). That allows more user control over the appearance of empty fields (than in the case of an empty string as the representation of an empty field).

Support of SESAM's "multiple fields" feature

The special "multiple fields" feature of SESAM allows a column to consist of an array of fields. Such a "multiple field" column can be created like this:

Example 1. Creating a "multiple field" column

```
CREATE TABLE multi_field_test (
    pkey CHAR(20) PRIMARY KEY,
    multi(3) CHAR(12)
)
```

and can be filled in using:

Example 2. Filling a "multiple field" column

```
INSERT INTO multi_field_test (pkey, multi(2..3) )
    VALUES ('Second', <'first_val', 'second_val'>)
```

Note that (like in this case) leading empty sub-fields are ignored, and the filled-in values are collapsed, so that in the above example the result will appear as multi(1..2) instead of multi(2..3).

When retrieving a result row, "multiple columns" are accessed like "inlined" additional columns. In the example above, "pkey" will have the index 0, and the three "multi(1..3)" columns will be accessible as indices 1 through 3.

See Also

For specific SESAM details, please refer to the SESAM/SQL-Server documentation (english) or the SESAM/SQL-Server documentation (german), both available online, or use the respective manuals.

Table of Contents

sesam_affected_rows

(PHP 3 CVS only)

sesam_affected_rows -- Get number of rows affected by an immediate query

Description

int **sesam_affected_rows** (string result_id)

result_id is a valid result id returned by **sesam_query()**.

Returns the number of rows affected by a query associated with *result_id*.

The **sesam_affected_rows()** function can only return useful values when used in combination with "immediate" SQL statements (updating operations like INSERT, UPDATE and DELETE) because SESAM does not deliver any "affected rows" information for "select type" queries. The number returned is the number of affected rows.

See also: **sesam_query()** and **sesam_execimm()**

```
$result = sesam_execimm ("DELETE FROM PHONE WHERE LASTNAME = '".strtoupper ($name)."'");
if (!$result) {
    ... error ...
}
print sesam_affected_rows ($result).
    " entries with last name ".$name." deleted.\n"
```

sesam_commit

(PHP 3 CVS only)

sesam_commit -- Commit pending updates to the SESAM database

Description

bool **sesam_commit** (void)

Returns: **TRUE** on success, **FALSE** on errors

sesam_commit() commits any pending updates to the database.

Note that there is no "auto-commit" feature as in other databases, as it could lead to accidental data loss. Uncommitted data at the end of the current script (or when calling **sesam_disconnect()**) will be discarded by an implied **sesam_rollback()** call.

See also: **sesam_rollback()**.

Example 1. Committing an update to the SESAM database

```
<?php
if (sesam_connect ("mycatalog", "myschema", "otto")) {
    if (!sesam_execimm ("INSERT INTO mytable VALUES (*, 'Small Test', <0, 8, 15>)"))
        die("insert failed");
    if (!sesam_commit())
        die("commit failed");
}
?>
```

sesam_connect

(PHP 3 CVS only)

sesam_connect -- Open SESAM database connection

Description

bool **sesam_connect** (string catalog, string schema, string user)

Returns **TRUE** if a connection to the SESAM database was made, or **FALSE** on error.

sesam_connect() establishes a connection to an SESAM database handler task. The connection is always "persistent" in the sense that only the very first invocation will actually load the driver from the configured SESAM OML PLAM library. Subsequent calls will reuse the driver and will immediately use the given catalog, schema, and user.

When creating a database, the *"catalog"* name is specified in the SESAM configuration directive

//ADD-SQL-DATABASE-CATALOG-LIST ENTRY-1 = *CATALOG(CATALOG-NAME = catalogname,...)

The *"schema"* references the desired database schema (see SESAM handbook).

The *"user"* argument references one of the users which are allowed to access this *"catalog"* / *"schema"* combination. Note that *"user"* is completely independent from both the system's user id's and from HTTP user/password protection. It appears in the SESAM configuration only.

See also **sesam_disconnect()**.

Example 1. Connect to a SESAM database

```
<?php
if (!sesam_connect ("mycatalog", "myschema", "otto")
    die("Unable to connect to SESAM");
?>
```

sesam_diagnostic

(PHP 3 CVS only)

sesam_diagnostic -- Return status information for last SESAM call

Description

array **sesam_diagnostic** (void)

Returns an associative array of status and return codes for the last SQL query/statement/command. Elements of the array are:

Table 1. Status information returned by sesam_diagnostic()

Element	Contents
$array["sqlstate"]	5 digit SQL return code (see the SESAM manual for the description of the possible values of SQLSTATE)
$array["rowcount"]	number of affected rows in last update/insert/delete (set after "immediate" statements only)
$array["errmsg"]	"human readable" error message string (set after errors only)
$array["errcol"]	error column number of previous error (0-based; or -1 if undefined. Set after errors only)
$array["errlin"]	error line number of previous error (0-based; or -1 if undefined. Set after errors only)

In the following example, a syntax error (E SEW42AE ILLEGAL CHARACTER) is displayed by including the offending SQL statement and pointing to the error location:

Example 1. Displaying SESAM error messages with error position

```
<?php
// Function which prints a formatted error message,
// displaying a pointer to the syntax error in the
// SQL statement
function PrintReturncode ($exec_str) {
    $err = Sesam_Diagnostic();
    $colspan=4; // 4 cols for: sqlstate, errlin, errcol, rowcount
    if ($err["errlin"] == -1)
        --$colspan;
    if ($err["errcol"] == -1)
        --$colspan;
    if ($err["rowcount"] == 0)
        --$colspan;
    echo "<TABLE BORDER>\n";
    echo "<TR><TH COLSPAN=".$colspan."><FONT COLOR=red>ERROR:</FONT> ".
              htmlspecialchars($err["errmsg"])."</TH></TR>\n";
    if ($err["errcol"] >= 0) {
        echo "<TR><TD COLSPAN=".$colspan."><PRE>\n";
        $errstmt = $exec_str."\n";
        for ($lin=0; $errstmt != ""; ++$lin) {
            if ($lin != $err["errlin"]) { // $lin is less or greater than errlin
                if (!($i = strchr ($errstmt, "\n")))
                    $i = "";
                $line = substr ($errstmt, 0, strlen($errstmt)-strlen($i)+1);
                $errstmt = substr($i, 1);
                if ($line != "\n")
                    print htmlspecialchars ($line);
            } else {
                if (! ($i = strchr ($errstmt, "\n")))
                    $i = "";
                $line = substr ($errstmt, 0, strlen ($errstmt)-strlen($i)+1);
                $errstmt = substr($i, 1);
                for ($col=0; $col < $err["errcol"]; ++$col)
```

```
                    echo (substr($line, $col, 1) == "\t") ? "\t" : ".";
                echo "<FONT COLOR=RED><BLINK>\\</BLINK></FONT>\n";
                print "<FONT COLOR=\"#880000\">".htmlspecialchars($line)."</FONT>";
                for ($col=0; $col < $err["errcol"]; ++$col)
                    echo (substr ($line, $col, 1) == "\t") ? "\t" : ".";
                echo "<FONT COLOR=RED><BLINK>/</BLINK></FONT>\n";
            }
        }
        echo "</PRE></TD></TR>\n";
    }
    echo "<TR>\n";
    echo " <TD>sqlstate=" . $err["sqlstate"] . "</TD>\n";
    if ($err["errlin"] != -1)
        echo " <TD>errlin=" . $err["errlin"] . "</TD>\n";
    if ($err["errcol"] != -1)
        echo " <TD>errcol=" . $err["errcol"] . "</TD>\n";
    if ($err["rowcount"] != 0)
         echo " <TD>rowcount=" . $err["rowcount"] . "</TD>\n";
    echo "</TR>\n";
    echo "</TABLE>\n";
}

if (!sesam_connect ("mycatalog", "phoneno", "otto"))
  die ("cannot connect");

$stmt = "SELECT * FROM phone\n".
        " WHERE@ LASTNAME='KRAEMER'\n".
        " ORDER BY FIRSTNAME";
if (!($result = sesam_query ($stmt)))
    PrintReturncode ($stmt);
?>
```

See also: **sesam_errormsg()** for simple access to the error string only

sesam_disconnect

(PHP 3 CVS only)

sesam_disconnect -- Detach from SESAM connection

Description

bool **sesam_disconnect** (void)

Returns: always **TRUE**.

sesam_disconnect() closes the logical link to a SESAM database (without actually disconnecting and unloading the driver).

Note that this isn't usually necessary, as the open connection is automatically closed at the end of the script's execution. Uncommitted data will be discarded, because an implicit **sesam_rollback()** is executed.

sesam_disconnect() will not close the persistent link, it will only invalidate the currently defined *"catalog"*, *"schema"* and *"user"* triple, so that any sesam function called after **sesam_disconnect()** will fail.

See also: **sesam_connect()**.

Example 1. Closing a SESAM connection

```
if (sesam_connect ("mycatalog", "myschema", "otto")) {
    ... some queries and stuff ...
    sesam_disconnect();
}
```

sesam_errormsg

(PHP 3 CVS only)

sesam_errormsg -- Returns error message of last SESAM call

Description

string **sesam_errormsg** (void)

Returns the SESAM error message associated with the most recent SESAM error.

```
if (!sesam_execimm ($stmt))
  printf ("%s<br>\n", sesam_errormsg());
```

See also: sesam_diagnostic() for the full set of SESAM SQL status information

sesam_execimm

(PHP 3 CVS only)

sesam_execimm -- Execute an "immediate" SQL-statement

Description

string **sesam_execimm** (string query)

Returns: A SESAM "result identifier" on success, or **FALSE** on error.

sesam_execimm() executes an "immediate" statement (i.e., a statement like UPDATE, INSERT or DELETE which returns no result, and has no INPUT or OUTPUT variables). "select type" queries can not be used with **sesam_execimm()**. Sets the *affected_rows* value for retrieval by the sesam_affected_rows() function.

Note that sesam_query() can handle both "immediate" and "select-type" queries. Use **sesam_execimm()** only if you know beforehand what type of statement will be executed. An attempt to use SELECT type queries with **sesam_execimm()** will return `$err["sqlstate"] == "42SBW"`.

The returned "result identifier" can not be used for retrieving anything but the sesam_affected_rows(); it is only returned for symmetry with the sesam_query() function.

```
$stmt = "INSERT INTO mytable VALUES ('one', 'two')";
$result = sesam_execimm ($stmt);
$err = sesam_diagnostic();
print ("sqlstate = ".$err["sqlstate"]."\n".
       "Affected rows = ".$err["rowcount"]." == ".
       sesam_affected_rows($result)."\n");
```

See also: sesam_query() and sesam_affected_rows().

sesam_fetch_array

(PHP 3 CVS only)

sesam_fetch_array -- Fetch one row as an associative array

Description

array **sesam_fetch_array** (string result_id [, int whence [, int offset]])

Returns an array that corresponds to the fetched row, or **FALSE** if there are no more rows.

sesam_fetch_array() is an alternative version of sesam_fetch_row(). Instead of storing the data in the numeric indices of the result array, it stores the data in associative indices, using the field names as keys.

result_id is a valid result id returned by sesam_query() (select type queries only!).

For the valid values of the optional *whence*and *offset* parameters, see the sesam_fetch_row() function for details.

sesam_fetch_array() fetches one row of data from the result associated with the specified result identifier. The row is returned as an associative array. Each result column is stored with an associative index equal to its column (aka. field) name. The column names are converted to lower case.

Columns without a field name (e.g., results of arithmetic operations) and empty fields are not stored in the array. Also, if two or more columns of the result have the same column names, the later column will take precedence. In this situation, either call sesam_fetch_row() or make an alias for the column.

```
SELECT TBL1.COL AS FOO, TBL2.COL AS BAR FROM TBL1, TBL2
```

A special handling allows fetching "multiple field" columns (which would otherwise all have the same column names). For each

column of a "multiple field", the index name is constructed by appending the string "(n)" where n is the sub-index of the multiple field column, ranging from 1 to its declared repetition factor. The indices are NOT zero based, in order to match the nomenclature used in the respective query syntax. For a column declared as:

```
CREATE TABLE ... ( ... MULTI(3) INT )
```

the associative indices used for the individual "multiple field" columns would be `"multi(1)"`, `"multi(2)"`, and `"multi(3)"` respectively.

Subsequent calls to **sesam_fetch_array()** would return the next (or prior, or n'th next/prior, depending on the scroll attributes) row in the result set, or **FALSE** if there are no more rows.

Example 1. SESAM fetch array

```
<?php
$result = sesam_query ("SELECT * FROM phone\n".
                       "  WHERE LASTNAME='".strtoupper($name)."'\n".
                       "  ORDER BY FIRSTNAME", 1);
if (!$result) {
    ... error ...
}
// print the table:
print "<TABLE BORDER>\n";
while (($row = sesam_fetch_array ($result)) && count ($row) > 0) {
    print " <TR>\n";
    print " <TD>".htmlspecialchars ($row["firstname"])."</TD>\n";
    print " <TD>".htmlspecialchars ($row["lastname"])."</TD>\n";
    print " <TD>".htmlspecialchars ($row["phoneno"])."</TD>\n";
    print " </TR>\n";
}
print "</TABLE>\n";
sesam_free_result ($result);
?>
```

See also: **sesam_fetch_row()** which returns an indexed array.

sesam_fetch_result

(PHP 3 CVS only)

sesam_fetch_result -- Return all or part of a query result

Description

mixed **sesam_fetch_result** (string result_id [, int max_rows])

Returns a mixed array with the query result entries, optionally limited to a maximum of *max_rows* rows. Note that both row and column indexes are zero-based.

Table 1. Mixed result set returned by sesam_fetch_result()

Array Element	Contents
int $arr["count"]	number of columns in result set (or zero if this was an "immediate" query)
int $arr["rows"]	number of rows in result set (between zero and *max_rows*)
bool $arr["truncated"]	**TRUE** if the number of rows was at least *max_rows*, **FALSE** otherwise. Note that even when this is **TRUE**, the next **sesam_fetch_result()** call may return zero rows because there are no more result entries.
mixed $arr[col][row]	result data for all the fields at row(`row`) and column(`col`), (where the integer index `row` is between 0 and `$arr["rows"]-1`, and `col` is between 0 and `$arr["count"]-1`). Fields may be empty, so you must check for the existence of a field by using the php **isset()** function. The type of the returned fields depend on the respective SQL type declared for its column (see SESAM overview for the conversions applied). SESAM "multiple fields" are "inlined" and treated like a sequence of columns.

Note that the amount of memory used up by a large query may be gigantic. Use the *max_rows* parameter to limit the maximum number of rows returned, unless you are absolutely sure that your result will not use up all available memory.

See also: **sesam_fetch_row()**, and **sesam_field_array()** to check for "multiple fields". See the description of the **sesam_query()** function for a complete example using **sesam_fetch_result()**.

sesam_fetch_row

(PHP 3 CVS only)

sesam_fetch_row -- Fetch one row as an array

Description

array **sesam_fetch_row** (string result_id [, int whence [, int offset]])

Returns an array that corresponds to the fetched row, or **FALSE** if there are no more rows.

The number of columns in the result set is returned in an associative array element $array["count"]. Because some of the result columns may be empty, the **count()** function can not be used on the result row returned by **sesam_fetch_row()**.

result_id is a valid result id returned by **sesam_query()** (select type queries only!).

whence is an optional parameter for a fetch operation on "scrollable" cursors, which can be set to the following predefined constants:

Table 1. Valid values for *"whence"* parameter

Value	Constant	Meaning
0	SESAM_SEEK_NEXT	read sequentially (after fetch, the internal default is set to SESAM_SEEK_NEXT)
1	SESAM_SEEK_PRIOR	read sequentially backwards (after fetch, the internal default is set to SESAM_SEEK_PRIOR)
2	SESAM_SEEK_FIRST	rewind to first row (after fetch, the default is set to SESAM_SEEK_NEXT)
3	SESAM_SEEK_LAST	seek to last row (after fetch, the default is set to SESAM_SEEK_PRIOR)
4	SESAM_SEEK_ABSOLUTE	seek to absolute row number given as *offset* (Zero-based. After fetch, the internal default is set to SESAM_SEEK_ABSOLUTE, and the internal offset value is auto-incremented)
5	SESAM_SEEK_RELATIVE	seek relative to current scroll position, where *offset* can be a positive or negative offset value.

This parameter is only valid for "scrollable" cursors.

When using "scrollable" cursors, the cursor can be freely positioned on the result set. If the *whence* parameter is omitted, the global default values for the scrolling type (initialized to: SESAM_SEEK_NEXT, and settable by **sesam_seek_row()**) are used. If *whence* is supplied, its value replaces the global default.

offset is an optional parameter which is only evaluated (and required) if *whence* is either SESAM_SEEK_RELATIVE or SESAM_SEEK_ABSOLUTE. This parameter is only valid for "scrollable" cursors.

sesam_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array (indexed by values between `0` and `$array["count"]-1`). Fields may be empty, so you must check for the existence of a field by using the php **isset()** function. The type of the returned fields depend on the respective SQL type declared for its column (see SESAM overview for the conversions applied). SESAM "multiple fields" are "inlined" and treated like a sequence of columns.

Subsequent calls to **sesam_fetch_row()** would return the next (or prior, or n'th next/prior, depending on the scroll attributes) row in the result set, or **FALSE** if there are no more rows.

Example 1. SESAM fetch rows

```
<?php
$result = sesam_query ("SELECT * FROM phone\n".
                       "  WHERE LASTNAME='".strtoupper($name)."'\n".
                       "  ORDER BY FIRSTNAME", 1);
if (!$result) {
    ... error ...
}
// print the table in backward order
print "<TABLE BORDER>\n";
$row = sesam_fetch_row ($result, SESAM_SEEK_LAST);
while (is_array ($row)) {
    print " <TR>\n";
    for ($col = 0; $col < $row["count"]; ++$col) {
        print "  <TD>".htmlspecialchars ($row[$col])."</TD>\n";
    }
    print " </TR>\n";
    // use implied SESAM_SEEK_PRIOR
    $row = sesam_fetch_row ($result);
}
print "</TABLE>\n";
sesam_free_result ($result);
?>
```

See also: **sesam_fetch_array()** which returns an associative array, and **sesam_fetch_result()** which returns many rows per invocation.

sesam_field_array

(PHP 3 CVS only)

sesam_field_array -- Return meta information about individual columns in a result

Description

array **sesam_field_array** (string result_id)

result_id is a valid result id returned by **sesam_query()**.

Returns a mixed associative/indexed array with meta information (column name, type, precision, ...) about individual columns of the result after the query associated with *result_id*.

Table 1. Mixed result set returned by sesam_field_array()

Array Element	Contents
int $arr["count"]	Total number of columns in result set (or zero if this was an "immediate" query). SESAM "multiple fields" are "inlined" and treated like the respective number of columns.
string $arr[col]["name"]	column name for column(col), where col is between 0 and $arr["count"]-1. The returned value can be the empty string (for dynamically computed columns). SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same column name.
string $arr[col]["count"]	The "count" attribute describes the repetition factor when the column has been declared as a "multiple field". Usually, the "count" attribute is 1. The first column of a "multiple field" column however contains the number of repetitions (the second and following column of the "multiple field" contain a "count" attribute of 1). This can be used to detect "multiple fields" in the result set. See the example shown in the **sesam_query()** description for a sample use of the "count" attribute.
string $arr[col]["type"]	php variable type of the data for column(col), where col is between 0 and $arr["count"]-1. The returned value can be one of • **integer** • **float** • **string** depending on the SQL type of the result. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same php type.
string $arr[col]["sqltype"]	SQL variable type of the column data for column(col), where col is between 0 and $arr["count"]-1. The returned value can be one of • "CHARACTER" • "VARCHAR" • "NUMERIC" • "DECIMAL" • "INTEGER" • "SMALLINT" • "FLOAT" • "REAL" • "DOUBLE" • "DATE" • "TIME" • "TIMESTAMP" describing the SQL type of the result. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same SQL type.

Array Element	Contents
string $arr[col]["length"]	The SQL "length" attribute of the SQL variable in column(`col`), where `col` is between 0 and `$arr["count"]-1`. The "length" attribute is used with "CHARACTER" and "VARCHAR" SQL types to specify the (maximum) length of the string variable. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same length attribute.
string $arr[col]["precision"]	The "precision" attribute of the SQL variable in column(`col`), where `col` is between 0 and `$arr["count"]-1`. The "precision" attribute is used with numeric and time data types. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same precision attribute.
string $arr[col]["scale"]	The "scale" attribute of the SQL variable in column(`col`), where `col` is between 0 and `$arr["count"]-1`. The "scale" attribute is used with numeric data types. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same scale attribute.

See the sesam_query() function for an example of the **sesam_field_array()** use.

sesam_field_name

(PHP 3 CVS only)

sesam_field_name -- Return one column name of the result set

Description

int **sesam_field_name** (string result_id, int index)

Returns the name of a field (i.e., the column name) in the result set, or **FALSE** on error.

For "immediate" queries, or for dynamic columns, an empty string is returned.

> **Note:** The column index is zero-based, not one-based as in SESAM.

See also: sesam_field_array(). It provides an easier interface to access the column names and types, and allows for detection of "multiple fields".

sesam_free_result

(PHP 3 CVS only)

sesam_free_result -- Releases resources for the query

Description

int **sesam_free_result** (string result_id)

Releases resources for the query associated with *result_id*. Returns **FALSE** on error.

sesam_num_fields

(PHP 3 CVS only)

sesam_num_fields -- Return the number of fields/columns in a result set

Description

int **sesam_num_fields** (string result_id)

After calling sesam_query() with a "select type" query, this function gives you the number of columns in the result. Returns an integer describing the total number of columns (aka. fields) in the current *result_id* result set or **FALSE** on error.

For "immediate" statements, the value zero is returned. The SESAM "multiple field" columns count as their respective dimension, i.e., a three-column "multiple field" counts as three columns.

See also: sesam_query() and sesam_field_array() for a way to distinguish between "multiple field" columns and regular columns.

sesam_query

(PHP 3 CVS only)

sesam_query -- Perform a SESAM SQL query and prepare the result

Description

string **sesam_query** (string query [, bool scrollable])

Returns: A SESAM "result identifier" on success, or **FALSE** on error.

A "result_id" resource is used by other functions to retrieve the query results.

sesam_query() sends a query to the currently active database on the server. It can execute both "immediate" SQL statements and "select type" queries. If an "immediate" statement is executed, then no cursor is allocated, and any subsequent sesam_fetch_row() or sesam_fetch_result() call will return an empty result (zero columns, indicating end-of-result). For "select type" statements, a result descriptor and a (scrollable or sequential, depending on the optional boolean *scrollable* parameter) cursor will be allocated. If *scrollable* is omitted, the cursor will be sequential.

When using "scrollable" cursors, the cursor can be freely positioned on the result set. For each "scrollable" query, there are global default values for the scrolling type (initialized to: SESAM_SEEK_NEXT) and the scrolling offset which can either be set once by sesam_seek_row() or each time when fetching a row using sesam_fetch_row().

For "immediate" statements, the number of affected rows is saved for retrieval by the sesam_affected_rows() function.

See also: sesam_fetch_row() and sesam_fetch_result().

Example 1. Show all rows of the "phone" table as a html table

```
<?php
if (!sesam_connect ("phonedb", "demo", "otto"))
    die ("cannot connect");
$result = sesam_query ("select * from phone");
if (!$result) {
    $err = sesam_diagnostic();
    die ($err["errmsg"]);
}
echo "<TABLE BORDER>\n";
// Add title header with column names above the result:
if ($cols = sesam_field_array ($result)) {
    echo " <TR><TH COLSPAN=".$cols["count"].">Result:</TH></TR>\n";
    echo " <TR>\n";
    for ($col = 0; $col < $cols["count"]; ++$col) {
        $colattr = $cols[$col];
        /* Span the table head over SESAM's "Multiple Fields": */
        if ($colattr["count"] > 1) {
            echo "  <TH COLSPAN=".$colattr["count"].">".$colattr["name"].
                "(1..".$colattr["count"].")</TH>\n";
            $col += $colattr["count"] - 1;
        } else
            echo "  <TH>" . $colattr["name"] . "</TH>\n";
    }
    echo " </TR>\n";
}

do {
    // Fetch the result in chunks of 100 rows max.
    $ok = sesam_fetch_result ($result, 100);
    for ($row=0; $row < $ok["rows"]; ++$row) {
        echo " <TR>\n";
        for ($col = 0; $col < $ok["cols"]; ++$col) {
            if (isset($ok[$col][$row]))
                echo "  <TD>" . $ok[$col][$row] . "</TD>\n";
            } else {
                echo "  <TD>-empty-</TD>\n";
            }
        }
        echo " </TR>\n";
    }
}
while ($ok["truncated"]) { // while there may be more data
    echo "</TABLE>\n";
}
// free result id
sesam_free_result($result);
?>
```

sesam_rollback

(PHP 3 CVS only)

sesam_rollback -- Discard any pending updates to the SESAM database

Description

bool **sesam_rollback** (void)

Returns: **TRUE** on success, **FALSE** on errors

sesam_rollback() discards any pending updates to the database. Also affected are result cursors and result descriptors.

At the end of each script, and as part of the **sesam_disconnect()** function, an implied **sesam_rollback()** is executed, discarding any pending changes to the database.

See also: **sesam_commit()**.

Example 1. Discarding an update to the SESAM database

```
<?php
if (sesam_connect ("mycatalog", "myschema", "otto")) {
    if (sesam_execimm ("INSERT INTO mytable VALUES (*, 'Small Test', <0, 8, 15>)")
        && sesam_execimm ("INSERT INTO othertable VALUES (*, 'Another Test', 1)"))
        sesam_commit();
    else
        sesam_rollback();
}
?>
```

sesam_seek_row

(PHP 3 CVS only)

sesam_seek_row -- Set scrollable cursor mode for subsequent fetches

Description

bool **sesam_seek_row** (string result_id, int whence [, int offset])

result_id is a valid result id (select type queries only, and only if a "scrollable" cursor was requested when calling **sesam_query()**).

whence sets the global default value for the scrolling type, it specifies the scroll type to use in subsequent fetch operations on "scrollable" cursors, which can be set to the following predefined constants:

Table 1. Valid values for *"whence"* parameter

Value	Constant	Meaning
0	SESAM_SEEK_NEXT	read sequentially
1	SESAM_SEEK_PRIOR	read sequentially backwards
2	SESAM_SEEK_FIRST	fetch first row (after fetch, the default is set to SESAM_SEEK_NEXT)
3	SESAM_SEEK_LAST	fetch last row (after fetch, the default is set to SESAM_SEEK_PRIOR)
4	SESAM_SEEK_ABSOLUTE	fetch absolute row number given as *offset* (Zero-based. After fetch, the default is set to SESAM_SEEK_ABSOLUTE, and the offset value is auto-incremented)
5	SESAM_SEEK_RELATIVE	fetch relative to current scroll position, where *offset* can be a positive or negative offset value (this also sets the default "offset" value for subsequent fetches).

offset is an optional parameter which is only evaluated (and required) if *whence* is either SESAM_SEEK_RELATIVE or SESAM_SEEK_ABSOLUTE.

sesam_settransaction

(PHP 3 CVS only)

sesam_settransaction -- Set SESAM transaction parameters

Description

bool **sesam_settransaction** (int isolation_level, int read_only)

Returns: **TRUE** if the values are valid, and the **settransaction()** operation was successful, **FALSE** otherwise.

sesam_settransaction() overrides the default values for the "isolation level" and "read-only" transaction parameters (which are set in the SESAM configuration file), in order to optimize subsequent queries and guarantee database consistency. The overridden values are used for the next transaction only.

sesam_settransaction() can only be called before starting a transaction, not after the transaction has been started already.

To simplify the use in php scripts, the following constants have been predefined in php (see SESAM handbook for detailed explanation of the semantics):

Table 1. Valid values for *"Isolation_Level"* parameter

Value	Constant	Meaning
1	SESAM_TXISOL_READ_UNCOMMITTED	Read Uncommitted
2	SESAM_TXISOL_READ_COMMITTED	Read Committed
3	SESAM_TXISOL_REPEATABLE_READ	Repeatable Read
4	SESAM_TXISOL_SERIALIZABLE	Serializable

Table 2. Valid values for *"Read_Only"* parameter

Value	Constant	Meaning
0	SESAM_TXREAD_READWRITE	Read/Write
1	SESAM_TXREAD_READONLY	Read-Only

The values set by **sesam_settransaction()** will override the default setting specified in the SESAM configuration file.

Example 1. Setting SESAM transaction parameters

```
<?php
sesam_settransaction (SESAM_TXISOL_REPEATABLE_READ,
                      SESAM_TXREAD_READONLY);
?>
```

XCIII. Session handling functions

Introduction

Session support in PHP consists of a way to preserve certain data across subsequent accesses. This enables you to build more customized applications and increase the appeal of your web site.

If you are familiar with the session management of PHPLIB, you will notice that some concepts are similar to PHP's session support.

A visitor accessing your web site is assigned an unique id, the so-called session id. This is either stored in a cookie on the user side or is propagated in the URL.

The session support allows you to register arbitrary numbers of variables to be preserved across requests. When a visitor accesses your site, PHP will check automatically (if session.auto_start is set to 1) or on your request (explicitly through **session_start()** or implicitly through **session_register()**) whether a specific session id has been sent with the request. If this is the case, the prior saved environment is recreated.

All registered variables are serialized after the request finishes. Registered variables which are undefined are marked as being not defined. On subsequent accesses, these are not defined by the session module unless the user defines them later.

> **Note:** Session handling was added in PHP 4.0.

> **Note:** Please note when working with sessions that a record of a session is not created until a variable has been registered using the **session_register()** function or by adding a new key to the `$_SESSION` superglobal array. This holds true regardless of if a session has been started using the **session_start()** function.

Sessions and security

External links: Session fixation

The session module cannot guarantee that the information you store in a session is only viewed by the user who created the session. You need to take additional measures to actively protect the integrity of the session, depending on the value associated with it.

Assess the importance of the data carried by your sessions and deploy addditional protections -- this usually comes at a price, reduced convenience for the user. For example, if you want to protect users from simple social engineering tactics, you need to enable session.use_only_cookies. In that case, cookies must be enabled unconditionally on the user side, or sessions will not work.

There are several ways to leak an existing session id to third parties. A leaked session id enables the third party to access all resources which are associated with a specific id. First, URLs carrying session ids. If you link to an external site, the URL including the session id might be stored in the external site's referrer logs. Second, a more active attacker might listen to your network traffic. If it is not encrypted, session ids will flow in plain text over the network. The solution here is to implement SSL on your server and make it mandatory for users.

Requirements

No external libraries are needed to build this extension.

> **Note:** Optionally you can use shared memory allocation (mm), developed by Ralf S. Engelschall, for session storage. You have to download mm and install it. This option is not available for Windows platforms. Note that the session storage module for mm does not guarantee that concurrent accesses to the same session are properly locked. It might be more appropiate to use a shared memory based filesystem (such as tmpfs on Solaris/Linux, or /dev/md on BSD) to store sessions in files, because they are properly locked.

Installation

Session support is enabled in PHP by default. If you would not like to build your PHP with session support, you should specify the `--disable-session` option to configure. To use shared memory allocation (mm) for session storage configure PHP `--with-mm[=DIR]` .

The windows version of `PHP` has built in support for this extension. You do not need to load any additional extension in order to use these functions.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. Session configuration options

Name	Default	Changeable
session.save_path	"/tmp"	PHP_INI_ALL
session.name	"PHPSESSID"	PHP_INI_ALL
session.save_handler	"files"	PHP_INI_ALL
session.auto_start	"0"	PHP_INI_ALL
session.gc_probability	"1"	PHP_INI_ALL
session.gc_maxlifetime	"1440"	PHP_INI_ALL

Name	Default	Changeable
session.serialize_handler	"php"	PHP_INI_ALL
session.cookie_lifetime	"0"	PHP_INI_ALL
session.cookie_path	"/"	PHP_INI_ALL
session.cookie_domain	""	PHP_INI_ALL
session.cookie_secure	""	PHP_INI_ALL
session.use_cookies	"1"	PHP_INI_ALL
session.use_only_cookies	"0"	PHP_INI_ALL
session.referer_check	""	PHP_INI_ALL
session.entropy_file	""	PHP_INI_ALL
session.entropy_length	"0"	PHP_INI_ALL
session.cache_limiter	"nocache"	PHP_INI_ALL
session.cache_expire	"180"	PHP_INI_ALL
session.use_trans_sid	"0"	PHP_INI_SYSTEM|PHP_INI_PERDIR
url_rewriter.tags	"a=href,area=href,frame=src,input=src,form=fakeentry"	PHP_INI_ALL

For further details and definition of the PHP_INI_* constants see **ini_set()**.

The session management system supports a number of configuration options which you can place in your `php.ini` file. We will give a short overview.

`session.save_handler` **string**

`session.save_handler` defines the name of the handler which is used for storing and retrieving data associated with a session. Defaults to `files`. See also **session_set_save_handler()**.

`session.save_path` **string**

`session.save_path` defines the argument which is passed to the save handler. If you choose the default files handler, this is the path where the files are created. Defaults to `/tmp`. If `session.save_path`'s path depth is more than 2, garbage collection will not be performed. See also **session_save_path()**.

Warning
If you leave this set to a world-readable directory, such as `/tmp` (the default), other users on the server may be able to hijack sessions by getting the list of files in that directory.

Note: Windows users have to change this variable in order to use PHP's session functions. Make sure to specify a valid path, e.g.: `c:/temp`.

`session.name` **string**

`session.name` specifies the name of the session which is used as cookie name. It should only contain alphanumeric characters. Defaults to `PHPSESSID`. See also **session_name()**.

`session.auto_start` **boolean**

`session.auto_start` specifies whether the session module starts a session automatically on request startup. Defaults to `0` (disabled).

`session.serialize_handler` **string**

`session.serialize_handler` defines the name of the handler which is used to serialize/deserialize data. Currently, a PHP internal format (name `php`) and WDDX is supported (name `wddx`). WDDX is only available, if PHP is compiled with WDDX support. Defaults to `php`.

`session.gc_probability` **integer**

`session.gc_probability` specifies the probability that the gc (garbage collection) routine is started on each request in percent. Defaults to `1`.

`session.gc_maxlifetime` **integer**

`session.gc_maxlifetime` specifies the number of seconds after which data will be seen as 'garbage' and cleaned up.

Note: If you are using the default file-based session handler, your filesystem must keep track of access times (atime). Windows FAT does not so you will have to come up with another way to handle garbage collecting your session if you are stuck with a FAT filesystem or any other fs where atime tracking is not available.

session.referer_check **string**

`session.referer_check` contains the substring you want to check each HTTP Referer for. If the Referer was sent by the client and the substring was not found, the embedded session id will be marked as invalid. Defaults to the empty string.

session.entropy_file **string**

`session.entropy_file` gives a path to an external resource (file) which will be used as an additional entropy source in the session id creation process. Examples are `/dev/random` or `/dev/urandom` which are available on many Unix systems.

session.entropy_length **integer**

`session.entropy_length` specifies the number of bytes which will be read from the file specified above. Defaults to `0` (disabled).

session.use_cookies **boolean**

`session.use_cookies` specifies whether the module will use cookies to store the session id on the client side. Defaults to `1` (enabled).

session.use_only_cookies **boolean**

`session.use_only_cookies` specifies whether the module will **only** use cookies to store the session id on the client side. Defaults to `0` (disabled, for backward compatibility). Enabling this setting prevents attacks involved passing session ids in URLs. This setting was added in `PHP` 4.3.0.

session.cookie_lifetime **integer**

`session.cookie_lifetime` specifies the lifetime of the cookie in seconds which is sent to the browser. The value 0 means "until the browser is closed." Defaults to `0`.See also **session_get_cookie_params()** and **session_set_cookie_params()**.

session.cookie_path **string**

`session.cookie_path` specifies path to set in session_cookie. Defaults to `/`.See also **session_get_cookie_params()** and **session_set_cookie_params()**.

session.cookie_domain **string**

`session.cookie_domain` specifies the domain to set in session_cookie. Default is none at all. See also **session_get_cookie_params()** and **session_set_cookie_params()**.

session.cookie_secure **boolean**

`session.cookie_secure` specifies whether cookies should only be sent over secure connections. Defaults to `off`. This setting was added in `PHP` 4.0.4. See also **session_get_cookie_params()** and **session_set_cookie_params()**.

session.cache_limiter **string**

`session.cache_limiter` specifies cache control method to use for session pages (none/nocache/private/private_no_expire/public). Defaults to `nocache`. See also **session_cache_limiter()**.

session.cache_expire **integer**

`session.cache_expire` specifies time-to-live for cached session pages in minutes, this has no effect for nocache limiter. Defaults to `180`. See also **session_cache_expire()**.

session.use_trans_sid **boolean**

`session.use_trans_sid` whether transparent sid support is enabled or not. Defaults to `0` (disabled).

> **Note:** For PHP 4.1.2 or less, it is enabled by compiling with `--enable-trans-sid`. From PHP 4.2.0, trans-sid feature is always compiled.
>
> URL based session management has additional security risks compared to cookie based session management. Users may send an URL that contains an active session ID to their friends by email or users may save an URL that contains a session ID to their bookmarks and access your site with the same session ID always, for example.

url_rewriter.tags **string**

`url_rewriter.tags` specifies which html tags are rewritten to include session id if transparent sid support is enabled. Defaults to `a=href,area=href,frame=src,input=src,form=fakeentry`

The `track_vars` and `register_globals` configuration settings influence how the session variables get stored and restored.

> **Note:** As of PHP 4.0.3, `track_vars` is always turned on.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

SID (string)

Constant containing the session name and session ID in the form of "name=ID".

Examples

Note: As of PHP 4.1.0, $_SESSION is available as a global variable just like $_POST, $_GET, $_REQUEST and so on. Unlike $HTTP_SESSION_VARS, $_SESSION is always global. Therefore, you do not need to use the **global** keyword for $_SESSION. Please note that this documentation has been changed to use $_SESSION everywhere. You can substitute $HTTP_SESSION_VARS for $_SESSION, if you prefer the former. Also note that you must start your session using **session_start()** before use of $_SESSION becomes available.

The keys in the $_SESSION associative array are subject to the same limitations as regular variable names in PHP, i.e. they cannot start with a number and must start with a letter or underscore. For more details see the section on variables in this manual.

If register_globals is disabled, only members of the global associative array $_SESSION can be registered as session variables. The restored session variables will only be available in the array $_SESSION.

Use of $_SESSION (or $HTTP_SESSION_VARS with PHP 4.0.6 or less) is recommended for improved security and code readablity. With $_SESSION, there is no need to use the **session_register()**, **session_unregister()**, **session_is_registered()** functions. Session variables are accessible like any other variables.

Example 1. Registering a variable with $_SESSION.

```
<?php
session_start();
// Use $HTTP_SESSION_VARS with PHP 4.0.6 or less
if (!isset($_SESSION['count'])) {
    $_SESSION['count'] = 0;
} else {
    $_SESSION['count']++;
}
?>
```

Example 2. Unregistering a variable with $_SESSION and register_globals disabled.

```
<?php
session_start();
// Use $HTTP_SESSION_VARS with PHP 4.0.6 or less
unset($_SESSION['count']);
?>
```

Example 3. Unregistering a variable with register_globals enabled, after registering it using $_SESSION.

```
<?php
session_start();
// With PHP 4.3 and later, you can also simply use the prior example.
session_unregister('count');
?>
```

If register_globals is enabled, then each global variable can be registered as session variable. Upon a restart of a session, these variables will be restored to corresponding global variables. Since PHP must know which global variables are registered as session variables, users need to register variables with **session_register()** function. You can avoid this by simply setting entries in $_SESSION.

Caution
If you are using $_SESSION and disable register_globals, do not use **session_register()**, **session_is_registered()** and **session_unregister()**, if your scripts shall work in PHP 4.2 and earlier. You can use these functions in 4.3 and later. If you enable register_globals, **session_unregister()** should be used since session variables are registered as global variables when session data is deserialized. Disabling register_globals is recommended for both security and performance reasons.

Example 4. Registering a variable with register_globals enabled

```
<?php
if (!session_is_registered('count')) {
    session_register("count");
    $count = 0;
}
else {
    $count++;
}
?>
```

If register_globals is enabled, then the global variables and the $_SESSION entries will automatically reference the same values which were registered in the prior session instance.

There is a defect in PHP 4.2.3 and earlier. If you register a new session variable by using **session_register()**, the entry in the global scope and the $_SESSION entry will not reference the same value until the next **session_start()**. I.e. a modification to the newly registered global variable will not be reflected by the $_SESSION entry. This has been corrected in PHP 4.3.

Passing the Session ID

There are two methods to propagate a session id:

- Cookies
- URL parameter

The session module supports both methods. Cookies are optimal, but because they are not always available, we also provide an alternative way. The second method embeds the session id directly into URLs.

PHP is capable of transforming links transparently. Unless you are using PHP 4.2 or later, you need to enable it manually when building PHP. Under UNIX, pass --enable-trans-sid to configure. If this build option and the run-time option session.use_trans_sid are enabled, relative URIs will be changed to contain the session id automatically.

> **Note:** The arg_separator.output php.ini directive allows to customize the argument seperator. For full XHTML conformance, specify & there.

Alternatively, you can use the constant SID which is always defined. If the client did not send an appropriate session cookie, it has the form session_name=session_id. Otherwise, it expands to an empty string. Thus, you can embed it unconditionally into URLs.

The following example demonstrates how to register a variable, and how to link correctly to another page using SID.

Example 5. Counting the number of hits of a single user

```
<?php
if (!session_is_registered('count')) {
    session_register('count');
    $count = 1;
}
else {
    $count++;
}
?>

Hello visitor, you have seen this page <?php echo $count; ?> times.<p>

To continue, <A HREF="nextpage.php?<?php echo SID?>">click here</A>
```

The <?php echo SID?> (<?=SID?> can be used if short_open_tag is enabled) is necessary to preserve the session id in the case that the user has disabled cookies. The <?=SID?> is not necessary, if --enable-trans-sid was used to compile PHP.

> **Note:** Non-relative URLs are assumed to point to external sites and hence don't append the SID, as it would be a security risk to leak the SID to a different server.

Custom Session Handlers

To implement database storage, or any other storage method, you will need to use **session_set_save_handler()** to create a set of user-level storage functions.

Table of Contents

session_cache_expire

(PHP 4 >= 4.2.0)

session_cache_expire -- Return current cache expire

Description

int **session_cache_expire** ([int new_cache_expire])

session_cache_expire() returns the current setting of session.cache_expire from `php.ini`. If *`new_cache_expire`* is given, the current cache expire is replaced with *`new_cache_expire`*.

Also see the session.cache_expire configuration directive.

session_cache_limiter

(PHP 4 >= 4.0.3)

session_cache_limiter -- Get and/or set the current cache limiter

Description

string **session_cache_limiter** ([string cache_limiter])

session_cache_limiter() returns the name of the current cache limiter. If *`cache_limiter`* is specified, the name of the current cache limiter is changed to the new value.

The cache limiter defines which cache control HTTP headers are sent to the client. These headers determine the rules by which the page content may be cached by the client and intermediate proxies. Setting the cache limiter to `nocache` disallows any client/proxy caching. A value of `public` permits caching by proxies and the client, whereas `private` disallows caching by proxies and permits the client to cache the contents.

In `private` mode, the Expire header sent to the client may cause confusion for some browsers, including Mozilla. You can avoid this problem by using `private_no_expire` mode. The expire header is never sent to the client in this mode.

> **Note:** `private_no_expire` was added in PHP 4.2.0.

The cache limiter is reset to the default value stored in session.cache_limiter at request startup time. Thus, you need to call

session_cache_limiter() for every request (and before **session_start()** is called).

Example 1. session_cache_limiter() example

```
<?php

/* set the cache limiter to 'private' */

session_cache_limiter('private');
$cache_limiter = session_cache_limiter();

echo "The cache limiter is now set to $cache_limiter<p>";
?>
```

Also see the session.cache_limiter configuration directive.

session_decode

(PHP 4)

session_decode -- Decodes session data from a string

Description

bool **session_decode** (string data)

session_decode() decodes the session data in *data*, setting variables stored in the session.

Voir aussi **session_encode()**

session_destroy

(PHP 4)

session_destroy -- Destroys all data registered to a session

Description

bool **session_destroy** (void)

session_destroy() destroys all of the data associated with the current session. It does not unset any of the global variables associated with the session, or unset the session cookie.

This function returns **TRUE** on success and **FALSE** on failure to destroy the session data.

Example 1. Destroying a session

```
<?php

// Initialize the session.
// If you are using session_name("something"), don't forget it now!
session_start();
// Unset all of the session variables.
session_unset();
// Finally, destroy the session.
session_destroy();

?>
```

Example 2. Destroying a session with $_SESSION

```
<?php

// Initialize the session.
// If you are using session_name("something"), don't forget it now!
session_start();
// Unset all of the session variables.
$_SESSION = array();
// Finally, destroy the session.
session_destroy();
```

```
?>
```

session_encode

(PHP 4)

session_encode -- Encodes the current session data as a string

Description

string **session_encode** (void)

session_encode() returns a string with the contents of the current session encoded within.

See also **session_decode()**

session_get_cookie_params

(PHP 4)

session_get_cookie_params -- Get the session cookie parameters

Description

array **session_get_cookie_params** (void)

The **session_get_cookie_params()** function returns an array with the current session cookie information, the array contains the following items:

- "lifetime" - The lifetime of the cookie.
- "path" - The path where information is stored.
- "domain" - The domain of the cookie.
- "secure" - The cookie should only be sent over secure connections. (This item was added in PHP 4.0.4.)

See also the configuration directives session.cookie_lifetime, session.cookie_path, session.cookie_domain, session.cookie_secure, and **session_set_cookie_params()**.

session_id

(PHP 4)

session_id -- Get and/or set the current session id

Description

string **session_id** ([string id])

session_id() returns the session id for the current session.

If *id* is specified, it will replace the current session id. **session_id()** needs to be called before **session_start()** for that purpose. Depending on the session handler, not all characters are allowed within the session id. For example, the file session handler only allows characters in the range `a-z, A-Z and 0-9`!

The constant SID can also be used to retrieve the current name and session id as a string suitable for adding to URLs. Note that SID is only defined if the client didn't send the right cookie. See also Session handling.

See also **session_start()**, **session_set_save_handler()**, and session.save_handler.

session_is_registered

(PHP 4)

session_is_registered -- Find out whether a global variable is registered in a session

Description

bool **session_is_registered** (string name)

session_is_registered() returns **TRUE** if there is a global variable with the name *name* registered in the current session.

> **Note:** If `$_SESSION` (or `$HTTP_SESSION_VARS` for PHP 4.0.6 or less) is used, use **isset()** to check a variable is registered in `$_SESSION`.

Caution
If you are using `$_SESSION` (or `$HTTP_SESSION_VARS`), do not use **session_register()**, **session_is_registered()** and **session_unregister()**.

session_module_name

(PHP 4)

session_module_name -- Get and/or set the current session module

Description

string **session_module_name** ([string module])

session_module_name() returns the name of the current session module. If *module* is specified, that module will be used instead.

session_name

(PHP 4)

session_name -- Get and/or set the current session name

Description

string **session_name** ([string name])

session_name() returns the name of the current session. If *name* is specified, the name of the current session is changed to its value.

The session name references the session id in cookies and URLs. It should contain only alphanumeric characters; it should be short and descriptive (i.e. for users with enabled cookie warnings). The session name is reset to the default value stored in `session.name` at request startup time. Thus, you need to call **session_name()** for every request (and before **session_start()** or **session_register()** are called).

Example 1. session_name() examples

```
<?php
/* set the session name to WebsiteID */
$previous_name = session_name("WebsiteID");
echo "The previous session name was $previous_name<p>";
?>
```

See also the session.name configuration directive.

session_readonly

(no version information, might be only in CVS)

session_readonly -- Begin session - reinitializes frozen variables, but no writeback on request end

Description

void **session_readonly** (void)

Read in session data without locking the session data. Changing session data is not possible, but frameset performance will be improved.

session_register

(PHP 4)

session_register -- Register one or more global variables with the current session

Description

bool **session_register** (mixed name [, mixed ...])

session_register() accepts a variable number of arguments, any of which can be either a string holding the name of a variable or an array consisting of variable names or other arrays. For each name, **session_register()** registers the global variable with that name in the current session.

Caution
If you want your script to work regardless of register_globals, you need to use the $_SESSION array. All $_SESSION entries are automatically registered. If your script uses session_register(), it will not work in environments where register_globals is disabled.

Caution
This registers a *global* variable. If you want to register a session variable from within a function, you need to make sure to make it global using the **global** keyword or the `$GLOBALS[]` array, or use the special session arrays as noted below.

Caution
If you are using `$_SESSION` (or `$HTTP_SESSION_VARS`), do not use **session_register()**, **session_is_registered()** and **session_unregister()**.

This function returns **TRUE** when all of the variables are successfully registered with the session.

If **session_start()** was not called before this function is called, an implicit call to **session_start()** with no parameters will be made. `$_SESSION` does not mimick this behavior and requires **session_start()** before use.

You can also create a session variable by simply setting the appropriate member of the `$_SESSION` or `$HTTP_SESSION_VARS` (PHP < 4.1.0) array.

```
$barney = "A big purple dinosaur.";
session_register("barney");

$_SESSION["zim"] = "An invader from another planet.";

# The old way was to use $HTTP_SESSION_VARS
$HTTP_SESSION_VARS["spongebob"] = "He's got square pants.";
```

> **Note:** It is currently impossible to register resource variables in a session. For example, you cannot create a connection to a database and store the connection id as a session variable and expect the connection to still be valid the next time the session is restored. PHP functions that return a resource are identified by having a return type of `resource` in their function definition. A list of functions that return resources are available in the resource types appendix.
>
> If `$_SESSION` (or `$HTTP_SESSION_VARS` for PHP 4.0.6 or less) is used, assign values to `$_SESSION`. For example:
> $_SESSION['var'] = 'ABC';

See also **session_is_registered()** and **session_unregister()**.

session_save_path

(PHP 4)

session_save_path -- Get and/or set the current session save path

Description

string **session_save_path** ([string path])

session_save_path() returns the path of the current directory used to save session data. If *path* is specified, the path to which data is saved will be changed. **session_save_path()** needs to be called before **session_start()** for that purpose.

> **Note:** On some operating systems, you may want to specify a path on a filesystem that handles lots of small files efficiently. For example, on Linux, reiserfs may provide better performance than ext2fs.

See also the session.save_path configuration directive.

session_set_cookie_params

(PHP 4)

session_set_cookie_params -- Set the session cookie parameters

Description

void **session_set_cookie_params** (int lifetime [, string path [, string domain [, bool secure]]])

Set cookie parameters defined in the `php.ini` file. The effect of this function only lasts for the duration of the script.

> **Note:** The *secure* parameter was added in PHP 4.0.4.

See also the configuration directives session.cookie_lifetime, session.cookie_path, session.cookie_domain, session.cookie_secure, and **session_get_cookie_params()**.

session_set_save_handler

(PHP 4)

session_set_save_handler -- Sets user-level session storage functions

Description

bool **session_set_save_handler** (string open, string close, string read, string write, string destroy, string gc)

session_set_save_handler() sets the user-level session storage functions which are used for storing and retrieving data associated with a session. This is most useful when a storage method other than those supplied by PHP sessions is preferred. i.e. Storing the session data in a local database. Returns **TRUE** on success or **FALSE** on failure.

> **Note:** The "write" handler is not executed until after the output stream is closed. Thus, output from debugging statements in the "write" handler will never be seen in the browser. If debugging output is necessary, it is suggested that the debug output be written to a file instead.
>
> **Note:** The write handler is not executed if the session contains no data; this applies even if empty session variables are registered. This differs to the default file-based session save handler, which creates empty session files.

The following example provides file based session storage similar to the PHP sessions default save handler *files*. This example could easily be extended to cover database storage using your favorite PHP supported database engine.

Read function must return string value always to make save handler work as expected. Return empty string if there is no data to read. Return values from other handlers are converted to boolean expression. TRUE for success, FALSE for failure.

Example 1. session_set_save_handler() example

```
<?php
function open ($save_path, $session_name) {
  global $sess_save_path, $sess_session_name;
```

```
  $sess_save_path = $save_path;
  $sess_session_name = $session_name;
  return(true);
}

function close() {
  return(true);
}

function read ($id) {
  global $sess_save_path, $sess_session_name;

  $sess_file = "$sess_save_path/sess_$id";
  if ($fp = @fopen($sess_file, "r")) {
    $sess_data = fread($fp, filesize($sess_file));
    return($sess_data);
  } else {
    return(""); // Must return "" here.
  }

}

function write ($id, $sess_data) {
  global $sess_save_path, $sess_session_name;

  $sess_file = "$sess_save_path/sess_$id";
  if ($fp = @fopen($sess_file, "w")) {
    return(fwrite($fp, $sess_data));
  } else {
    return(false);
  }

}

function destroy ($id) {
  global $sess_save_path, $sess_session_name;

  $sess_file = "$sess_save_path/sess_$id";
  return(@unlink($sess_file));
}

/*****************************************
 * WARNING - You will need to implement some *
 * sort of garbage collection routine here.  *
 *****************************************/
function gc ($maxlifetime) {
  return true;
}

session_set_save_handler ("open", "close", "read", "write", "destroy", "gc");

session_start();

// proceed to use sessions normally

?>
```

See also the session.save_handler configuration directive.

session_start

(PHP 4)

session_start -- Initialize session data

Description

bool **session_start** (void)

session_start() creates a session or resumes the current one based on the current session id that's being passed via a request, such as GET, POST, or a cookie.

If you want to use a named session, you must call **session_name()** before calling **session_start()**.

This function always returns **TRUE**.

> **Note:** If you are using cookie-based sessions, you must call **session_start()** before anything is output to the browser.

session_start() will register internal output handler for URL rewriting when `trans-sid` is enabled. If a user uses `ob_gzhandler` or like with **ob_start()**, the order of output handler is important for proper output. For example, user must register `ob_gzhandler` before session start.

> **Note:** Use of zlib.output_compression is recommended rather than **ob_gzhandler()**

session_unregister

(PHP 4)

session_unregister -- Unregister a global variable from the current session

Description

bool **session_unregister** (string name)

session_unregister() unregisters the global variable named *name* from the current session.

This function returns **TRUE** when the variable is successfully unregistered from the session.

> **Note:** If `$_SESSION` (or `$HTTP_SESSION_VARS` for PHP 4.0.6 or less) is used, use **unset()** to unregister a session variable.

Caution
This function does not unset the corresponding global variable for *name*, it only prevents the variable from being saved as part of the session. You must call **unset()** to remove the corresponding global variable.

Caution
If you are using `$_SESSION` (or `$HTTP_SESSION_VARS`), do not use **session_register()**, **session_is_registered()** and **session_unregister()**.

session_unset

(PHP 4)

session_unset -- Free all session variables

Description

void **session_unset** (void)

The **session_unset()** function frees all session variables currently registered.

> **Note:** If `$_SESSION` (or `$HTTP_SESSION_VARS` for PHP 4.0.6 or less) is used, use **unset()** to unregister session variable. i.e. $_SESSION = array();

session_write_close

(PHP 4 >= 4.0.4)

session_write_close -- Write session data and end session

Description

void **session_write_close** (void)

End the current session and store session data.

Session data is usually stored after your script terminated without the need to call **session_write_close()**, but as session data is locked to prevent concurrent writes only one script may operate on a session at any time. When using framesets together with sessions you will experience the frames loading one by one due to this locking. You can reduce the time needed to load all the frames by ending the session as soon as all changes to session variables are done.

XCIV. Shared Memory Functions

Introduction

Shmop is an easy to use set of functions that allows PHP to read, write, create and delete UNIX shared memory segments. These functions will not typically work on Windows, as it does not support shared memory. As of Windows 2000 though, enabling the `php_shmop.dll` in your `php.ini` will enable this functionality though.

Note: In PHP 4.0.3, these functions were prefixed by `shm` rather than `shmop`.

Requirements

No external libraries are needed to build this extension.

Installation

To use shmop you will need to compile PHP with the `--enable-shmop` parameter in your configure line.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

Predefined Constants

This extension has no constants defined.

Examples

Example 1. Shared Memory Operations Overview

```
<?php

// Create 100 byte shared memory block with system id if 0xff3
$shm_id = shmop_open(0xff3, "c", 0644, 100);
if(!$shm_id) {
    echo "Couldn't create shared memory segment\n";
}

// Get shared memory block's size
$shm_size = shmop_size($shm_id);
echo "SHM Block Size: ".$shm_size. " has been created.\n";

// Lets write a test string into shared memory
$shm_bytes_written = shmop_write($shm_id, "my shared memory block", 0);
if($shm_bytes_written != strlen("my shared memory block")) {
    echo "Couldn't write the entire length of data\n";
}

// Now lets read the string back
$my_string = shmop_read($shm_id, 0, $shm_size);
if(!$my_string) {
    echo "Couldn't read from shared memory block\n";
}
echo "The data inside shared memory was: ".$my_string."\n";

//Now lets delete the block and close the shared memory segment
if(!shmop_delete($shm_id)) {
    echo "Couldn't mark shared memory block for deletion.";
}
shmop_close($shm_id);
```

```
?>
```

Table of Contents

shmop_close

(PHP 4 >= 4.0.4)

shmop_close -- Close shared memory block

Description

int **shmop_close** (int shmid)

shmop_close() is used to close a shared memory block.

shmop_close() takes the shmid, which is the shared memory block identifier created by **shmop_open()**.

Example 1. Closing shared memory block

```
<?php
shmop_close($shm_id);
?>
```

This example will close shared memory block identified by `$shm_id`.

shmop_delete

(PHP 4 >= 4.0.4)

shmop_delete -- Delete shared memory block

Description

int **shmop_delete** (int shmid)

shmop_delete() is used to delete a shared memory block.

shmop_delete() takes the shmid, which is the shared memory block identifier created by **shmop_open()**. On success 1 is returned, on failure 0 is returned.

Example 1. Deleting shared memory block

```
<?php
shmop_delete($shm_id);
?>
```

This example will delete shared memory block identified by `$shm_id`.

shmop_open

(PHP 4 >= 4.0.4)

shmop_open -- Create or open shared memory block

Description

int **shmop_open** (int key, string flags, int mode, int size)

shmop_open() can create or open a shared memory block.

shmop_open() takes 4 parameters: key, which is the system's id for the shared memory block, this parameter can be passed as a decimal or hex. The second parameter are the flags that you can use:

- "a" for access (sets SHM_RDONLY for shmat) use this flag when you need to open an existing shared memory segment for read only
- "c" for create (sets IPC_CREATE) use this flag when you need to create a new shared memory segment or if a segment with the same key exists, try to open it for read and write
- "w" for read & write access use this flag when you need to read and write to a shared memory segment, use this flag in most cases.
- "n" create a new memory segment (sets IPC_CREATE|IPC_EXCL) use this flag when you want to create a new shared memory segment but if one already exists with the same flag, fail. This is useful for security purposes, using this you can prevent race condition exploits.

The third parameter is the mode, which are the permissions that you wish to assign to your memory segment, those are the same as permission for a file. Permissions need to be passed in octal form ex. 0644. The last parameter is size of the shared memory block you wish to create in bytes.

> **Note:** Note: the 3rd and 4th should be entered as 0 if you are opening an existing memory segment. On success **shmop_open()** will return an id that you can use to access the shared memory segment you've created.

Example 1. Create a new shared memory block

```
<?php
$shm_id = shmop_open(0x0fff, "c", 0644, 100);
?>
```

This example opened a shared memory block with a system id of oxofff.

shmop_read

(PHP 4 >= 4.0.4)

shmop_read -- Read data from shared memory block

Description

string **shmop_read** (int shmid, int start, int count)

shmop_read() will read a string from shared memory block.

shmop_read() takes 3 parameters: shmid, which is the shared memory block identifier created by **shmop_open()**, offset from which to start reading and count on the number of bytes to read.

Example 1. Reading shared memory block

```
<?php
$shm_data = shmop_read($shm_id, 0, 50);
?>
```

This example will read 50 bytes from shared memory block and place the data inside `$shm_data`.

shmop_size

(PHP 4 >= 4.0.4)

shmop_size -- Get size of shared memory block

Description

int **shmop_size** (int shmid)

shmop_size() is used to get the size, in bytes of the shared memory block.

shmop_size() takes the shmid, which is the shared memory block identifier created by **shmop_open()**, the function will return and int, which represents the number of bytes the shared memory block occupies.

Example 1. Getting the size of the shared memory block

```
<?php
$shm_size = shmop_size($shm_id);
?>
```

This example will put the size of shared memory block identified by `$shm_id` into `$shm_size`.

shmop_write

(PHP 4 >= 4.0.4)

shmop_write -- Write data into shared memory block

Description

int **shmop_write** (int shmid, string data, int offset)

shmop_write() will write a string into shared memory block.

shmop_write() takes 3 parameters: shmid, which is the shared memory block identifier created by **shmop_open()**, data, a string that you want to write into shared memory block and offset, which specifies where to start writing data inside the shared memory segment.

Example 1. Writing to shared memory block

```
<?php
$shm_bytes_written = shmop_write($shm_id, $my_string, 0);
?>
```

This example will write data inside `$my_string` into shared memory block, `$shm_bytes_written` will contain the number of bytes written.

XCV. Shockwave Flash functions

Introduction

PHP offers the ability to create Shockwave Flash files via Paul Haeberli's libswf module.

> **Note:** SWF support was added in PHP 4 RC2.
>
> The libswf does not have support for Windows. The development of that library has been stopped, and the source is not available to port it to another systems.
>
> For up to date SWF support take a look at the MING functions.

Requirements

You need the libswf library to compile PHP with support for this extension. You can download libswf at ftp://ftp.sgi.com/sgi/graphics/grafica/flash.

Installation

Once you have libswf all you need to do is to configure `--with-swf[=DIR]` where DIR is a location containing the directories include and lib. The include directory has to contain the `swf.h` file and the lib directory has to contain the `libswf.a` file. If you unpack the libswf distribution the two files will be in one directory. Consequently you will have to copy the files to the proper location manually.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

`MOD_COLOR` (integer)

`MOD_MATRIX` (integer)

`TYPE_PUSHBUTTON` (integer)

`TYPE_MENUBUTTON` (integer)

`BSHitTest` (float)

`BSDown` (float)

`BSOver` (float)

`BSUp` (float)

`OverDowntoIdle` (integer)

`IdletoOverDown` (integer)

`OutDowntoIdle` (integer)

`OutDowntoOverDown` (integer)

`OverDowntoOutDown` (integer)

`OverUptoOverDown` (integer)

`OverUptoIdle` (integer)

`IdletoOverUp` (integer)

`ButtonEnter` (integer)

`ButtonExit` (integer)

`MenuEnter` (integer)

`MenuExit` (integer)

Examples

Once you've successfully installed PHP with Shockwave Flash support you can then go about creating Shockwave files from PHP. You would be surprised at what you can do, take the following code:

Example 1. SWF example

```
<?php
swf_openfile ("test.swf", 256, 256, 30, 1, 1, 1);
swf_ortho2 (-100, 100, -100, 100);
swf_defineline (1, -70, 0, 70, 0, .2);
swf_definerect (4, 60, -10, 70, 0, 0);
swf_definerect (5, -60, 0, -70, 10, 0);
swf_addcolor (0, 0, 0, 0);

swf_definefont (10, "Mod");
swf_fontsize (5);
swf_fontslant (10);
swf_definetext (11, "This be Flash wit PHP!", 1);

swf_pushmatrix ();
swf_translate (-50, 80, 0);
swf_placeobject (11, 60);
swf_popmatrix ();

for ($i = 0; $i < 30; $i++) {
    $p = $i/(30-1);
    swf_pushmatrix ();
    swf_scale (1-($p*.9), 1, 1);
    swf_rotate (60*$p,  'z');
    swf_translate (20+20*$p, $p/1.5, 0);
    swf_rotate (270*$p,  'z');
    swf_addcolor ($p, 0, $p/1.2, -$p);
    swf_placeobject (1, 50);
    swf_placeobject (4, 50);
    swf_placeobject (5, 50);
    swf_popmatrix ();
    swf_showframe ();
}

for ($i = 0; $i < 30; $i++) {
    swf_removeobject (50);
    if (($i%4) == 0) {
        swf_showframe ();
    }
}

swf_startdoaction ();
swf_actionstop ();
swf_enddoaction ();

swf_closefile ();
?>
```

Table of Contents

swf_getfontinfo -- The height in pixels of a capital A and a lowercase x
swf_getframe -- Get the frame number of the current frame
swf_labelframe -- Label the current frame
swf_lookat -- Define a viewing transformation
swf_modifyobject -- Modify an object
swf_mulcolor -- Sets the global multiply color to the rgba value specified
swf_nextid -- Returns the next free object id
swf_oncondition -- Describe a transition used to trigger an action list
swf_openfile -- Open a new Shockwave Flash file
swf_ortho2 -- Defines 2D orthographic mapping of user coordinates onto the current viewport
swf_ortho -- Defines an orthographic mapping of user coordinates onto the current viewport
swf_perspective -- Define a perspective projection transformation
swf_placeobject -- Place an object onto the screen
swf_polarview -- Define the viewer's position with polar coordinates
swf_popmatrix -- Restore a previous transformation matrix
swf_posround -- Enables or Disables the rounding of the translation when objects are placed or moved
swf_pushmatrix -- Push the current transformation matrix back unto the stack
swf_removeobject -- Remove an object
swf_rotate -- Rotate the current transformation
swf_scale -- Scale the current transformation
swf_setfont -- Change the current font
swf_setframe -- Switch to a specified frame
swf_shapearc -- Draw a circular arc
swf_shapecurveto3 -- Draw a cubic bezier curve
swf_shapecurveto -- Draw a quadratic bezier curve between two points
swf_shapefillbitmapclip -- Set current fill mode to clipped bitmap
swf_shapefillbitmaptile -- Set current fill mode to tiled bitmap
swf_shapefilloff -- Turns off filling
swf_shapefillsolid -- Set the current fill style to the specified color
swf_shapelinesolid -- Set the current line style
swf_shapelineto -- Draw a line
swf_shapemoveto -- Move the current position
swf_showframe -- Display the current frame
swf_startbutton -- Start the definition of a button
swf_startdoaction -- Start a description of an action list for the current frame
swf_startshape -- Start a complex shape
swf_startsymbol -- Define a symbol
swf_textwidth -- Get the width of a string
swf_translate -- Translate the current transformations
swf_viewport -- Select an area for future drawing

swf_actiongeturl

(PHP 4)

swf_actiongeturl -- Get a URL from a Shockwave Flash movie

Description

void **swf_actiongeturl** (string url, string target)

The **swf_actionGetUrl()** function gets the URL specified by the parameter *url* with the target *target*.

swf_actiongotoframe

(PHP 4)

swf_actiongotoframe -- Play a frame and then stop

Description

void **swf_actiongotoframe** (int framenumber)

The **swf_actionGotoFrame()** function will go to the frame specified by *framenumber*, play it, and then stop.

swf_actiongotolabel

(PHP 4)

swf_actiongotolabel -- Display a frame with the specified label

Description

void **swf_actiongotolabel** (string label)

The **swf_actionGotoLabel()** function displays the frame with the label given by the *label* parameter and then stops.

swf_actionnextframe

(PHP 4)

swf_actionnextframe -- Go foward one frame

Description

void **swf_actionnextframe** (void)

Go foward one frame.

swf_actionplay

(PHP 4)

swf_actionplay -- Start playing the flash movie from the current frame

Description

void **swf_actionplay** (void)

Start playing the flash movie from the current frame.

swf_actionprevframe

(PHP 4)

swf_actionprevframe -- Go backwards one frame

Description

void **swf_actionprevframe** (void)

swf_actionsettarget

(PHP 4)

swf_actionsettarget -- Set the context for actions

Description

void **swf_actionsettarget** (string target)

The **swf_actionSetTarget()** function sets the context for all actions. You can use this to control other flash movies that are

currently playing.

swf_actionstop

(PHP 4)

swf_actionstop -- Stop playing the flash movie at the current frame

Description

void **swf_actionstop** (void)

Stop playing the flash movie at the current frame.

swf_actiontogglequality

(PHP 4)

swf_actiontogglequality -- Toggle between low and high quality

Description

void **swf_actiontogglequality** (void)

Toggle the flash movie between high and low quality.

swf_actionwaitforframe

(PHP 4)

swf_actionwaitforframe -- Skip actions if a frame has not been loaded

Description

void **swf_actionwaitforframe** (int framenumber, int skipcount)

The **swf_actionWaitForFrame()** function will check to see if the frame, specified by the *framenumber* parameter has been loaded, if not it will skip the number of actions specified by the *skipcount* parameter. This can be useful for "Loading..." type animations.

swf_addbuttonrecord

(PHP 4)

swf_addbuttonrecord -- Controls location, appearance and active area of the current button

Description

void **swf_addbuttonrecord** (int states, int shapeid, int depth)

The **swf_addbuttonrecord()** function allows you to define the specifics of using a button. The first parameter, *states*, defines what states the button can have, these can be any or all of the following constants: BSHitTest, BSDown, BSOver or BSUp. The second parameter, the *shapeid* is the look of the button, this is usually the object id of the shape of the button. The *depth* parameter is the placement of the button in the current frame.

Example 1. swf_addbuttonrecord() function example

```
swf_startButton ($objid, TYPE_MENUBUTTON);
    swf_addButtonRecord (BSDown|BSOver, $buttonImageId, 340);
    swf_onCondition (MenuEnter);
        swf_actionGetUrl ("http://www.designmultimedia.com", "_level1");
```

```
    swf_onCondition (MenuExit);
        swf_actionGetUrl ("", "_level1");
swf_endButton ();
```

swf_addcolor

(PHP 4)

swf_addcolor -- Set the global add color to the rgba value specified

Description

void **swf_addcolor** (float r, float g, float b, float a)

The **swf_addcolor()** function sets the global add color to the *rgba* color specified. This color is then used (implicitly) by the **swf_placeobject()**, **swf_modifyobject()** and the **swf_addbuttonrecord()** functions. The color of the object will be add by the *rgba* values when the object is written to the screen.

Note: The *rgba* values can be either positive or negative.

swf_closefile

(PHP 4)

swf_closefile -- Close the current Shockwave Flash file

Description

void **swf_closefile** ([int return_file])

Close a file that was opened by the **swf_openfile()** function. If the *return_file* parameter is set then the contents of the SWF file are returned from the function.

Example 1. Creating a simple flash file based on user input and outputting it and saving it in a database

```
<?php

// The $text variable is submitted by the
// user

// Global variables for database
// access (used in the swf_savedata() function)
$DBHOST = "localhost";
$DBUSER = "sterling";
$DBPASS = "secret";

swf_openfile ("php://stdout", 256, 256, 30, 1, 1, 1);

    swf_definefont (10, "Ligon-Bold");
        swf_fontsize (12);
        swf_fontslant (10);

    swf_definetext (11, $text, 1);

    swf_pushmatrix ();
        swf_translate (-50, 80, 0);
        swf_placeobject (11, 60);
    swf_popmatrix ();

    swf_showframe ();

    swf_startdoaction ();
        swf_actionstop ();
    swf_enddoaction ();

$data = swf_closefile (1);

$data ?
  swf_savedata ($data) :
  die ("Error could not save SWF file");

// void swf_savedata (string data)
// Save the generated file a database
// for later retrieval
function swf_savedata ($data)
```

```
{
    global $DBHOST,
           $DBUSER,
           $DBPASS;

    $dbh = @mysql_connect ($DBHOST, $DBUSER, $DBPASS);

    if (!$dbh) {
        die (sprintf ("Error [%d]: %s",
                      mysql_errno (), mysql_error ()));
    }

    $stmt = "INSERT INTO swf_files (file) VALUES ('$data')";

    $sth = @mysql_query ($stmt, $dbh);

    if (!$sth) {
        die (sprintf ("Error [%d]: %s",
                      mysql_errno (), mysql_error ()));
    }

    @mysql_free_result ($sth);
    @mysql_close ($dbh);
}
?>
```

swf_definebitmap

(PHP 4)

swf_definebitmap -- Define a bitmap

Description

void **swf_definebitmap** (int objid, string image_name)

The **swf_definebitmap()** function defines a bitmap given a GIF, JPEG, RGB or FI image. The image will be converted into a Flash JPEG or Flash color map format.

swf_definefont

(PHP 4)

swf_definefont -- Defines a font

Description

void **swf_definefont** (int fontid, string fontname)

The **swf_definefont()** function defines a font given by the *fontname* parameter and gives it the id specified by the *fontid* parameter. It then sets the font given by *fontname* to the current font.

swf_defineline

(PHP 4)

swf_defineline -- Define a line

Description

void **swf_defineline** (int objid, float x1, float y1, float x2, float y2, float width)

The **swf_defineline()** defines a line starting from the x coordinate given by *x1* and the y coordinate given by *y1* parameter. Up to the x coordinate given by the *x2* parameter and the y coordinate given by the *y2* parameter. It will have a width defined by the *width* parameter.

swf_definepoly

(PHP 4)

swf_definepoly -- Define a polygon

Description

void **swf_definepoly** (int objid, array coords, int npoints, float width)

The **swf_definepoly()** function defines a polygon given an array of x, y coordinates (the coordinates are defined in the parameter *coords*). The parameter *npoints* is the number of overall points that are contained in the array given by *coords*. The *width* is the width of the polygon's border, if set to 0.0 the polygon is filled.

swf_definerect

(PHP 4)

swf_definerect -- Define a rectangle

Description

void **swf_definerect** (int objid, float x1, float y1, float x2, float y2, float width)

The **swf_definerect()** defines a rectangle with an upper left hand coordinate given by the x, *x1*, and the y, *y1*. And a lower right hand coordinate given by the x coordinate, *x2*, and the y coordinate, *y2* . Width of the rectangles border is given by the *width* parameter, if the width is 0.0 then the rectangle is filled.

swf_definetext

(PHP 4)

swf_definetext -- Define a text string

Description

void **swf_definetext** (int objid, string str, int docenter)

Define a text string (the *str* parameter) using the current font and font size. The *docenter* is where the word is centered, if *docenter* is 1, then the word is centered in x.

swf_endbutton

(PHP 4)

swf_endbutton -- End the definition of the current button

Description

void **swf_endbutton** (void)

The **swf_endButton()** function ends the definition of the current button.

swf_enddoaction

(PHP 4)

swf_enddoaction -- End the current action

Description

void **swf_enddoaction** (void)

Ends the current action started by the **swf_startdoaction()** function.

swf_endshape

(PHP 4)

swf_endshape -- Completes the definition of the current shape

Description

void **swf_endshape** (void)

The **swf_endshape()** completes the definition of the current shape.

swf_endsymbol

(PHP 4)

swf_endsymbol -- End the definition of a symbol

Description

void **swf_endsymbol** (void)

The **swf_endsymbol()** function ends the definition of a symbol that was started by the **swf_startsymbol()** function.

swf_fontsize

(PHP 4)

swf_fontsize -- Change the font size

Description

void **swf_fontsize** (float size)

The **swf_fontsize()** function changes the font size to the value given by the *size* parameter.

swf_fontslant

(PHP 4)

swf_fontslant -- Set the font slant

Description

void **swf_fontslant** (float slant)

Set the current font slant to the angle indicated by the *slant* parameter. Positive values create a foward slant, negative values create a negative slant.

swf_fonttracking

(PHP 4)

swf_fonttracking -- Set the current font tracking

Description

void **swf_fonttracking** (float tracking)

Set the font tracking to the value specified by the *tracking* parameter. This function is used to increase the spacing between letters and text, positive values increase the space and negative values decrease the space between letters.

swf_getbitmapinfo

(PHP 4)

swf_getbitmapinfo -- Get information about a bitmap

Description

array **swf_getbitmapinfo** (int bitmapid)

The **swf_getbitmapinfo()** function returns an array of information about a bitmap given by the *bitmapid* parameter. The returned array has the following elements:

- "size" - The size in bytes of the bitmap.
- "width" - The width in pixels of the bitmap.
- "height" - The height in pixels of the bitmap.

swf_getfontinfo

(PHP 4)

swf_getfontinfo -- The height in pixels of a capital A and a lowercase x

Description

array **swf_getfontinfo** (void)

The **swf_getfontinfo()** function returns an associative array with the following parameters:

- Aheight - The height in pixels of a capital A.
- xheight - The height in pixels of a lowercase x.

swf_getframe

(PHP 4)

swf_getframe -- Get the frame number of the current frame

Description

int **swf_getframe** (void)

The **swf_getframe()** function gets the number of the current frame.

swf_labelframe

(PHP 4)

swf_labelframe -- Label the current frame

Description

void **swf_labelframe** (string name)

Label the current frame with the name given by the *name* parameter.

swf_lookat

(PHP 4)

swf_lookat -- Define a viewing transformation

Description

void **swf_lookat** (float view_x, float view_y, float view_z, float reference_x, float reference_y, float reference_z, float twist)

The **swf_lookat()** function defines a viewing transformation by giving the viewing position (the parameters *view_x*, *view_y*, and *view_z*) and the coordinates of a reference point in the scene, the reference point is defined by the *reference_x*, *reference_y* , and *reference_z* parameters. The *twist* controls the rotation along with viewer's z axis.

swf_modifyobject

(PHP 4)

swf_modifyobject -- Modify an object

Description

void **swf_modifyobject** (int depth, int how)

Updates the position and/or color of the object at the specified depth, *depth*. The parameter *how* determines what is updated. *how* can either be the constant MOD_MATRIX or MOD_COLOR or it can be a combination of both (MOD_MATRIX|MOD_COLOR).

MOD_COLOR uses the current mulcolor (specified by the function **swf_mulcolor()**) and addcolor (specified by the function **swf_addcolor()**) to color the object. MOD_MATRIX uses the current matrix to position the object.

swf_mulcolor

(PHP 4)

swf_mulcolor -- Sets the global multiply color to the rgba value specified

Description

void **swf_mulcolor** (float r, float g, float b, float a)

The **swf_mulcolor()** function sets the global multiply color to the *rgba* color specified. This color is then used (implicitly) by the **swf_placeobject()**, **swf_modifyobject()** and the **swf_addbuttonrecord()** functions. The color of the object will be multiplied by the *rgba* values when the object is written to the screen.

> **Note:** The *rgba* values can be either positive or negative.

swf_nextid

(PHP 4)

swf_nextid -- Returns the next free object id

Description

int **swf_nextid** (void)

The **swf_nextid()** function returns the next available object id.

swf_oncondition

(PHP 4)

swf_oncondition -- Describe a transition used to trigger an action list

Description

void **swf_oncondition** (int transition)

The **swf_onCondition()** function describes a transition that will trigger an action list. There are several types of possible transitions, the following are for buttons defined as TYPE_MENUBUTTON:

- IdletoOverUp
- OverUptoIdle
- OverUptoOverDown
- OverDowntoOverUp
- IdletoOverDown
- OutDowntoIdle
- MenuEnter (IdletoOverUp|IdletoOverDown)
- MenuExit (OverUptoIdle|OverDowntoIdle)

For TYPE_PUSHBUTTON there are the following options:

- IdletoOverUp
- OverUptoIdle
- OverUptoOverDown
- OverDowntoOverUp
- OverDowntoOutDown
- OutDowntoOverDown
- OutDowntoIdle
- ButtonEnter (IdletoOverUp|OutDowntoOverDown)
- ButtonExit (OverUptoIdle|OverDowntoOutDown)

swf_openfile

(PHP 4)

swf_openfile -- Open a new Shockwave Flash file

Description

void **swf_openfile** (string filename, float width, float height, float framerate, float r, float g, float b)

The **swf_openfile()** function opens a new file named *filename* with a width of *width* and a height of *height* a frame rate of *framerate* and background with a red color of *r* a green color of *g* and a blue color of *b*.

The **swf_openfile()** must be the first function you call, otherwise your script will cause a segfault. If you want to send your

output to the screen make the filename: "php://stdout" (support for this is in 4.0.1 and up).

swf_ortho2

(PHP 4)

swf_ortho2 -- Defines 2D orthographic mapping of user coordinates onto the current viewport

Description

void **swf_ortho2** (float xmin, float xmax, float ymin, float ymax)

The **swf_ortho2()** function defines a two dimensional orthographic mapping of user coordinates onto the current viewport, this defaults to one to one mapping of the area of the Flash movie. If a perspective transformation is desired, the **swf_perspective ()** function can be used.

swf_ortho

(PHP 4 >= 4.0.1)

swf_ortho -- Defines an orthographic mapping of user coordinates onto the current viewport

Description

void **swf_ortho** (float xmin, float xmax, float ymin, float ymax, float zmin, float zmax)

The **swf_ortho()** function defines a orthographic mapping of user coordinates onto the current viewport.

swf_perspective

(PHP 4)

swf_perspective -- Define a perspective projection transformation

Description

void **swf_perspective** (float fovy, float aspect, float near, float far)

The **swf_perspective()** function defines a perspective projection transformation. The *fovy* parameter is field-of-view angle in the y direction. The *aspect* parameter should be set to the aspect ratio of the viewport that is being drawn onto. The *near* parameter is the near clipping plane and the *far* parameter is the far clipping plane.

> **Note:** Various distortion artifacts may appear when performing a perspective projection, this is because Flash players only have a two dimensional matrix. Some are not to pretty.

swf_placeobject

(PHP 4)

swf_placeobject -- Place an object onto the screen

Description

void **swf_placeobject** (int objid, int depth)

Places the object specified by *objid* in the current frame at a depth of *depth*. The *objid* parameter and the *depth* must be between 1 and 65535.

This uses the current mulcolor (specified by **swf_mulcolor()**) and the current addcolor (specified by **swf_addcolor()**) to color the object and it uses the current matrix to position the object.

Note: Full RGBA colors are supported.

swf_polarview

(PHP 4)

swf_polarview -- Define the viewer's position with polar coordinates

Description

void **swf_polarview** (float dist, float azimuth, float incidence, float twist)

The **swf_polarview()** function defines the viewer's position in polar coordinates. The *dist* parameter gives the distance between the viewpoint to the world space origin. The *azimuth* parameter defines the azimuthal angle in the x,y coordinate plane, measured in distance from the y axis. The *incidence* parameter defines the angle of incidence in the y,z plane, measured in distance from the z axis. The incidence angle is defined as the angle of the viewport relative to the z axis. Finally the *twist* specifies the amount that the viewpoint is to be rotated about the line of sight using the right hand rule.

swf_popmatrix

(PHP 4)

swf_popmatrix -- Restore a previous transformation matrix

Description

void **swf_popmatrix** (void)

The **swf_popmatrix()** function pushes the current transformation matrix back onto the stack.

swf_posround

(PHP 4)

swf_posround -- Enables or Disables the rounding of the translation when objects are placed or moved

Description

void **swf_posround** (int round)

The **swf_posround()** function enables or disables the rounding of the translation when objects are placed or moved, there are times when text becomes more readable because rounding has been enabled. The *round* is whether to enable rounding or not, if set to the value of 1, then rounding is enabled, if set to 0 then rounding is disabled.

swf_pushmatrix

(PHP 4)

swf_pushmatrix -- Push the current transformation matrix back unto the stack

Description

void **swf_pushmatrix** (void)

The **swf_pushmatrix()** function pushes the current transformation matrix back onto the stack.

swf_removeobject

(PHP 4)

swf_removeobject -- Remove an object

Description

void **swf_removeobject** (int depth)

Removes the object at the depth specified by *depth*.

swf_rotate

(PHP 4)

swf_rotate -- Rotate the current transformation

Description

void **swf_rotate** (float angle, string axis)

The **swf_rotate()** rotates the current transformation by the angle given by the *angle* parameter around the axis given by the *axis* parameter. Valid values for the axis are 'x' (the x axis), 'y' (the y axis) or 'z' (the z axis).

swf_scale

(PHP 4)

swf_scale -- Scale the current transformation

Description

void **swf_scale** (float x, float y, float z)

The **swf_scale()** scales the x coordinate of the curve by the value of the *x* parameter, the y coordinate of the curve by the value of the *y* parameter, and the z coordinate of the curve by the value of the *z* parameter.

swf_setfont

(PHP 4)

swf_setfont -- Change the current font

Description

void **swf_setfont** (int fontid)

The **swf_setfont()** sets the current font to the value given by the *fontid* parameter.

swf_setframe

(PHP 4)

swf_setframe -- Switch to a specified frame

Description

void **swf_setframe** (int framenumber)

The **swf_setframe()** changes the active frame to the frame specified by *framenumber*.

swf_shapearc

(PHP 4)

swf_shapearc -- Draw a circular arc

Description

void **swf_shapearc** (float x, float y, float r, float ang1, float ang2)

The **swf_shapeArc()** function draws a circular arc from angle A given by the *ang1* parameter to angle B given by the *ang2* parameter. The center of the circle has an x coordinate given by the *x* parameter and a y coordinate given by the *y*, the radius of the circle is given by the *r* parameter.

swf_shapecurveto3

(PHP 4)

swf_shapecurveto3 -- Draw a cubic bezier curve

Description

void **swf_shapecurveto3** (float x1, float y1, float x2, float y2, float x3, float y3)

Draw a cubic bezier curve using the x,y coordinate pairs *x1*, *y1* and *x2*,*y2* as off curve control points and the x,y coordinate *x3*, *y3* as an endpoint. The current position is then set to the x,y coordinate pair given by *x3*,*y3*.

swf_shapecurveto

(PHP 4)

swf_shapecurveto -- Draw a quadratic bezier curve between two points

Description

void **swf_shapecurveto** (float x1, float y1, float x2, float y2)

The **swf_shapecurveto()** function draws a quadratic bezier curve from the current location, though the x coordinate given by *x1* and the y coordinate given by *y1* to the x coordinate given by *x2* and the y coordinate given by *y2*. The current position is then set to the x,y coordinates given by the *x2* and *y2* parameters

swf_shapefillbitmapclip

(PHP 4)

swf_shapefillbitmapclip -- Set current fill mode to clipped bitmap

Description

void **swf_shapefillbitmapclip** (int bitmapid)

Sets the fill to bitmap clipped, empty spaces will be filled by the bitmap given by the *bitmapid* parameter.

swf_shapefillbitmaptile

(PHP 4)

swf_shapefillbitmaptile -- Set current fill mode to tiled bitmap

Description

void **swf_shapefillbitmaptile** (int bitmapid)

Sets the fill to bitmap tile, empty spaces will be filled by the bitmap given by the *bitmapid* parameter (tiled).

swf_shapefilloff

(PHP 4)

swf_shapefilloff -- Turns off filling

Description

void **swf_shapefilloff** (void)

The **swf_shapeFillOff()** function turns off filling for the current shape.

swf_shapefillsolid

(PHP 4)

swf_shapefillsolid -- Set the current fill style to the specified color

Description

void **swf_shapefillsolid** (float r, float g, float b, float a)

The **swf_shapeFillSolid()** function sets the current fill style to solid, and then sets the fill color to the values of the *rgba* parameters.

swf_shapelinesolid

(PHP 4)

swf_shapelinesolid -- Set the current line style

Description

void **swf_shapelinesolid** (float r, float g, float b, float a, float width)

The **swf_shapeLineSolid()** function sets the current line style to the color of the *rgba* parameters and width to the *width* parameter. If o.o is given as a width then no lines are drawn.

swf_shapelineto

(PHP 4)

swf_shapelineto -- Draw a line

Description

void **swf_shapelineto** (float x, float y)

The **swf_shapeLineTo()** draws a line to the x,y coordinates given by the *x* parameter & the *y* parameter. The current position is then set to the x,y parameters.

swf_shapemoveto

(PHP 4)

swf_shapemoveto -- Move the current position

Description

void **swf_shapemoveto** (float x, float y)

The **swf_shapeMoveTo()** function moves the current position to the x coordinate given by the *x* parameter and the y position given by the *y* parameter.

swf_showframe

(PHP 4)

swf_showframe -- Display the current frame

Description

void **swf_showframe** (void)

The swf_showframe function will output the current frame.

swf_startbutton

(PHP 4)

swf_startbutton -- Start the definition of a button

Description

void **swf_startbutton** (int objid, int type)

The **swf_startbutton()** function starts off the definition of a button. The *type* parameter can either be TYPE_MENUBUTTON or TYPE_PUSHBUTTON. The TYPE_MENUBUTTON constant allows the focus to travel from the button when the mouse is down, TYPE_PUSHBUTTON does not allow the focus to travel when the mouse is down.

swf_startdoaction

(PHP 4)

swf_startdoaction -- Start a description of an action list for the current frame

Description

void **swf_startdoaction** (void)

The **swf_startdoaction()** function starts the description of an action list for the current frame. This must be called before actions are defined for the current frame.

swf_startshape

(PHP 4)

swf_startshape -- Start a complex shape

Description

void **swf_startshape** (int objid)

The **swf_startshape()** function starts a complex shape, with an object id given by the *objid* parameter.

swf_startsymbol

(PHP 4)

swf_startsymbol -- Define a symbol

Description

void **swf_startsymbol** (int objid)

Define an object id as a symbol. Symbols are tiny flash movies that can be played simultaneously. The *objid* parameter is the object id you want to define as a symbol.

swf_textwidth

(PHP 4)

swf_textwidth -- Get the width of a string

Description

float **swf_textwidth** (string str)

The **swf_textwidth()** function gives the width of the string, *str*, in pixels, using the current font and font size.

swf_translate

(PHP 4)

swf_translate -- Translate the current transformations

Description

void **swf_translate** (float x, float y, float z)

The **swf_translate()** function translates the current transformation by the *x*, *y*, and *z* values given.

swf_viewport

(PHP 4)

swf_viewport -- Select an area for future drawing

Description

void **swf_viewport** (float xmin, float xmax, float ymin, float ymax)

The **swf_viewport()** function selects an area for future drawing for *xmin* to *xmax* and *ymin* to *ymax*, if this function is not called the area defaults to the size of the screen.

XCVI. SNMP functions

Introduction

Requirements

In order to use the SNMP functions on Unix you need to install the UCD SNMP package. On Windows these functions are only available on NT and not on Win95/98.

Installation

Important: In order to use the UCD SNMP package, you need to define NO_ZEROLENGTH_COMMUNITY to 1 before compiling it. After configuring UCD SNMP, edit `config.h` and search for NO_ZEROLENGTH_COMMUNITY. Uncomment the #define line. It should look like this afterwards:

```
#define NO_ZEROLENGTH_COMMUNITY 1
```

Now compile PHP `--with-snmp[=DIR]`.

If you see strange segmentation faults in combination with SNMP commands, you did not follow the above instructions. If you do not want to recompile UCD SNMP, you can compile PHP with the `--enable-ucd-snmp-hack` switch which will work around the misfeature.

The Windows distribution contains support files for SNMP in the `mibs` directory. This directory should be moved to `DRIVE:\usr\mibs`, where DRIVE must be replaced with the driveletter where `PHP` is installed on, e.g.: `c:\usr\mibs`.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

Predefined Constants

This extension has no constants defined.

Table of Contents

snmp_get_quick_print

(PHP 3>= 3.0.8, PHP 4)

snmp_get_quick_print -- Fetch the current value of the UCD library's quick_print setting

Description

bool **snmp_get_quick_print** (void)

Returns the current value stored in the UCD Library for quick_print. quick_print is off by default.

```
$quickprint = snmp_get_quick_print();
```

Above function call would return **FALSE** if quick_print is off, and **TRUE** if quick_print is on.

snmp_get_quick_print() is only available when using the UCD SNMP library. This function is not available when using the Windows SNMP library.

See: **snmp_set_quick_print()** for a full description of what quick_print does.

snmp_set_quick_print

(PHP 3>= 3.0.8, PHP 4)

snmp_set_quick_print -- Set the value of quick_print within the UCD SNMP library

Description

void **snmp_set_quick_print** (bool quick_print)

Sets the value of quick_print within the UCD SNMP library. When this is set (1), the SNMP library will return 'quick printed' values. This means that just the value will be printed. When quick_print is not enabled (default) the UCD SNMP library prints extra information including the type of the value (i.e. IpAddress or OID). Additionally, if quick_print is not enabled, the library prints additional hex values for all strings of three characters or less.

Setting quick_print is often used when using the information returned rather then displaying it.

```
snmp_set_quick_print(0);
$a = snmpget("127.0.0.1", "public", ".1.3.6.1.2.1.2.2.1.9.1");
echo "$a<BR>\n";
snmp_set_quick_print(1);
$a = snmpget("127.0.0.1", "public", ".1.3.6.1.2.1.2.2.1.9.1");
echo "$a<BR>\n";
```

The first value printed might be: 'Timeticks: (0) 0:00:00.00', whereas with quick_print enabled, just '0:00:00.00' would be printed.

By default the UCD SNMP library returns verbose values, quick_print is used to return only the value.

Currently strings are still returned with extra quotes, this will be corrected in a later release.

snmp_set_quick_print() is only available when using the UCD SNMP library. This function is not available when using the Windows SNMP library.

snmpget

(PHP 3, PHP 4)

snmpget -- Fetch an SNMP object

Description

string **snmpget** (string hostname, string community, string object_id [, int timeout [, int retries]])

Returns SNMP object value on success and **FALSE** on error.

The **snmpget()** function is used to read the value of an SNMP object specified by the *object_id*. SNMP agent is specified by the *hostname* and the read community is specified by the *community* parameter.

```
$syscontact = snmpget("127.0.0.1", "public", "system.SysContact.0");
```

snmprealwalk

(PHP 3>= 3.0.8, PHP 4)

snmprealwalk -- Return all objects including their respective object ID within the specified one

Description

array **snmprealwalk** (string host, string community, string object_id [, int timeout [, int retries]])

Warning
This function is currently not documented; only the argument list is available.

snmpset

(PHP 3>= 3.0.12, PHP 4)

snmpset -- Set an SNMP object

Description

bool **snmpset** (string hostname, string community, string object_id, string type, mixed value [, int timeout [, int retries]])

Sets the specified SNMP object value, returning **TRUE** on success and **FALSE** on error.

The **snmpset()** function is used to set the value of an SNMP object specified by the *object_id*. SNMP agent is specified by the *hostname* and the read community is specified by the *community* parameter.

snmpwalk

(PHP 3, PHP 4)

snmpwalk -- Fetch all the SNMP objects from an agent

Description

array **snmpwalk** (string hostname, string community, string object_id [, int timeout [, int retries]])

Returns an array of SNMP object values starting from the *object_id* as root and **FALSE** on error.

snmpwalk() function is used to read all the values from an SNMP agent specified by the *hostname*. *Community* specifies the read community for that agent. A **NULL** *object_id* is taken as the root of the SNMP objects tree and all objects under that tree are returned as an array. If *object_id* is specified, all the SNMP objects below that *object_id* are returned.

```
$a = snmpwalk("127.0.0.1", "public", "");
```

Above function call would return all the SNMP objects from the SNMP agent running on localhost. One can step through the values with a loop

```
for ($i=0; $i < count($a); $i++) {
    echo $a[$i];
}
```

snmpwalkoid

(PHP 3>= 3.0.8, PHP 4)

snmpwalkoid -- Query for a tree of information about a network entity

Description

array **snmpwalkoid** (string hostname, string community, string object_id [, int timeout [, int retries]])

Returns an associative array with object ids and their respective object value starting from the *object_id* as root and **FALSE** on error.

snmpwalkoid() function is used to read all object ids and their respective values from an SNMP agent specified by the *hostname*. Community specifies the read *community* for that agent. A **NULL** *object_id* is taken as the root of the SNMP objects tree and all objects under that tree are returned as an array. If *object_id* is specified, all the SNMP objects below that *object_id* are returned.

The existence of **snmpwalkoid()** and **snmpwalk()** has historical reasons. Both functions are provided for backward compatibility.

```
$a = snmpwalkoid("127.0.0.1", "public", "");
```

Above function call would return all the SNMP objects from the SNMP agent running on localhost. One can step through the values with a loop

```
for (reset($a); $i = key($a); next($a)) {
    echo "$i: $a[$i]<br>\n";
}
```

XCVII. Socket functions

Introduction

The socket extension implements a low-level interface to the socket communication functions based on the popular BSD sockets, providing the possibility to act as a socket server as well as a client.

For a more generic client-side socket interface, see **fsockopen()** and **pfsockopen()**.

When using these functions, it is important to remember that while many of them have identical names to their C counterparts, they often have different declarations. Please be sure to read the descriptions to avoid confusion.

Those unfamiliar with socket programming can find a lot of useful material in the appropriate Unix man pages, and there is a great deal of tutorial information on socket programming in C on the web, much of which can be applied, with slight modifications, to socket programming in PHP. The UNIX Socket FAQ might be a good start.

Warning
This extension is *EXPERIMENTAL*. The behaviour of this extension -- including the names of its functions and anything else documented about this extension -- may change without notice in a future release of PHP. Use this extension at your own risk.

Requirements

No external libraries are needed to build this extension.

Installation

The socket functions described here are part of an extension to PHP which must be enabled at compile time by giving the `--enable-sockets` option to **configure**.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

AF_UNIX (integer)

AF_INET (integer)

SOCK_STREAM (integer)

SOCK_DGRAM (integer)

SOCK_RAW (integer)

SOCK_SEQPACKET (integer)

SOCK_RDM (integer)

MSG_OOB (integer)

MSG_WAITALL (integer)

MSG_PEEK (integer)

MSG_DONTROUTE (integer)

SO_DEBUG (integer)

SO_REUSEADDR (integer)

SO_KEEPALIVE (integer)

SO_DONTROUTE (integer)

SO_LINGER (integer)

SO_BROADCAST (integer)

SO_OOBINLINE (integer)

SO_SNDBUF (integer)

SO_RCVBUF (integer)

SO_SNDLOWAT (integer)

SO_RCVLOWAT (integer)

SO_SNDTIMEO (integer)

SO_RCVTIMEO (integer)

SO_TYPE (integer)

SO_ERROR (integer)

SOL_SOCKET (integer)

PHP_NORMAL_READ (integer)

PHP_BINARY_READ (integer)

SOL_TCP (integer)

SOL_UDP (integer)

Socket Errors

The socket extension was written to provide a useable interface to the powerful BSD sockets. Care has been taken that the functions work equally well on Win32 and Unix implementations. Almost all of the sockets functions may fail under certain

conditions and therefore emit an **E_WARNING** message describing the error. Sometimes this doesn't happen to the desire of the developer. For example the function **socket_read()** may suddenly emit an **E_WARNING** message because the connection broke unexpectedly. It's common to suppress the warning with the @-operator and catch the error code within the application with the **socket_last_error()** function. You may call the **socket_strerror()** function with this error code to retrieve a string describing the error. See their description for more information.

> **Note:** The **E_WARNING** messages generated by the socket extension are in english though the retrieved error message will appear depending on the current locale (**LC_MESSAGES**):
>
> ```
> Warning - socket_bind() unable to bind address [98]: Die Adresse wird bereits verwendet
> ```

Examples

Example 1. Socket example: Simple TCP/IP server

This example shows a simple talkback server. Change the `address` and `port` variables to suit your setup and execute. You may then connect to the server with a command similar to: **telnet 192.168.1.53 10000** (where the address and port match your setup). Anything you type will then be output on the server side, and echoed back to you. To disconnect, enter 'quit'.

```
#!/usr/local/bin/php -q
<?php
error_reporting (E_ALL);

/* Allow the script to hang around waiting for connections. */
set_time_limit (0);

/* Turn on implicit output flushing so we see what we're getting
 * as it comes in. */
ob_implicit_flush ();

$address = '192.168.1.53';
$port = 10000;

if (($sock = socket_create (AF_INET, SOCK_STREAM, 0)) < 0) {
    echo "socket_create() failed: reason: " . socket_strerror ($sock) . "\n";
}

if (($ret = socket_bind ($sock, $address, $port)) < 0) {
    echo "socket_bind() failed: reason: " . socket_strerror ($ret) . "\n";
}

if (($ret = socket_listen ($sock, 5)) < 0) {
    echo "socket_listen() failed: reason: " . socket_strerror ($ret) . "\n";
}

do {
    if (($msgsock = socket_accept($sock)) < 0) {
        echo "socket_accept() failed: reason: " . socket_strerror ($msgsock) . "\n";
        break;
    }
    /* Send instructions. */
    $msg = "\nWelcome to the PHP Test Server. \n" .
        "To quit, type 'quit'. To shut down the server type 'shutdown'.\n";
    socket_write($msgsock, $msg, strlen($msg));

    do {
        if (FALSE === ($buf = socket_read ($msgsock, 2048))) {
            echo "socket_read() failed: reason: " . socket_strerror ($ret) . "\n";
            break 2;
        }
        if (!$buf = trim ($buf)) {
            continue;
        }
        if ($buf == 'quit') {
            break;
        }
        if ($buf == 'shutdown') {
            socket_close ($msgsock);
            break 2;
        }
        $talkback = "PHP: You said '$buf'.\n";
        socket_write ($msgsock, $talkback, strlen ($talkback));
        echo "$buf\n";
    } while (true);
    socket_close ($msgsock);
} while (true);

socket_close ($sock);
?>
```

Example 2. Socket example: Simple TCP/IP client

This example shows a simple, one-shot HTTP client. It simply connects to a page, submits a HEAD request, echoes the reply, and

exits.

```
<?php
error_reporting (E_ALL);

echo "<h2>TCP/IP Connection</h2>\n";

/* Get the port for the WWW service. */
$service_port = getservbyname ('www', 'tcp');

/* Get the IP address for the target host. */
$address = gethostbyname ('www.example.com');

/* Create a TCP/IP socket. */
$socket = socket_create (AF_INET, SOCK_STREAM, 0);
if ($socket < 0) {
    echo "socket_create() failed: reason: " . socket_strerror ($socket) . "\n";
} else {
    echo "OK.\n";
}

echo "Attempting to connect to '$address' on port '$service_port'...";
$result = socket_connect ($socket, $address, $service_port);
if ($result < 0) {
    echo "socket_connect() failed.\nReason: ($result) " . socket_strerror($result) . "\n";
} else {
    echo "OK.\n";
}

$in = "HEAD / HTTP/1.0\r\n\r\n";
$out = '';

echo "Sending HTTP HEAD request...";
socket_write ($socket, $in, strlen ($in));
echo "OK.\n";

echo "Reading response:\n\n";
while ($out = socket_read ($socket, 2048)) {
    echo $out;
}

echo "Closing socket...";
socket_close ($socket);
echo "OK.\n\n";
?>
```

Table of Contents

socket_writev -- Writes to a file descriptor, fd, using the scatter-gather array defined by iovec_id

socket_accept

(PHP 4 >= 4.1.0)

socket_accept -- Accepts a connection on a socket

Description

resource **socket_accept** (resource socket)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

After the socket *socket* has been created using **socket_create()**, bound to a name with **socket_bind()**, and told to listen for connections with **socket_listen()**, this function will accept incoming connections on that socket. Once a successful connection is made, a new socket resource is returned, which may be used for communication. If there are multiple connections queued on the socket, the first will be used. If there are no pending connections, **socket_accept()** will block until a connection becomes present. If *socket* has been made non-blocking using **socket_set_blocking()** or **socket_set_nonblock()**, **FALSE** will be returned.

The socket resource returned by **socket_accept()** may not be used to accept new connections. The original listening socket *socket*, however, remains open and may be reused.

Returns a new socket resource on success, or **FALSE** on error. The actual error code can be retrieved by calling **socket_last_error()**. This error code may be passed to **socket_strerror()** to get a textual explanation of the error.

See also **socket_bind()**, **socket_connect()**, **socket_listen()**, **socket_create()**, and **socket_strerror()**.

socket_bind

(PHP 4 >= 4.1.0)

socket_bind -- Binds a name to a socket

Description

bool **socket_bind** (resource socket, string address [, int port])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

socket_bind() binds the name given in *address* to the socket described by *socket*, which must be a valid socket resource created with **socket_create()**.

The *address* parameter is either an IP address in dotted-quad notation (e.g. `127.0.0.1`), if the socket is of the **AF_INET** family; or the pathname of a Unix-domain socket, if the socket family is **AF_UNIX**.

The *port* parameter is only used when connecting to an **AF_INET** socket, and designates the port on the remote host to which a connection should be made.

Returns **TRUE** on success or **FALSE** on failure. The error code can be retrieved with **socket_last_error()**. This code may be passed to **socket_strerror()** to get a textual explanation of the error.

See also **socket_connect()**, **socket_listen()**, **socket_create()**, **socket_last_error()** and **socket_strerror()**.

socket_clear_error

(PHP 4 >= 4.2.0)

socket_clear_error -- Clears the error on the socket or the last error code

Description

void **socket_clear_error** ([resource socket])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

This function clears the error code on the given socket or the global last socket error.

This function allows explicitely resetting the error code value either of a socket or of the extension global last error code. This may be useful to detect within a part of the application if an error occured or not.

See also **socket_last_error()** and **socket_strerror()**.

socket_close

(PHP 4 >= 4.1.0)

socket_close -- Closes a socket resource

Description

void **socket_close** (resource socket)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

socket_close() closes the socket resource given by `socket`.

> **Note: socket_close()** can't be used on PHP file resources created with **fopen()**, **popen()**, **fsockopen()**, or **pfsockopen()**; it is meant for sockets created with **socket_create()** or **socket_accept()**.

See also **socket_bind()**, **socket_listen()**, **socket_create()** and **socket_strerror()**.

socket_connect

(PHP 4 >= 4.1.0)

socket_connect -- Initiates a connection on a socket

Description

bool **socket_connect** (resource socket, string address [, int port])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Initiates a connection using the socket resource `socket`, which must be a valid socket resource created with **socket_create()**.

The `address` parameter is either an IP address in dotted-quad notation (e.g. `127.0.0.1`), if the socket is of the **AF_INET** family; or the pathname of a Unix-domain socket, if the socket family is **AF_UNIX**.

The `port` parameter is only used when connecting to an **AF_INET** socket, and designates the port on the remote host to which a connection should be made.

Returns **TRUE** on success or **FALSE** on failure. The error code can be retrieved with **socket_last_error()**. This code may be passed to **socket_strerror()** to get a textual explanation of the error.

See also **socket_bind()**, **socket_listen()**, **socket_create()**, **socket_last_error()** and **socket_strerror()**.

socket_create_listen

(PHP 4 >= 4.1.0)

socket_create_listen -- Opens a socket on port to accept connections

Description

resource **socket_create_listen** (int port [, int backlog])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

This function is meant to ease the task of creating a new socket which only listens to accept new connections.

socket_create_listen() create a new socket resource of type **AF_INET** listening on *all* local interfaces on the given port waiting for new connections.

The *backlog* parameter defines the maximum length the queue of pending connections may grow to. **SOMAXCONN** may be passed as *backlog* parameter, see **socket_listen()** for more information.

socket_create_listen() returns a new socket resource on success or **FALSE** on error. The error code can be retrieved with **socket_last_error()**. This code may be passed to **socket_strerror()** to get a textual explanation of the error.

> **Note:** If you want to create a socket which only listens on a certain interfaces you need to use **socket_create()**, **socket_bind()** and **socket_listen()**.

See also **socket_create()**, **socket_bind()**, **socket_listen()**, **socket_last_error()** and **socket_strerror()**.

socket_create_pair

(PHP 4 >= 4.1.0)

socket_create_pair -- Creates a pair of indistinguishable sockets and stores them in fds.

Description

bool **socket_create_pair** (int domain, int type, int protocol, array &fd)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

socket_create

(PHP 4 >= 4.1.0)

socket_create -- Create a socket (endpoint for communication)

Description

resource **socket_create** (int domain, int type, int protocol)

Creates and returns a socket resource, also referred to as an endpoint of communication. A typical network connection is made up of 2 sockets, one performing the role of the client, and another performing the role of the server.

The *domain* parameter specifies the protocol family to be used by the socket.

Table 1. Available address/protocol families

Domain	Description
AF_INET	IPv4 Internet based protocols. TCP and UDP are common protocols of this protocol family.
AF_UNIX	Local communication protocol family. High efficiency and low overhead make it a great form of IPC (Interprocess Communication).

The *type* parameter selects the type of communication to be used by the socket.

Table 2. Available socket types

Type	Description
SOCK_STREAM	Provides sequenced, reliable, full-duplex, connection-based byte streams. An out-of-band data transmission mechanism may be supported. The TCP protocol is based on this socket type.
SOCK_DGRAM	Supports datagrams (connectionless, unreliable messages of a fixed maximum length). The UDP protocol is based on this socket type.
SOCK_SEQPACKET	Provides a sequenced, reliable, two-way connection-based data transmission path for datagrams of fixed maximum length; a consumer is required to read an entire packet with each read call.
SOCK_RAW	Provides raw network protocol access. This special type of socket can be used to manually construct any type of protocol. A common use for this socket type is to perform ICMP requests (like ping, traceroute, etc).
SOCK_RDM	Provides a reliable datagram layer that does not guarantee ordering. This is most likely not implemented on your operating system.

The *protocol* parameter sets the specific protocol within the specified *domain* to be used when communicating on the returned socket. The proper value can be retrieved by name by using **getprotobyname()**. If the desired protocol is TCP, or UDP the corresponding constants **SOL_TCP**, and **SOL_UDP** can also be used.

Table 3. Common protocols

Name	Description
icmp	The Internet Control Message Protocol is used primarily by gateways and hosts to report errors in datagram communication. The "ping" command (present in most modern operating systems) is an example application of ICMP.
udp	The User Datagram Protocol is a connectionless, unreliable, protocol with fixed record lengths. Due to these aspects, UDP requires a minimum amount of protocol overhead.
tcp	The Transmission Control Protocol is a reliable, connection based, stream oriented, full duplex protocol. TCP guarantees that all data packets will be received in the order in which they were sent. If any packet is somehow lost during communication, TCP will automatically retransmit the packet until the destination host acknowledges that packet. For reliability and performance reasons, the TCP implementation itself decides the appropriate octet boundaries of the underlying datagram communication layer. Therefore, TCP applications must allow for the possibility of partial record transmission.

socket_create() Returns a socket resource on success, or **FALSE** on error. The actual error code can be retrieved by calling **socket_last_error()**. This error code may be passed to **socket_strerror()** to get a textual explanation of the error.

> **Note:** If an invalid *domain* or *type* is given, **socket_create()** defaults to **AF_INET** and **SOCK_STREAM** respectively and additionally emits an **E_WARNING** message.

See also **socket_accept()**, **socket_bind()**, **socket_connect()**, **socket_listen()**, **socket_last_error()**, and **socket_strerror()**.

socket_get_option

(PHP 4 >= 4.3.0)

socket_get_option -- Gets socket options for the socket

Description

mixed **socket_get_option** (resource socket, int level, int optname)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

Note: This function used to be called `socket_getopt()` prior to PHP 4.3.0

socket_getpeername

(PHP 4 >= 4.1.0)

socket_getpeername -- Queries the remote side of the given socket which may either result in host/port or in a UNIX filesystem path, dependent on its type.

Description

bool **socket_getpeername** (resource socket, string &addr [, int &port])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

If the given socket is of type **AF_INET**, **socket_getpeername()** will return the peers (remote) *IP address* in dotted-quad notation (e.g. `127.0.0.1`) in the *address* parameter and, if the optional *port* parameter is present, also the associated port.

If the given socket is of type **AF_UNIX**, **socket_getpeername()** will return the UNIX filesystem path (e.g. `/var/run/daemon.sock`) in the *address* parameter.

Returns **TRUE** on success or **FALSE** on failure. **socket_getpeername()** may also return **FALSE** if the socket type is not any of **AF_INET** or **AF_UNIX**, in which case the last socket error code is *not* updated.

See also **socket_getpeername()**, socket_last_error() and socket_strerror().

socket_getsockname

(PHP 4 >= 4.1.0)

socket_getsockname -- Queries the local side of the given socket which may either result in host/port or in a UNIX filesystem path, dependent on its type.

Description

bool **socket_getsockname** (resource socket, string &addr [, int &port])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

If the given socket is of type **AF_INET**, **socket_getsockname()** will return the local *IP address* in dotted-quad notation (e.g. `127.0.0.1`) in the *address* parameter and, if the optional *port* parameter is present, also the associated port.

If the given socket is of type **AF_UNIX**, **socket_getsockname()** will return the UNIX filesystem path (e.g. `/var/run/daemon.sock`) in the *address* parameter.

Returns **TRUE** on success or **FALSE** on failure. **socket_getsockname()** may also return **FALSE** if the socket type is not any of **AF_INET** or **AF_UNIX**, in which case the last socket error code is *not* updated.

See also socket_getpeername(), socket_last_error() and socket_strerror().

socket_iovec_add

(PHP 4 >= 4.1.0)

socket_iovec_add -- Adds a new vector to the scatter/gather array

Description

bool **socket_iovec_add** (resource iovec, int iov_len)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

socket_iovec_alloc

(PHP 4 >= 4.1.0)

socket_iovec_alloc -- ...]) Builds a 'struct iovec' for use with sendmsg, recvmsg, writev, and readv

Description

resource **socket_iovec_alloc** (int num_vectors [, int])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

socket_iovec_delete

(PHP 4 >= 4.1.0)

socket_iovec_delete -- Deletes a vector from an array of vectors

Description

bool **socket_iovec_delete** (resource iovec, int iov_pos)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

socket_iovec_fetch

(PHP 4 >= 4.1.0)

socket_iovec_fetch -- Returns the data held in the iovec specified by iovec_id[iovec_position]

Description

string **socket_iovec_fetch** (resource iovec, int iovec_position)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

socket_iovec_free

(PHP 4 >= 4.1.0)

socket_iovec_free -- Frees the iovec specified by iovec_id

Description

bool **socket_iovec_free** (resource iovec)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

socket_iovec_set

(PHP 4 >= 4.1.0)

socket_iovec_set -- Sets the data held in iovec_id[iovec_position] to new_val

Description

bool **socket_iovec_set** (resource iovec, int iovec_position, string new_val)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

socket_last_error

(PHP 4 >= 4.1.0)

socket_last_error -- Returns the last error on the socket

Description

int **socket_last_error** ([resource socket])

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

This function returns a socket error code.

If a socket resource is passed to this function, the last error which occured on this particular socket is returned. If the socket

resource is ommited, the error code of the last failed socket function is returned. The latter is in particular helpful for functions like socket_create() which don't return a socket on failure and socket_select() which can fail for reasons not directly tied to a particular socket. The error code is suitable to be fed to socket_strerror() which returns a string describing the given error code.

```
if (false == ($socket = @socket_create(AF_INET, SOCK_STREAM, SOL_TCP))) {
    die("Couldn't create socket, error code is: " . socket_last_error() .
        ",error message is: " . socket_strerror(socket_last_error()));
}
```

Note: socket_last_error() does not clear the error code, use socket_clear_error() for this purpose.

socket_listen

(PHP 4 >= 4.1.0)

socket_listen -- Listens for a connection on a socket

Description

bool **socket_listen** (resource socket [, int backlog])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

After the socket *socket* has been created using socket_create() and bound to a name with socket_bind(), it may be told to listen for incoming connections on *socket*.

A maximum of *backlog* incoming connections will be queued for processing. If a connection request arrives with the queue full the client may receive an error with an indication of ECONNREFUSED, or, if the underlying protocol supports retransmission, the request may be ignored so that retries may succeed.

> **Note:** The maximum number passed to the *backlog* parameter highly depends on the underlying platform. On linux, it is silently truncated to **SOMAXCONN**. On win32, if passed **SOMAXCONN**, the underlying service provider responsible for the socket will set the backlog to a maximum *reasonable* value. There is no standard provision to find out the actual backlog value on this platform.

socket_listen() is applicable only to sockets of type **SOCK_STREAM** or **SOCK_SEQPACKET**.

Returns **TRUE** on success or **FALSE** on failure. The error code can be retrieved with socket_last_error(). This code may be passed to socket_strerror() to get a textual explanation of the error.

See also socket_accept(), socket_bind(), socket_connect(), socket_create() and socket_strerror().

socket_read

(PHP 4 >= 4.1.0)

socket_read -- Reads a maximum of length bytes from a socket

Description

string **socket_read** (resource socket, int length [, int type])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

The function **socket_read()** reads from the socket resource *socket* created by the socket_create() or socket_accept() functions. The maximum number of bytes read is specified by the *length* parameter. Otherwise you can use \r, \n, or \o to end reading (depending on the *type* parameter, see below).

socket_read() returns the data as a string on success, or **FALSE** on error. The error code can be retrieved with socket_last_error(). This code may be passed to socket_strerror() to get a textual representation of the error.

> **Note: socket_read()** may return a zero length string ("") indicating the end of communication (i.e. the remote end

point has closed the connection).

Optional *type* parameter is a named constant:

- PHP_BINARY_READ - use the system `read()` function. Safe for reading binary data. (Default in PHP >= 4.1.0)
- PHP_NORMAL_READ - reading stops at \n or \r. (Default in PHP <= 4.0.6)

See also **socket_accept()**, **socket_bind()**, **socket_connect()**, **socket_listen()**, **socket_last_error()**, **socket_strerror()** and **socket_write()**.

socket_readv

(PHP 4 >= 4.1.0)

socket_readv -- Reads from an fd, using the scatter-gather array defined by iovec_id

Description

bool **socket_readv** (resource socket, resource iovec_id)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

socket_recv

(PHP 4 >= 4.1.0)

socket_recv -- Receives data from a connected socket

Description

string **socket_recv** (resource socket, int len, int flags)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

socket_recvfrom

(PHP 4 >= 4.1.0)

socket_recvfrom -- Receives data from a socket, connected or not

Description

int **socket_recvfrom** (resource socket, string &buf, int len, int flags, string &name [, int &port])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

socket_recvmsg

(PHP 4 >= 4.1.0)

socket_recvmsg -- Used to receive messages on a socket, whether connection-oriented or not

Description

bool **socket_recvmsg** (resource socket, resource iovec, array &control, int &controllen, int &flags, string &addr [, int &port])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

socket_select

(PHP 4 >= 4.1.0)

socket_select -- Runs the select() system call on the given arrays of sockets with a timeout specified by tv_sec and tv_usec

Description

int **socket_select** (resource &read, resource &write, resource &except, int tv_sec [, int tv_usec])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

The **socket_select()** accepts arrays of sockets and waits for them to change status. Those coming with BSD sockets background will recognize that those socket resource arrays are in fact the so-called file descriptor sets. Three independent arrays of socket resources are watched.

The sockets listed in the *read* array will be watched to see if characters become available for reading (more precisely, to see if a read will not block - in particular, a socket resource is also ready on end-of-file, in which case a **socket_read()** will return a zero length string).

The sockets listed in the *write* array will be watched to see if a write will not block.

The sockets listed in the *except* array will be watched for exceptions.

Warning
On exit, the arrays are modified to indicate which socket resource actually changed status.

You do not need to pass every array to **socket_select()**. You can leave it out and use an empty array or **NULL** instead. Also do not forget that those arrays are passed *by reference* and will be modified after **socket_select()** returns.

Example:

```
/* Prepare the read array */
$read = array($socket1, $socket2);

if (false === ($num_changed_sockets = socket_select($read, $write = NULL, $except = NULL, 0))) {
    /* Error handling */
else if ($num_changed_sockets > 0) {
    /* At least at one of the sockets something interesting happened */
}
```

Note: Due a limitation in the current Zend Engine it is not possible to pass a constant modifier like **NULL** directly as a

parameter to a function which expects this parameter to be passed by reference. Instead use a temporary variable or an expression with the leftmost member being a temporary variable:

```
socket_select($r, $w, $e = NULL, 0);
```

The *tv_sec* and *tv_usec* together form the *timeout* parameter. The *timeout* is an upper bound on the amount of time elapsed before **socket_select()** return. *tv_sec* may be zero , causing **socket_select()** to return immediately. This is useful for polling. If *tv_sec* is **NULL** (no timeout), **socket_select()** can block indefinitely.

On success **socket_select()** returns the number of socket resorces contained in the modified arrays, which may be zero if the timeout expires before anything interesting happens. On error **FALSE** is returned. The error code can be retrieved with socket_last_error().

Note: Be sure to use the === operator when checking for an error. Since the **socket_select()** may return o the comparison with == would evaluate to **TRUE**:

```
if (false === socket_select($r, $w, $e = NULL, 0)) {
    echo "socket_select() failed, reason: " . socket_strerror(socket_last_error()) . "\n";
}
```

Note: Be aware that some socket implementations need to be handled very carefully. A few basic rules:

- You should always try to use **socket_select()** without timeout. Your program should have nothing to do if there is no data available. Code that depends on timeouts is not usually portable and difficult to debug.
- No socket resource must be added to any set if you do not intend to check its result after the **socket_select()** call, and respond appropriately. After **socket_select()** returns, all socket resources in all arrays must be checked. Any socket resource that is available for writing must be written to, and any socket resource available for reading must be read from.
- If you read/write to a socket returns in the arrays be aware that they do not necessarily read/write the full amount of data you have requested. Be prepared to even only be able to read/write a single byte.
- It's common to most socket implementations that the only exception caught with the *except* array is out-of-bound data received on a socket.

See also socket_read(), socket_write(), socket_last_error() and socket_strerror().

socket_send

(PHP 4 >= 4.1.0)

socket_send -- Sends data to a connected socket

Description

int **socket_send** (resource socket, string buf, int len, int flags)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

socket_sendmsg

(PHP 4 >= 4.1.0)

socket_sendmsg -- Sends a message to a socket, regardless of whether it is connection-oriented or not

Description

bool **socket_sendmsg** (resource socket, resource iovec, int flags, string addr [, int port])

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

socket_sendto

(PHP 4 >= 4.1.0)

socket_sendto -- Sends a message to a socket, whether it is connected or not

Description

int **socket_sendto** (resource socket, string buf, int len, int flags, string addr [, int port])

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

socket_set_nonblock

(PHP 4 >= 4.1.0)

socket_set_nonblock -- Sets nonblocking mode for file descriptor fd

Description

bool **socket_set_nonblock** (resource socket)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

socket_set_option

(PHP 4 >= 4.3.0)

socket_set_option -- Sets socket options for the socket

Description

bool **socket_set_option** (resource socket, int level, int optname, int)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

Note: This function used to be called `socket_setopt()` prior to PHP 4.3.0

socket_shutdown

(PHP 4 >= 4.1.0)

socket_shutdown -- Shuts down a socket for receiving, sending, or both.

Description

bool **socket_shutdown** (resource socket [, int how])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

socket_strerror

(PHP 4 >= 4.1.0)

socket_strerror -- Return a string describing a socket error

Description

string **socket_strerror** (int errno)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

socket_strerror() takes as its *errno* parameter a socket error code as returned by **socket_last_error()** and returns the corresponding explanatory text. This makes it a bit more pleasant to figure out why something didn't work; for instance, instead of having to track down a system include file to find out what '-111' means, you just pass it to **socket_strerror()**, and it tells you what happened.

Example 1. socket_strerror() example

```
<?php
if (false == ($socket = @socket_create(AF_INET, SOCK_STREAM, 0))) {
    echo "socket_create() failed: reason: " . socket_strerror(socket_last_error()) . "\n";
}

if (false == (@socket_bind($socket, '127.0.0.1', 80))) {
    echo "socket_bind() failed: reason: " . socket_strerror(socket_last_error($socket)) . "\n";
}
?>
```

The expected output from the above example (assuming the script is not run with root privileges):

```
socket_bind() failed: reason: Permission denied
```

See also **socket_accept()**, **socket_bind()**, **socket_connect()**, **socket_listen()**, and **socket_create()**.

socket_write

(PHP 4 >= 4.1.0)

socket_write -- Write to a socket

Description

int **socket_write** (resource socket, string buffer [, int length])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

The function **socket_write()** writes to the socket *socket* from *buffer*.

The optional parameter *length* can specify an alternate length of bytes written to the socket. If this length is greater then the buffer length, it is silently truncated to the length of the buffer.

Returns the number of bytes successfully written to the socket or **FALSE** one error. The error code can be retrieved with **socket_last_error()**. This code may be passed to **socket_strerror()** to get a textual explanation of the error.

> **Note: socket_write()** does not necessarily write all bytes from the given buffer. It's valid that, depending on the network buffers etc., only a certain amount of data, even one byte, is written though your buffer is greater. You have to watch out so you don't unintentionally forget to transmit the rest of your data.
>
> **Note:** It is perfectly valid for **socket_write()** to return zero which means no bytes have been written. Be sure to use the === operator to check for **FALSE** in case of an error.

See also **socket_accept()**, **socket_bind()**, **socket_connect()**, **socket_listen()**, **socket_read()** and **socket_strerror()**.

socket_writev

(PHP 4 >= 4.1.0)

socket_writev -- Writes to a file descriptor, fd, using the scatter-gather array defined by iovec_id

Description

bool **socket_writev** (resource socket, resource iovec_id)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

XCVIII. Stream functions

Introduction

Streams were introduced with PHP 4.3.0 as a way of generalizing file, network, data compression, and other opperations which share a common set of functions and uses. In its simplest definition, a `stream` is a `resource` object which exhibits streamable behavior. That is, it can be read from or written to in a linear fashion, and may be able to **fseek()** to an arbitrary locations within the stream.

A `wrapper` is additional code which tells the stream how to handle specific protocols/encodings. For example, the `http` wrapper knows how to translate a URL into an `HTTP/1.0` request for a file on a remote server. There are many wrappers built into PHP by default (See Appendix I), and additional, custom wrappers may be added either within a PHP script using **stream_register_wrapper()**, or directly from an extension using the API Reference in Chapter 43. Because any variety of wrapper may be added to PHP, there is no set limit on what can be done with them. To access the list of currently registered wrappers, use **stream_get_wrappers()**.

A `filter` is a final piece of code which may perform opperations on data as it is being read from or written to a stream. Any number of filters may be stacked onto a stream. Custom filters can be defined in a PHP script using **stream_register_filter()** or in an extension using the API Reference in Chapter 43. To access the list of currently registered filters, use **stream_get_filters()**.

A stream is referenced as: *scheme*://*target*

- *scheme*(string) - The name of the wrapper to be used. Examples include: file, http, https, ftp, ftps, compress.zlib, compress.bz2, and php. See Appendix I for a list of PHP builtin wrappers. If no wrapper is specified, the function default is used (typically `file://`).
- *target* - Depends on the wrapper used. For filesystem related streams this is typically a path and filename of the desired file. For network related streams this is typically a hostname, often with a path appended. Again, see Appendix I for a description of targets for builtin streams.

Requirements

No external libraries are needed to build this extension.

Installation

Streams are an integral part of PHP as of version 4.3.0. No steps are required to enable them.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Stream Classes

User designed wrappers can be registered via **stream_register_wrapper()**, using the class definition shown on that manual page.

`class` php_user_filter is predefined and is an abstract baseclass for use with user defined filters. See the manual page for **stream_register_filter()** for details on implementing user defined filters.

Predefined Constants

Constant	Description
STREAM_USE_PATH	`Flag` indicating if the `stream` used the include path.
STREAM_REPORT_ERRORS	`Flag` indicating if the `wrapper` is responsible for raising errors using **trigger_error()** during opening of the stream. If this flag is not set, you should not raise any errors.

Stream Errors

As with any file or socket related function, an opperation on a stream may fail for a variety of normal reasons (i.e.: Unable to connect to remote host, file not found, etc...). A stream related call may also fail because the desired stream is not registered on the running system. See the array returned by **stream_get_wrappers()** for a list of streams supported by your installation of PHP. As with most PHP internal functions if a failure occours an **E_WARNING** message will be generated describing the nature of the error.

Examples

Example 1. Using file_get_contents() to retrieve data from multiple sources

```
<?php
/* Read local file from /home/bar */
$localfile = file_get_contents("/home/bar/foo.txt");

/* Identical to above, explicitly naming FILE scheme */
$localfile = file_get_contents("file:///home/bar/foo.txt");

/* Read remote file from www.example.com using HTTP */
$httpfile  = file_get_contents("http://www.example.com/foo.txt");

/* Read remote file from www.example.com using HTTPS */
$httpsfile = file_get_contents("https://www.example.com/foo.txt");

/* Read remote file from ftp.example.com using FTP */
$ftpfile   = file_get_contents("ftp://user:pass@ftp.example.com/foo.txt");

/* Read remote file from ftp.example.com using FTPS */
$ftpsfile  = file_get_contents("ftps://user:pass@ftp.example.com/foo.txt");
?>
```

Example 2. Making a POST request to an https server

```
<?php
/* Send POST request to https://secure.example.com/form_action.php
 * Include form elements named "foo" and "bar" with dummy values
 */

$sock = fsockopen("ssl://secure.example.com", 443, $errno, $errstr, 30);
if (!$sock) die("$errstr ($errno)\n");

$data = "foo=" . urlencode("Value for Foo") . "&bar=" . urlencode("Value for Bar");

fputs($sock, "POST /form_action.php HTTP/1.0\r\n");
fputs($sock, "Host: secure.example.com\r\n");
fputs($sock, "Content-type: application/x-www-url-encoded\r\n");
fputs($sock, "Content-length: " . strlen($data) . "\r\n");
fputs($sock, "Accept: */*\r\n");
fputs($sock, "\r\n");
fputs($sock, "$data\r\n");
fputs($sock, "\r\n");

$headers = "";
while ($str = trim(fgets($sock, 4096)))
  $headers .= "$str\n";

print "\n";

$body = "";
while (!feof($sock))
  $body .= fgets($sock, 4096);

fclose($sock);
?>
```

Example 3. Writting data to a compressed file

```
<?php
/* Create a compressed file containing an arbitrarty string
 * File can be read back using compress.zlib stream or just
 * decompressed from the command line using 'gzip -d foo-bar.txt.gz'
 */
$fp = fopen("compress.zlib://foo-bar.txt.gz","wb");
if (!$fp) die("Unable to create file.");

fwrite($fp, "This is a test.\n");

fclose($fp);
?>
```

Table of Contents

stream_set_timeout -- Set timeout period on a stream
stream_set_write_buffer -- Sets file buffering on the given stream

stream_context_create

(PHP 4 >= 4.3.0)

stream_context_create -- Create a streams context

Description

resource **stream_context_create** (array params)

Creates and returns a stream context with any parameters supplied in *params* preset.

See Also: **stream_context_set_params()**

stream_context_get_options

(PHP 4 >= 4.3.0)

stream_context_get_options -- Retrieve options for a stream/wrapper/context

Description

bool **stream_context_get_options** (resource stream|context)

Returns an array of options on the specified stream or context.

stream_context_set_option

(PHP 4 >= 4.3.0)

stream_context_set_option -- Sets an option for a stream/wrapper/context

Description

bool **stream_context_set_option** (resource context|stream, string wrapper, string option, mixed value)

Sets an option on the specified context. *value* is set to *option* for *wrapper*

stream_context_set_params

(PHP 4 >= 4.3.0)

stream_context_set_params -- Set parameters for a stream/wrapper/context

Description

bool **stream_context_set_params** (resource stream|context, array params)

params should be an associative array of the structure: `$params['paramname'] = "paramvalue";`.

Table 1. Parameters

Parameters	Purpose
`notification`	Name of user-defined callback function to be called whenever a stream triggers a notification.

stream_filter_append

(PHP 4 >= 4.3.0)

stream_filter_append -- Attach a filter to a stream.

Description

bool **stream_filter_append** (resource stream, string filtername [, string params])

Adds *filtername* to the list of filters attached to *stream*. This filter will be added with the specified *params* to the *end* of the list and will therefore be called last during stream opperations. To add a filter to the beginning of the list, use **stream_filter_prepend()**.

> **Note:** **stream_register_filter()** must be called first in order to register the desired user filter to *filtername*.

See Also: **stream_register_filter()**, and **stream_filter_prepend()**

stream_filter_prepend

(PHP 4 >= 4.3.0)

stream_filter_prepend -- Attach a filter to a stream.

Description

bool **stream_filter_prepend** (resource stream, string filtername [, string params])

Adds *filtername* to the list of filters attached to *stream*. This filter will be added with the specified *params* to the *beginning* of the list and will therefore be called first during stream opperations. To add a filter to the end of the list, use **stream_filter_append()**.

> **Note:** **stream_register_filter()** must be called first in order to register the desired user filter to *filtername*.

See Also: **stream_register_filter()**, and **stream_filter_append()**

stream_get_filters

(PHP 5 CVS only)

stream_get_filters -- Retrieve list of registered filters

Description

array **stream_get_filters** (void)

Returns an indexed array containing the name of all stream filters available on the running system.

Example 1. Using stream_get_filters()

```
<?php
$streamlist = stream_get_filters();
print_r($streamlist);

/* Output will be similar to the following
   Note: there may be more or fewer
         filters in your version of PHP
   -------------------------------------
Array (
  [0] => string.rot13
  [1] => string.toupper
  [2] => string.tolower
  [3] => string.base64
  [4] => string.quoted-printable
)
 */
?>
```

See Also: **stream_register_filter()**, and **stream_get_wrappers()**

stream_get_meta_data

(PHP 4 >= 4.3.0)

stream_get_meta_data -- Retrieves header/meta data from streams/file pointers

Description

array **stream_get_meta_data** (resource stream)

Returns information about an existing *stream*. The stream can be any stream created by **fopen()**, **fsockopen()** and **pfsockopen()**. The result array contains the following items:

- *timed_out* (bool) - **TRUE** if the stream timed out while waiting for data on the last call to **fread()** or **fgets()**.
- *blocked* (bool) - **TRUE** if the stream is in blocking IO mode. See **socket_set_blocking()**.
- *eof* (bool) - **TRUE** if the stream has reached end-of-file. Note that for socket streams this member can be **TRUE** even when *unread_bytes* is non-zero. To determine if there is more data to be read, use **feof()** instead of reading this item.
- *unread_bytes* (int) - the number of bytes currently contained in the read buffer.

The following items were added in PHP 4.3:

- *stream_type* (string) - a label describing the underlying implementation of the stream.
- *wrapper_type* (string) - a label describing the protocol wrapper implementation layered over the stream. See Appendix I for more information about wrappers.
- *wrapper_data* (mixed) - wrapper specific data attached to this stream. See Appendix I for more information about wrappers and their wrapper data.
- *filters* (array) - and array containing the names of any filters that have been stacked onto this stream. Filters are currently undocumented.

  Note: This function was introduced in PHP 4.3, but prior to this version, **socket_get_status()** could be used to retrieve the first four items, for *socket based streams only*.

  In PHP 4.3 and later, **socket_get_status()** is an alias for this function.

  Note: This function does NOT work on sockets created by the Socket extension.

stream_get_wrappers

(PHP 5 CVS only)

stream_get_wrappers -- Retrieve list of registered streams

Description

array **stream_get_wrappers** (void)

Returns an indexed array containing the name of all stream wrappers available on the running system.

See Also: **stream_register_wrapper()**

stream_register_filter

(PHP 5 CVS only)

stream_register_filter -- Register a stream filter implemented as a PHP class derived from `php_user_filter`

Description

boolean **stream_register_filter** (string filtername, string classname)

stream_register_filter() allows you to implement your own filter on any registered stream used with all the other filesystem functions (such as **fopen()**, **fread()** etc.).

To implement a filter, you need to define a class as an extension of `php_user_fitler` with a number of member functions as defined below. When performing read/write opperations on the stream to which your filter is attached, PHP will pass the data through your filter (and any other filters attached to that stream) so that the data may be modified as desired. You must implement the methods exactly as described below - doing otherwise will lead to undefined behaviour.

stream_register_filter() will return **FALSE** if the *filtername* is already defined.

int **write** (string data)

This method is called whenever data is written to the attached stream (such as with **fwrite()**). After modifying *data* as needed your filter should issue: `return parent::write($data);` so that the next filter in the chain can perform its filter. When no filters remain, the stream will write *data* in its final form.

> **Note:** If your filter alters the length of *data*, for example by removing the first character, before passing onto `parent::write($data);` it must be certain to include that stolen character in the return count.

```
class myfilter extends php_user_filter {
  function write($data) {
    $data = substr($data,1);
    $written_by_parent = parent::write($data);
    return ($written_by_parent + 1);
  }
}
```

string **read** (int maxlength)

This method is called whenever data is read from the attached stream (such as with **fread()**). A filter should first call `parent::read($maxlength);` to retrieve the data from the previous filter who, ultimately, retrieved it from the stream. Your filter may then modify the data as needed and `return` it. Your filter should never return more than *maxlength* bytes. Since `parent::read($maxlength);` will also not return more than *maxlength* bytes this will ordinarily be a non-issue. However, if your filter increases the size of the data being returned, you should either call `parent::read($maxlength-$x);` where *x* is the most your filter will grow the size of the data read. Alternatively, you can build a read-buffer into your class.

int **flush** (bool closing)

This method is called in response to a request to flush the attached stream (such as with **fflush()** or **fclose()**). The *closing* parameter tells you whether the stream is, in fact, in the process of closing. The default action is to simply call: `return parent::flush($closing);` , your filter may wish to perform additional writes and/or cleanup calls prior to or directly after a successful flush.

void **oncreate** (void)

This method is called during instantiation of the filter class object. If your filter allocates or initializes any other resources (such as a buffer), this is the place to do it.

void **onclose** (void)

This method is called upon filter shutdown (typically, this is also during stream shutdown), and is executed *after* the `flush` method is called. If any resources were allocated or initialzed during `oncreate` this would be the time to destroy or dispose of them.

The example below implements a filter named `rot13` on the `foo-bar.txt` stream which will perform ROT-13 encryption on all letter characters written to/read from that stream.

Example 1. Filter for ROT13 encoding data on foo-bar.txt stream

```
<?php

/* Define our filter class */
class rot13_filter extends php_user_filter {
  function read($length) {
    $tempstr = parent::read($length);
```

```
    for($i = 0; $i < strlen($tempstr); $i++)
      if (($tempstr[$i] >= 'A' AND $tempstr[$i] <= 'M') OR
          ($tempstr[$i] >= 'a' AND $tempstr[$i] <= 'm')) $tempstr[$i] = chr(ord($tempstr[$i]) + 13);
      else if (($tempstr[$i] >= 'N' AND $tempstr[$i] <= 'Z') OR
               ($tempstr[$i] >= 'n' AND $tempstr[$i] <= 'z')) $tempstr[$i] = chr(ord($tempstr[$i]) - 13);
    return $tempstr;
  }

  function write($data) {
    for($i = 0; $i < strlen($data); $i++)
      if (($data[$i] >= 'A' AND $data[$i] <= 'M') OR
          ($data[$i] >= 'a' AND $data[$i] <= 'm')) $data[$i] = chr(ord($data[$i]) + 13);
      else if (($data[$i] >= 'N' AND $data[$i] <= 'Z') OR
               ($data[$i] >= 'n' AND $data[$i] <= 'z')) $data[$i] = chr(ord($data[$i]) - 13);
    return parent::write($data);
  }
}

/* Register our filter with PHP */
stream_register_filter("rot13", "rot13_filter")
    or die("Failed to register filter");

$fp = fopen("foo-bar.txt", "w");

/* Attach the registered filter to the stream just opened */
stream_filter_append($fp, "rot13");

fwrite($fp, "Line1\n");
fwrite($fp, "Word - 2\n");
fwrite($fp, "Easy As 123\n");

fclose($fp);

/* The filter only applies to the $fp stream
 * so this readfile will read -without- applying
 * a second pass of rot13 encoding
 */
readfile("foo-bar.txt");

/* Output
 * ------

Yvar1
Jbeq - 2
Rnfl Nf 123

 */
?>
```

See Also: **stream_register_wrapper()**, **stream_filter_prepend()**, and **stream_filter_append()**

stream_register_wrapper

(PHP 4 >= 4.3.0)

stream_register_wrapper -- Register a URL wrapper implemented as a PHP class

Description

boolean **stream_register_wrapper** (string protocol, string classname)

stream_register_wrapper() allows you to implement your own protocol handlers and streams for use with all the other filesystem functions (such as **fopen()**, **fread()** etc.).

To implement a wrapper, you need to define a class with a number of member functions, as defined below. When someone fopens your stream, PHP will create an instance of *classname* and then call methods on that instance. You must implement the methods exactly as described below - doing otherwise will lead to undefined behaviour.

stream_register_wrapper() will return **FALSE** if the *protocol* already has a handler.

boolean **stream_open** (string path, string mode, int options, string opened_path)

This method is called immediately after your stream object is created. *path* specifies the URL that was passed to **fopen()** and that this object is expected to retrieve. You can use **parse_url()** to break it apart.

mode is the mode used to open the file, as detailed for **fopen()**. You are responsible for checking that *mode* is valid for the *path* requested.

options holds additional flags set by the streams API. It can hold one or more of the following values OR'd together.

Flag	Description
STREAM_USE_PATH	If *path* is relative, search for the resource using the include_path.
STREAM_REPORT_ERRORS	If this flag is set, you are responsible for raising errors using **trigger_error()** during opening of the stream. If this flag is not set, you should not raise any errors.

If the *path* is opened successfully, and STREAM_USE_PATH is set in *options*, you should set *opened_path* to the full path of the file/resource that was actually opened.

If the requested resource was opened successfully, you should return **TRUE**, otherwise you should return **FALSE**

void **stream_close** (void)

This method is called when the stream is closed, using **fclose()**. You must release any resources that were locked or allocated by the stream.

string **stream_read** (int count)

This method is called in response to **fread()** and **fgets()** calls on the stream. You must return up-to *count* bytes of data from the current read/write position as a string. If there are less than *count* bytes available, return as many as are available. If no more data is available, return either **FALSE** or an empty string. You must also update the read/write position of the stream by the number of bytes that were successfully read.

int **stream_write** (string data)

This method is called in response to **fwrite()** calls on the stream. You should store *data* into the underlying storage used by your stream. If there is not enough room, try to store as many bytes as possible. You should return the number of bytes that were successfully stored in the stream, or o if none could be stored. You must also update the read/write position of the stream by the number of bytes that were successfully written.

boolean **stream_eof** (void)

This method is called in response to **feof()** calls on the stream. You should return **TRUE** if the read/write position is at the end of the stream and if no more data is available to be read, or **FALSE** otherwise.

int **stream_tell** (void)

This method is called in response to **ftell()** calls on the stream. You should return the current read/write position of the stream.

boolean **stream_seek** (int offset, int whence)

This method is called in response to **fseek()** calls on the stream. You should update the read/write position of the stream according to *offset* and *whence*. See **fseek()** for more information about these parameters. Return **TRUE** if the position was updated, **FALSE** otherwise.

boolean **stream_flush** (void)

This method is called in response to **fflush()** calls on the stream. If you have cached data in your stream but not yet stored it into the underlying storage, you should do so now. Return **TRUE** if the cached data was successfully stored (or if there was no data to store), or **FALSE** if the data could not be stored.

The example below implements a var:// protocol handler that allows read/write access to a named global variable using standard filesystem stream functions such as **fread()**. The var:// protocol implemented below, given the url "var://foo" will read/write data to/from $GLOBALS["foo"].

Example 1. A Stream for reading/writing global variables

```
class VariableStream {
    var $position;
    var $varname;

    function stream_open($path, $mode, $options, &$opened_path)
    {
        $url = parse_url($path);
        $this->varname = $url["host"];
        $this->position = 0;

        return true;
```

```
    }

    function stream_read($count)
    {
        $ret = substr($GLOBALS[$this->varname], $this->position, $count);
        $this->position += strlen($ret);
        return $ret;
    }

    function stream_write($data)
    {
        $left = substr($GLOBALS[$this->varname], 0, $this->position);
        $right = substr($GLOBALS[$this->varname], $this->position + strlen($data));
        $GLOBALS[$this->varname] = $left . $data . $right;
        $this->position += strlen($data);
        return strlen($data);
    }

    function stream_tell()
    {
        return $this->position;
    }

    function stream_eof()
    {
        return $this->position >= strlen($GLOBALS[$this->varname]);
    }

    function stream_seek($offset, $whence)
    {
        switch($whence) {
            case SEEK_SET:
                if ($offset < strlen($GLOBALS[$this->varname]) && $offset >= 0) {
                     $this->position = $offset;
                     return true;
                } else {
                     return false;
                }
                break;

            case SEEK_CUR:
                if ($offset >= 0) {
                     $this->position += $offset;
                     return true;
                } else {
                     return false;
                }
                break;

            case SEEK_END:
                if (strlen($GLOBALS[$this->varname]) + $offset >= 0) {
                     $this->position = strlen($GLOBALS[$this->varname]) + $offset;
                     return true;
                } else {
                     return false;
                }
                break;

            default:
                return false;
        }
    }
}

stream_register_wrapper("var", "VariableStream")
    or die("Failed to register protocol");

$myvar = "";

$fp = fopen("var://myvar", "r+");

fwrite($fp, "line1\n");
fwrite($fp, "line2\n");
fwrite($fp, "line3\n");

rewind($fp);
while(!feof($fp)) {
    echo fgets($fp);
}
fclose($fp);
var_dump($myvar);
```

stream_select

(PHP 4 >= 4.3.0)

stream_select -- Runs the equivalent of the select() system call on the given arrays of streams with a timeout specified by tv_sec and tv_usec

Description

int **stream_select** (resource &read, resource &write, resource &except, int tv_sec [, int tv_usec])

The **stream_select()** function accepts arrays of streams and waits for them to change status. Its opperation is equivalent to that of the **socket_select()** function except in that it acts on streams.

The streams listed in the *read* array will be watched to see if characters become available for reading (more precisely, to see if a read will not block - in particular, a stream resource is also ready on end-of-file, in which case an **fread()** will return a zero length string).

The streams listed in the *write* array will be watched to see if a write will not block.

The streams listed in the *except* array will be watched for exceptions.

Warning
On exit, the arrays are modified to indicate which stream resource actually changed status.

You do not need to pass every array to **stream_select()**. You can leave it out and use an empty array or **NULL** instead. Also do not forget that those arrays are passed *by reference* and will be modified after **stream_select()** returns.

Example:

```
/* Prepare the read array */
$read = array($stream1, $stream2);

if (false === ($num_changed_streams = stream_select($read, $write = NULL, $except = NULL, 0))) {
    /* Error handling */
else if ($num_changed_streams > 0) {
    /* At least on one of the streams something interesting happened */
}
```

> **Note:** Due a limitation in the current Zend Engine it is not possible to pass a constant modifier like **NULL** directly as a parameter to a function which expects this parameter to be passed by reference. Instead use a temporary variable or an expression with the leftmost member being a temporary variable:
>
> ```
> stream_select($r, $w, $e = NULL, 0);
> ```

The *tv_sec* and *tv_usec* together form the *timeout* parameter. The *timeout* is an upper bound on the amount of time elapsed before **stream_select()** returns. *tv_sec* may be zero , causing **stream_select()** to return immediately. This is useful for polling. If *tv_sec* is **NULL** (no timeout), **stream_select()** can block indefinitely.

On success **stream_select()** returns the number of stream resorces contained in the modified arrays, which may be zero if the timeout expires before anything interesting happens. On error **FALSE** is returned.

> **Note:** Be sure to use the === operator when checking for an error. Since the **stream_select()** may return o the comparison with == would evaluate to **TRUE**:
>
> ```
> if (false === stream_select($r, $w, $e = NULL, 0)) {
>     echo "stream_select() failed\n";
> }
> ```

> **Note:** Be aware that some stream implementations need to be handled very carefully. A few basic rules:
>
> - You should always try to use **stream_select()** without timeout. Your program should have nothing to do if there is no data available. Code that depends on timeouts is not usually portable and difficult to debug.
> - If you read/write to a stream returned in the arrays be aware that they do not necessarily read/write the full amount of data you have requested. Be prepared to even only be able to read/write a single byte.

See also **stream_set_blocking()**

stream_set_blocking

(PHP 4 >= 4.3.0)

stream_set_blocking -- Set blocking/non-blocking mode on a stream

Description

bool **stream_set_blocking** (resource stream, int mode)

If *mode* is **FALSE**, the given stream will be switched to non-blocking mode, and if **TRUE**, it will be switched to blocking mode. This affects calls like **fgets()** and **fread()** that read from the stream. In non-blocking mode an **fgets()** call will always return right away while in blocking mode it will wait for data to become available on the stream.

This function was previously called as **set_socket_blocking()** and later **socket_set_blocking()** but this usage is deprecated.

> **Note:** Prior to PHP 4.3, this function only worked on socket based streams. Since PHP 4.3, this function works for any stream that supports non-blocking mode (currently, regular files and socket streams).

stream_set_timeout

(PHP 4 >= 4.3.0)

stream_set_timeout -- Set timeout period on a stream

Description

bool **stream_set_timeout** (resource stream, int seconds, int microseconds)

Sets the timeout value on *stream*, expressed in the sum of *seconds* and *microseconds*.

Example 1. stream_set_timeout() Example

```
<?php
$fp = fsockopen("www.example.com", 80);
if(!$fp) {
    echo "Unable to open\n";
} else {
    fputs($fp, "GET / HTTP/1.0\n\n");
    $start = time();
    stream_set_timeout($fp, 2);
    $res = fread($fp, 2000);
    var_dump(stream_get_meta_data($fp));
    fclose($fp);
    print $res;
}
?>
```

> **Note:** As of PHP 4.3, this function can (potentially) work on any kind of stream. In PHP 4.3, socket based streams are still the only kind supported in the PHP core, although streams from other extensions may support this function.

This function was previously called as **set_socket_timeout()** and later **socket_set_timeout()** but this usage is deprecated.

See also: **fsockopen()** and **fopen()**.

stream_set_write_buffer

(PHP 4 >= 4.3.0)

stream_set_write_buffer -- Sets file buffering on the given stream

Description

int **stream_set_write_buffer** (resource stream, int buffer)

Output using **fwrite()** is normally buffered at 8K. This means that if there are two processes wanting to write to the same output stream (a file), each is paused after 8K of data to allow the other to write. **stream_set_write_buffer()** sets the buffering for write operations on the given filepointer *stream* to *buffer* bytes. If *buffer* is 0 then write operations are unbuffered. This ensures that all writes with **fwrite()** are completed before other processes are allowed to write to that output stream.

The function returns 0 on success, or EOF if the request cannot be honored.

The following example demonstrates how to use **stream_set_write_buffer()** to create an unbuffered stream.

Example 1. stream_set_write_buffer() example

```
$fp = fopen($file, "w");
if ($fp) {
  stream_set_write_buffer($fp, 0);
  fputs($fp, $output);
  fclose($fp);
```

```
}
```

See also **fopen()** and **fwrite()**.

XCIX. String functions

Introduction

These functions all manipulate strings in various ways. Some more specialized sections can be found in the regular expression and URL handling sections.

For information on how strings behave, especially with regard to usage of single quotes, double quotes, and escape sequences, see the Strings entry in the Types section of the manual.

Requirements

No external libraries are needed to build this extension.

Installation

There is no installation needed to use these functions; they are part of the PHP core.

Predefined Constants

This extension has no constants defined.

See Also

For even more powerful string handling and manipulating functions take a look at the POSIX regular expression functions and the Perl compatible regular expression functions.

Table of Contents

ltrim -- Strip whitespace from the beginning of a string
md5_file -- Calculates the md5 hash of a given filename
md5 -- Calculate the md5 hash of a string
metaphone -- Calculate the metaphone key of a string
money_format -- Formats a number as a currency string
nl_langinfo -- Query language and locale information
nl2br -- Inserts HTML line breaks before all newlines in a string
number_format -- Format a number with grouped thousands
ord -- Return ASCII value of character
parse_str -- Parses the string into variables
print -- Output a string
printf -- Output a formatted string
quoted_printable_decode -- Convert a quoted-printable string to an 8 bit string
quotemeta -- Quote meta characters
rtrim -- Strip whitespace from the end of a string
setlocale -- Set locale information
sha1_file -- Calculate the sha1 hash of a file
sha1 -- Calculate the sha1 hash of a string
similar_text -- Calculate the similarity between two strings
soundex -- Calculate the soundex key of a string
sprintf -- Return a formatted string
sscanf -- Parses input from a string according to a format
str_pad -- Pad a string to a certain length with another string
str_repeat -- Repeat a string
str_replace -- Replace all occurrences of the search string with the replacement string
str_rot13 -- Perform the rot13 transform on a string
str_shuffle -- Randomly shuffles a string
str_word_count -- Return information about words used in a string
strcasecmp -- Binary safe case-insensitive string comparison
strchr -- Find the first occurrence of a character
strcmp -- Binary safe string comparison
strcoll -- Locale based string comparison
strcspn -- Find length of initial segment not matching mask
strip_tags -- Strip HTML and PHP tags from a string
stripcslashes -- Un-quote string quoted with **addcslashes()**
stripslashes -- Un-quote string quoted with **addslashes()**
stristr -- Case-insensitive **strstr()**
strlen -- Get string length
strnatcasecmp -- Case insensitive string comparisons using a "natural order" algorithm
strnatcmp -- String comparisons using a "natural order" algorithm
strncasecmp -- Binary safe case-insensitive string comparison of the first n characters
strncmp -- Binary safe string comparison of the first n characters
strpos -- Find position of first occurrence of a string
strrchr -- Find the last occurrence of a character in a string
strrev -- Reverse a string
strrpos -- Find position of last occurrence of a char in a string
strspn -- Find length of initial segment matching mask
strstr -- Find first occurrence of a string
strtok -- Tokenize string
strtolower -- Make a string lowercase
strtoupper -- Make a string uppercase
strtr -- Translate certain characters
substr_count -- Count the number of substring occurrences
substr_replace -- Replace text within a portion of a string
substr -- Return part of a string
trim -- Strip whitespace from the beginning and end of a string
ucfirst -- Make a string's first character uppercase
ucwords -- Uppercase the first character of each word in a string
vprintf -- Output a formatted string
vsprintf -- Return a formatted string
wordwrap -- Wraps a string to a given number of characters using a string break character.

addcslashes

(PHP 4)

addcslashes -- Quote string with slashes in a C style

Description

string **addcslashes** (string str, string charlist)

Returns a string with backslashes before characters that are listed in *charlist* parameter. It escapes \n, \r etc. in C-like style, characters with ASCII code lower than 32 and higher than 126 are converted to octal representation.

Be careful if you choose to escape characters o, a, b, f, n, r, t and v. They will be converted to \o, \a, \b, \f, \n, \r, \t and \v. In PHP \o (**NULL**), \r (carriage return), \n (newline) and \t (tab) are predefined escape sequences, while in C all of these are predefined escape sequences.

charlist like "\o..\37", which would escape all characters with ASCII code between 0 and 31.

Example 1. addcslashes() example

```
$escaped = addcslashes($not_escaped, "\0..\37!@\177..\377");
```

When you define a sequence of characters in the charlist argument make sure that you know what characters come between the characters that you set as the start and end of the range.

```
echo addcslashes('foo[ ]', 'A..z');
// output:  \f\o\o\[ \]
// All upper and lower-case letters will be escaped
// ... but so will the [\]^_` and any tabs, line
// feeds, carriage returns, etc.
```

Also, if the first character in a range has a lower ASCII value than the second character in the range, no range will be constructed. Only the start, end and period characters will be escaped. Use the **ord()** function to find the ASCII value for a character.

```
echo addcslashes("zoo['.']", 'z..A');
// output:  \zoo['\.']
```

See also **stripcslashes()**, **stripslashes()**, **htmlspecialchars()**, and **quotemeta()**.

addslashes

(PHP 3, PHP 4)

addslashes -- Quote string with slashes

Description

string **addslashes** (string str)

Returns a string with backslashes before characters that need to be quoted in database queries etc. These characters are single quote ('), double quote ("), backslash (\) and NUL (the **NULL** byte).

> **Note:** magic_quotes_gpc is ON by default.

See also **stripslashes()**, **htmlspecialchars()**, and **quotemeta()**.

bin2hex

(PHP 3>= 3.0.9, PHP 4)

bin2hex -- Convert binary data into hexadecimal representation

Description

string **bin2hex** (string str)

Returns an ASCII string containing the hexadecimal representation of *str*. The conversion is done byte-wise with the high-nibble first.

See also **pack()** and **unpack()**.

chop

chop -- Alias of **rtrim()**

Description

This function is an alias of **rtrim()**.

> **Note: chop()** is different than the Perl `chop()` function, which removes the last character in the string.

chr

(PHP 3, PHP 4)

chr -- Return a specific character

Description

string **chr** (int ascii)

Returns a one-character string containing the character specified by *ascii*.

Example 1. chr() example

```
$str .= chr(27); /* add an escape character at the end of $str */
/* Often this is more useful */
$str = sprintf("The string ends in escape: %c", 27);
```

You can find an ASCII-table over here: http://www.asciitable.com.

This function complements **ord()**. See also **sprintf()** with a format string of `%c`.

chunk_split

(PHP 3>= 3.0.6, PHP 4)

chunk_split -- Split a string into smaller chunks

Description

string **chunk_split** (string body [, int chunklen [, string end]])

Can be used to split a string into smaller chunks which is useful for e.g. converting base64_encode output to match RFC 2045 semantics. It inserts *end* (defaults to "\r\n") every *chunklen* characters (defaults to 76). It returns the new string leaving the original string untouched.

Example 1. chunk_split() example

```
# format $data using RFC 2045 semantics
$new_string = chunk_split(base64_encode($data));
```

See also **explode()**, **split()** **wordwrap()** and RFC 2045.

convert_cyr_string

(PHP 3>= 3.0.6, PHP 4)

convert_cyr_string -- Convert from one Cyrillic character set to another

Description

string **convert_cyr_string** (string str, string from, string to)

This function returns the given string converted from one Cyrillic character set to another. The *from* and *to* arguments are single characters that represent the source and target Cyrillic character sets. The supported types are:

- k - koi8-r
- w - windows-1251
- i - iso8859-5
- a - x-cp866
- d - x-cp866
- m - x-mac-cyrillic

count_chars

(PHP 4)

count_chars -- Return information about characters used in a string

Description

mixed **count_chars** (string string [, int mode])

Counts the number of occurrences of every byte-value (0..255) in *string* and returns it in various ways. The optional parameter *Mode* default to 0. Depending on *mode* **count_chars()** returns one of the following:

- 0 - an array with the byte-value as key and the frequency of every byte as value.
- 1 - same as 0 but only byte-values with a frequency greater than zero are listed.
- 2 - same as 0 but only byte-values with a frequency equal to zero are listed.
- 3 - a string containing all used byte-values is returned.
- 4 - a string containing all not used byte-values is returned.

See also **strpos()** and **substr_count()**.

crc32

(PHP 4 >= 4.0.1)

crc32 -- Calculates the crc32 polynomial of a string

Description

int **crc32** (string str)

Generates the cyclic redundancy checksum polynomial of 32-bit lengths of the *str*. This is usually used to validate the integrity of data being transmitted.

> **Note:** Because PHP's integer type is signed, and many crc32 checksums will result in negative integers, you need to use the "%u" formatter of **sprintf()** or **printf()** to get the string representation of the unsigned crc32 checksum.

This second example shows how to print a converted checksum with the **printf()** function :

Example 1. Displaying a crc32 checksum

```
<?php
$checksum = crc32("The quick brown fox jumped over the lazy dog.");
printf("%u\n", $checksum);
?>
```

See also **md5()**

crypt

(PHP 3, PHP 4)

crypt -- One-way string encryption (hashing)

Description

string **crypt** (string str [, string salt])

crypt() will return an encrypted string using the standard Unix DES-based encryption algorithm or alternative algorithms that may be available on the system. Arguments are a string to be encrypted and an optional salt string to base the encryption on. See the Unix man page for your crypt function for more information.

If the salt argument is not provided, one will be randomly generated by PHP.

Some operating systems support more than one type of encryption. In fact, sometimes the standard DES-based encryption is replaced by an MD5-based encryption algorithm. The encryption type is triggered by the salt argument. At install time, PHP determines the capabilities of the crypt function and will accept salts for other encryption types. If no salt is provided, PHP will auto-generate a standard two character salt by default, unless the default encryption type on the system is MD5, in which case a random MD5-compatible salt is generated. PHP sets a constant named CRYPT_SALT_LENGTH which tells you whether a regular two character salt applies to your system or the longer twelve character salt is applicable.

If you are using the supplied salt, you should be aware that the salt is generated once. If you are calling this function recursively, this may impact both appearance and security.

The standard DES-based encryption **crypt()** returns the salt as the first two characters of the output. It also only uses the first eight characters of *str*, so longer strings that start with the same eight characters will generate the same result (when the same salt is used).

On systems where the crypt() function supports multiple encryption types, the following constants are set to 0 or 1 depending on whether the given type is available:

- CRYPT_STD_DES - Standard DES-based encryption with a two character salt
- CRYPT_EXT_DES - Extended DES-based encryption with a nine character salt
- CRYPT_MD5 - MD5 encryption with a twelve character salt starting with 1
- CRYPT_BLOWFISH - Blowfish encryption with a sixteen character salt starting with 2

  Note: There is no decrypt function, since **crypt()** uses a one-way algorithm.

Example 1. crypt() examples

```
<?php
$password = crypt("My1sTpassword"); # let salt be generated

# You should pass the entire results of crypt() as the salt for comparing a
# password, to avoid problems when different hashing algorithms are used. (As
# it says above, standard DES-based password hashing uses a 2-character salt,
# but MD5-based hashing uses 12.)
if (crypt($user_input,$password) == $password) {
   echo "Password verified!";
}
?>
```

See also **md5()** and the Mcrypt extension.

echo

(PHP 3, PHP 4)

echo -- Output one or more strings

Description

echo (string arg1 [, string argn...])

Outputs all parameters.

echo() is not actually a function (it is a language construct) so you are not required to use parentheses with it. In fact, if you want to pass more than one parameter to echo, you must not enclose the parameters within parentheses. It is not possible to use **echo()** in a variable function context.

Example 1. echo() examples

```
<?php
echo "Hello World";

echo "This spans
multiple lines. The newlines will be
output as well";

echo "This spans\nmultiple lines. The newlines will be\noutput as well.";

echo "Escaping characters is done \"Like this\".";

//You can use variables inside of an echo statement
$foo = "foobar";
$bar = "barbaz";

echo "foo is $foo"; // foo is foobar

// Using single quotes will print the variable name, not the value
echo 'foo is $foo'; // foo is $foo

// If you are not using any other characters, you can just echo variables
echo $foo;          // foobar
echo $foo,$bar;     // foobarbarbaz

echo <<<END
This uses the "here document" syntax to output
multiple lines with $variable interpolation. Note
that the here document terminator must appear on a
line with just a semicolon no extra whitespace!
END;

// Because echo is not a function, following code is invalid.
($some_var) ? echo('true'): echo('false');

// However, the following examples will work:
($some_var) ? print('true'): print('false'); // print is a function
echo $some_var ? 'true': 'false'; // changing the statement around
?>
```

echo() also has a shortcut syntax, where you can immediately follow the opening tag with an equals sign.

```
I have <?=$foo?> foo.
```

> **Note:** This short syntax only works with the short_open_tag configuration setting enabled.

See also **print()**, **printf()**, and **flush()**.

explode

(PHP 3, PHP 4)

explode -- Split a string by string

Description

array **explode** (string separator, string string [, int limit])

Returns an array of strings, each of which is a substring of *string* formed by splitting it on boundaries formed by the string *separator*. If *limit* is set, the returned array will contain a maximum of *limit* elements with the last element containing the rest of *string*.

If *separator* is an empty string (""), **explode()** will return **FALSE**. If *separator* contains a value that is not contained in *string*, then **explode()** will return an array containing *string*.

> **Note:** The *limit* parameter was added in PHP 4.0.1

Example 1. explode() examples

```
$pizza = "piece1 piece2 piece3 piece4 piece5 piece6";
$pieces = explode(" ", $pizza);

$data = "foo:*:1023:1000::/home/foo:/bin/sh";
list($user,$pass,$uid,$gid,$gecos,$home,$shell) = explode(":",$data);
```

> **Note:** Although **implode()** can for historical reasons accept its parameters in either order, **explode()** cannot. You must ensure that the *separator* argument comes before the *string* argument.

See also **preg_split()**, **spliti()**, **split()**, and **implode()**.

fprintf

(PHP 5 CVS only)

fprintf -- Write a formatted string to a stream

Description

int **fprintf** (resource handle, string format [, mixed args])

Write a string produced according to the formatting string *format* to the stream resource specified by *handle*..

The format string is composed of zero or more directives: ordinary characters (excluding %) that are copied directly to the result, and *conversion specifications*, each of which results in fetching its own parameter. This applies to **fprintf()**, **sprintf()**, and **printf()**.

Each conversion specification consists of a percent sign (%), followed by one or more of these elements, in order:

1. An optional *padding specifier* that says what character will be used for padding the results to the right string size. This may be a space character or a 0 (zero character). The default is to pad with spaces. An alternate padding character can be specified by prefixing it with a single quote ('). See the examples below.
2. An optional *alignment specifier* that says if the result should be left-justified or right-justified. The default is right-justified; a - character here will make it left-justified.
3. An optional number, a *width specifier* that says how many characters (minimum) this conversion should result in.
4. An optional *precision specifier* that says how many decimal digits should be displayed for floating-point numbers. This option has no effect for other types than **float**. (Another function useful for formatting numbers is **number_format()**.)
5. A *type specifier* that says what type the argument data should be treated as. Possible types:

   % - a literal percent character. No argument is required.
   b - the argument is treated as an integer, and presented as a binary number.
   c - the argument is treated as an integer, and presented as the character with that ASCII value.
   d - the argument is treated as an integer, and presented as a (signed) decimal number.
   u - the argument is treated as an integer, and presented as an unsigned decimal number.
   f - the argument is treated as a **float**, and presented as a floating-point number.
   o - the argument is treated as an integer, and presented as an octal number.
   s - the argument is treated as and presented as a string.
   x - the argument is treated as an integer and presented as a hexadecimal number (with lowercase letters).
   X - the argument is treated as an integer and presented as a hexadecimal number (with uppercase letters).

See also: **printf()**, **sprintf()**, **sscanf()**, **fscanf()**, **vsprintf()**, and **number_format()**.

Examples

Example 1. sprintf(): zero-padded integers

```
$isodate = sprintf("%04d-%02d-%02d", $year, $month, $day);
```

Example 2. sprintf(): formatting currency

```
$money1 = 68.75;
$money2 = 54.35;
```

```
$money = $money1 + $money2;
// echo $money will output "123.1";
$formatted = sprintf("%01.2f", $money);
// echo $formatted will output "123.10"
```

get_html_translation_table

(PHP 4)

get_html_translation_table -- Returns the translation table used by **htmlspecialchars()** and **htmlentities()**

Description

string **get_html_translation_table** (int table [, int quote_style])

get_html_translation_table() will return the translation table that is used internally for **htmlspecialchars()** and **htmlentities()**. There are two new defines (*HTML_ENTITIES*, *HTML_SPECIALCHARS*) that allow you to specify the table you want. And as in the **htmlspecialchars()** and **htmlentities()** functions you can optionally specify the quote_style you are working with. The default is ENT_COMPAT mode. See the description of these modes in **htmlspecialchars()**.

Example 1. Translation Table Example

```
$trans = get_html_translation_table(HTML_ENTITIES);
$str = "Hallo & <Frau> & Krämer";
$encoded = strtr($str, $trans);
```

The `$encoded` variable will now contain: `"Hallo & <Frau> & Kr&auml;mer"`.

The cool thing is using **array_flip()** to change the direction of the translation.

```
$trans = array_flip($trans);
$original = strtr($encoded, $trans);
```

The content of `$original` would be: "Hallo & <Frau> & Krämer".

See also **htmlspecialchars()**, **htmlentities()**, **strtr()**, and **array_flip()**.

hebrev

(PHP 3, PHP 4)

hebrev -- Convert logical Hebrew text to visual text

Description

string **hebrev** (string hebrew_text [, int max_chars_per_line])

The optional parameter *max_chars_per_line* indicates maximum number of characters per line will be output. The function tries to avoid breaking words.

See also **hebrevc()**

hebrevc

(PHP 3, PHP 4)

hebrevc -- Convert logical Hebrew text to visual text with newline conversion

Description

string **hebrevc** (string hebrew_text [, int max_chars_per_line])

This function is similar to **hebrev()** with the difference that it converts newlines (\n) to "
\n". The optional parameter *max_chars_per_line* indicates maximum number of characters per line will be output. The function tries to avoid breaking words.

See also **hebrev()**

html_entity_decode

(PHP 4 >= 4.3.0)

html_entity_decode -- Convert all HTML entities to their applicable characters

Description

string **html_entity_decode** (string string [, int quote_style [, string charset]])

html_entity_decode() is the opposite of **htmlentities()** in that it converts all HTML entities to their applicable characters from *string*.

The optional second *quote_style* parameter lets you define what will be done with 'single' and "double" quotes. It takes on one of three constants with the default being **ENT_COMPAT**:

Table 1. Available *quote_style* constants

Constant Name	Description
ENT_COMPAT	Will convert double-quotes and leave single-quotes alone.
ENT_QUOTES	Will convert both double and single quotes.
ENT_NOQUOTES	Will leave both double and single quotes unconverted.

The ISO-8859-1 character set is used as default for the optional third *charset*. This defines the character set used in conversion.

Example 1. Decoding html entities

```
<?php
$orig = "I'll \"walk\" the <b>dog</b> now";

$a = htmlentities($orig);

$b = html_entity_decode($a);

echo $a; // I'll &quot;walk&quot; the &lt;b&gt;dog&lt;/b&gt; now

echo $b; // I'll "walk" the <b>dog</b> now


// For users prior to PHP 4.3.0 you may do this:
function unhtmlentities ($string)
{
    $trans_tbl = get_html_translation_table (HTML_ENTITIES);
    $trans_tbl = array_flip ($trans_tbl);
    return strtr ($string, $trans_tbl);
}

$c = unhtmlentities($a);

echo $c; // I'll "walk" the <b>dog</b> now

?>
```

> **Note:** You might wonder why trim(html_entity_decode(' ')); doesn't reduce the string to an empty string, that's because the ' ' entity is not ASCII code 32 (which is stripped by **trim()**) but ASCII code 160 (oxao) in the default ISO 8859-1 characterset.

See also **htmlentities()**, **htmlspecialchars()**, **get_html_translation_table()**, **htmlspecialchars()** and **urldecode()**.

htmlentities

(PHP 3, PHP 4)

htmlentities -- Convert all applicable characters to HTML entities

Description

string **htmlentities** (string string [, int quote_style [, string charset]])

This function is identical to **htmlspecialchars()** in all ways, except with **htmlentities()**, all characters which have HTML character entity equivalents are translated into these entities.

Like **htmlspecialchars()**, the optional second *quote_style* parameter lets you define what will be done with 'single' and "double" quotes. It takes on one of three constants with the default being **ENT_COMPAT**:

Table 1. Available *quote_style* constants

Constant Name	Description
ENT_COMPAT	Will convert double-quotes and leave single-quotes alone.
ENT_QUOTES	Will convert both double and single quotes.
ENT_NOQUOTES	Will leave both double and single quotes unconverted.

Support for the optional *quote* parameter was added in PHP 4.0.3.

Like **htmlspecialchars()**, it takes an optional third argument which defines character set used in conversion. Support for this argument was added in PHP 4.1.0. Presently, the ISO-8859-1 character set is used as the default.

If you're wanting to decode instead (the reverse) you can use **html_entity_decode()**.

See also **html_entity_decode()**, **get_html_translation_table()**, **htmlspecialchars()**, **nl2br()**, and **urlencode()**.

htmlspecialchars

(PHP 3, PHP 4)

htmlspecialchars -- Convert special characters to HTML entities

Description

string **htmlspecialchars** (string string [, int quote_style [, string charset]])

Certain characters have special significance in HTML, and should be represented by HTML entities if they are to preserve their meanings. This function returns a string with some of these conversions made; the translations made are those most useful for everyday web programming. If you require all HTML character entities to be translated, use **htmlentities()** instead.

This function is useful in preventing user-supplied text from containing HTML markup, such as in a message board or guest book application. The optional second argument, quote_style, tells the function what to do with single and double quote characters. The default mode, ENT_COMPAT, is the backwards compatible mode which only translates the double-quote character and leaves the single-quote untranslated. If ENT_QUOTES is set, both single and double quotes are translated and if ENT_NOQUOTES is set neither single nor double quotes are translated.

The translations performed are:

- '&' (ampersand) becomes '&'
- '"' (double quote) becomes '"' when ENT_NOQUOTES is not set.
- ''' (single quote) becomes ''' only when ENT_QUOTES is set.
- '<' (less than) becomes '<'
- '>' (greater than) becomes '>'

Example 1. htmlspecialchars() example

```
$new = htmlspecialchars("<a href='test'>Test</a>", ENT_QUOTES);
```

Note that this function does not translate anything beyond what is listed above. For full entity translation, see **htmlentities()**. Support for the optional second argument was added in PHP 3.0.17 and PHP 4.0.3.

The third argument defines character set used in conversion. The default character set is ISO-8859-1. Support for this third argument was added in PHP 4.1.0.

See also **get_html_translation_table()**, **htmlentities()** and **nl2br()**.

implode

(PHP 3, PHP 4)

implode -- Join array elements with a string

Description

string **implode** (string glue, array pieces)

Returns a string containing a string representation of all the array elements in the same order, with the glue string between each element.

Example 1. implode() example

```
<?php

$array = array('lastname', 'email', 'phone');
$comma_separated = implode(",", $array);

print $comma_separated; // lastname,email,phone

?>
```

> **Note: implode()** can, for historical reasons, accept its parameters in either order. For consistency with **explode()**, however, it may be less confusing to use the documented order of arguments.

See also **explode()**, and **split()**.

join

(PHP 3, PHP 4)

join -- Join array elements with a string

Description

string **join** (string glue, array pieces)

join() is an alias to **implode()**, and is identical in every way.

See also **explode()**, **implode()**, and **split()**.

levenshtein

(PHP 3>= 3.0.17, PHP 4 >= 4.0.1)

levenshtein -- Calculate Levenshtein distance between two strings

Description

int **levenshtein** (string str1, string str2)

int **levenshtein** (string str1, string str2, int cost_ins, int cost_rep, int cost_del)

int **levenshtein** (string str1, string str2, function cost)

This function returns the Levenshtein-Distance between the two argument strings or -1, if one of the argument strings is longer than the limit of 255 characters (255 should be more than enough for name or dictionary comparison, and nobody serious would be doing genetic analysis with PHP).

The Levenshtein distance is defined as the minimal number of characters you have to replace, insert or delete to transform *str1* into *str2*. The complexity of the algorithm is `O(m*n)`, where `n` and `m` are the length of *str1* and *str2* (rather good when compared to **similar_text()**, which is O(max(n,m)**3), but still expensive).

In its simplest form the function will take only the two strings as parameter and will calculate just the number of insert, replace and delete operations needed to transform *str1* into *str2*.

A second variant will take three additional parameters that define the cost of insert, replace and delete operations. This is more general and adaptive than variant one, but not as efficient.

The third variant (which is not implemented yet) will be the most general and adaptive, but also the slowest alternative. It will call a user-supplied function that will determine the cost for every possible operation.

The user-supplied function will be called with the following arguments:

- operation to apply: 'I', 'R' or 'D'
- actual character in string 1
- actual character in string 2
- position in string 1
- position in string 2
- remaining characters in string 1
- remaining characters in string 2

The user-supplied function has to return a positive integer describing the cost for this particular operation, but it may decide to use only some of the supplied arguments.

The user-supplied function approach offers the possibility to take into account the relevance of and/or difference between certain symbols (characters) or even the context those symbols appear in to determine the cost of insert, replace and delete operations, but at the cost of losing all optimizations done regarding cpu register utilization and cache misses that have been worked into the other two variants.

See also **soundex()**, **similar_text()**, and **metaphone()**.

localeconv

(PHP 4 >= 4.0.5)

localeconv -- Get numeric formatting information

Description

array **localeconv** (void)

Returns an associative array containing localized numeric and monetary formatting information.

localeconv() returns data based upon the current locale as set by **setlocale()**. The associative array that is returned contains the following fields:

Array element	Description
decimal_point	Decimal point character
thousands_sep	Thousands separator
grouping	Array containing numeric groupings
int_curr_symbol	International currency symbol (i.e. USD)
currency_symbol	Local currency symbol (i.e. $)
mon_decimal_point	Monetary decimal point character
mon_thousands_sep	Monetary thousands separator
mon_grouping	Array containing monetary groupings
positive_sign	Sign for positive values
negative_sign	Sign for negative values
int_frac_digits	International fractional digits
frac_digits	Local fractional digits
p_cs_precedes	**TRUE** if currency_symbol precedes a positive value, **FALSE** if it succeeds one

Array element	Description
p_sep_by_space	**TRUE** if a space separates currency_symbol from a positive value, **FALSE** otherwise
n_cs_precedes	**TRUE** if currency_symbol precedes a negative value, **FALSE** if it succeeds one
n_sep_by_space	**TRUE** if a space separates currency_symbol from a negative value, **FALSE** otherwise
p_sign_posn	0 Parentheses surround the quantity and currency_symbol 1 The sign string precedes the quantity and currency_symbol 2 The sign string succeeds the quantity and currency_symbol 3 The sign string immediately precedes the currency_symbol 4 The sign string immediately succeeds the currency_symbol
n_sign_posn	0 Parentheses surround the quantity and currency_symbol 1 The sign string precedes the quantity and currency_symbol 2 The sign string succeeds the quantity and currency_symbol 3 The sign string immediately precedes the currency_symbol 4 The sign string immediately succeeds the currency_symbol

The grouping fields contain arrays that define the way numbers should be grouped. For example, the grouping field for the en_US locale, would contain a 2 item array with the values 3 and 3. The higher the index in the array, the farther left the grouping is. If an array element is equal to CHAR_MAX, no further grouping is done. If an array element is equal to 0, the previous element should be used.

Example 1. localeconv() example

```
setlocale(LC_ALL, "en_US");

$locale_info = localeconv();

echo "<PRE>\n";
echo "-------------------------------------------\n";
echo "  Monetary information for current locale:  \n";
echo "-------------------------------------------\n\n";

echo "int_curr_symbol:   {$locale_info["int_curr_symbol"]}\n";
echo "currency_symbol:   {$locale_info["currency_symbol"]}\n";
echo "mon_decimal_point: {$locale_info["mon_decimal_point"]}\n";
echo "mon_thousands_sep: {$locale_info["mon_thousands_sep"]}\n";
echo "positive_sign:     {$locale_info["positive_sign"]}\n";
echo "negative_sign:     {$locale_info["negative_sign"]}\n";
echo "int_frac_digits:   {$locale_info["int_frac_digits"]}\n";
echo "frac_digits:       {$locale_info["frac_digits"]}\n";
echo "p_cs_precedes:     {$locale_info["p_cs_precedes"]}\n";
echo "p_sep_by_space:    {$locale_info["p_sep_by_space"]}\n";
echo "n_cs_precedes:     {$locale_info["n_cs_precedes"]}\n";
echo "n_sep_by_space:    {$locale_info["n_sep_by_space"]}\n";
echo "p_sign_posn:       {$locale_info["p_sign_posn"]}\n";
echo "n_sign_posn:       {$locale_info["n_sign_posn"]}\n";
echo "</PRE>\n";
```

The constant CHAR_MAX is also defined for the use mentioned above.

See also **setlocale()**.

ltrim

(PHP 3, PHP 4)

ltrim -- Strip whitespace from the beginning of a string

Description

string **ltrim** (string str [, string charlist])

> **Note:** The second parameter was added in PHP 4.1.0

This function returns a string with whitespace stripped from the beginning of *str*. Without the second parameter, **ltrim()** will strip these characters:

- " " (ASCII 32 (0x20)), an ordinary space.
- "\t" (ASCII 9 (0x09)), a tab.

- "\n" (ASCII 10 (0x0A)), a new line (line feed).
- "\r" (ASCII 13 (0x0D)), a carriage return.
- "\0" (ASCII 0 (0x00)), the NUL-byte.
- "\x0B" (ASCII 11 (0x0B)), a vertical tab.

You can also specify the characters you want to strip, by means of the *charlist* parameter. Simply list all characters that you want to be stripped. With .. you can specify a range of characters.

Example 1. Usage example of ltrim()

```
<?php

$text = "\t\tThese are a few words :) ...  ";
$trimmed = ltrim($text);
// $trimmed = "These are a few words :) ...  "
$trimmed = ltrim($text," \t.");
// $trimmed = "These are a few words :) ...  "
$clean = ltrim($binary,"\0x00..\0x1F");
// trim the ASCII control characters at the beginning of $binary
// (from 0 to 31 inclusive)

?>
```

See also **trim()** and **rtrim()**.

md5_file

(PHP 4 >= 4.2.0)

md5_file -- Calculates the md5 hash of a given filename

Description

string **md5_file** (string filename)

Calculates the MD5 hash of the specified *filename* using the RSA Data Security, Inc. MD5 Message-Digest Algorithm, and returns that hash.

This function has the same purpose of the command line utility md5sum.

See also **md5()** and **crc32()**

md5

(PHP 3, PHP 4)

md5 -- Calculate the md5 hash of a string

Description

string **md5** (string str)

Calculates the MD5 hash of *str* using the RSA Data Security, Inc. MD5 Message-Digest Algorithm, and returns that hash. The hash is a 32-character hexadecimal number.

See also **crc32()**, **md5_file()**, and **sha1()**.

metaphone

(PHP 4)

metaphone -- Calculate the metaphone key of a string

Description

string **metaphone** (string str)

Calculates the metaphone key of *str*.

Similar to **soundex()** metaphone creates the same key for similar sounding words. It's more accurate than **soundex()** as it knows the basic rules of English pronunciation. The metaphone generated keys are of variable length.

Metaphone was developed by Lawrence Philips <lphilips@verity.com>. It is described in ["Practical Algorithms for Programmers", Binstock & Rex, Addison Wesley, 1995].

money_format

(PHP 4 >= 4.3.0)

money_format -- Formats a number as a currency string

Description

string **money_format** (string format, float number)

money_format() returns a formatted version of *number*. This function wraps the C library function **strfmon()**, with the difference that this implementation converts only one number at a time.

The format specification consists of the following sequence:

- a % character
- optional flags
- optional field width
- optional left precision
- optional right precision
- a required conversion character

Flags. One or more of the optional flags below can be used:

=f

The character = followed by a a (single byte) character *f* to be used as the numeric fill character. The default fill character is space.

^

Disable the use of grouping characters (as defined by the current locale).

+ or (

Specify the formatting style for positive and negative numbers. If + is used, the locale's equivalent for + and - will be used. If (is used, negative amounts are enclosed in parenthesis. If no specification is given, the default is +.

!

Suppress the currency symbol from the output string.

-

If present, it will make all fields left-justified (padded to the right), as opposed to the default which is for the fields to be right-justified (padded to the left).

Field width.

w

A decimal digit string specifying a minimum field width. Field will be right-justified unless the flag - is used. Default value is 0 (zero).

Left precision.

#n

The maximum number of digits (*n*) expected to the left of the decimal character (e.g. the decimal point). It is used usually to keep formatted output aligned in the same columns, using the fill character if the number of digits is less than *n*. If the number of actual digits is bigger than *n*, then this specification is ignored.

If grouping has not been suppressed using the ^ flag, grouping separators will be inserted before the fill characters (if any) are added. Grouping separators will not be applied to fill characters, even if the fill character is a digit.

To ensure alignment, any characters appearing before or after the number in the formatted output such as currency or sign symbols are padded as necessary with space characters to make their positive and negative formats an equal length.

Right precision .

.p

A period followed by the number of digits (*p*) after the decimal character. If the value of *p* is 0 (zero), the decimal character and the digits to its right will be omitted. If no right precision is included, the default will dictated by the current local in use. The amount being formatted is rounded to the specified number of digits prior to formatting.

Conversion characters .

`i`

The number is formatted according to the locale's international currency format (e.g. for the USA locale: USD 1,234.56).

`n`

The number is formatted according to the locale's national currency format (e.g. for the de_DE locale: DM1.234,56).

`%`

Returns the the `%` character.

Note: The **LC_MONETARY** category of the locale settings, affects the behavior of this function. Use **setlocale()** to set to the appropriate default locale before using this function.

Characters before and after the formatting string will be returned unchanged.

Example 1. money_format() Example

We will use different locales and format specifications to illustrate the use of this function.

```
<?php

    $number = 1234.56;

    // let's print the international format for the en_US locale
    setlocale(LC_MONETARY, 'en_US');
    echo money_format('%i', $number)."\n";
    // USD 1,234.56

    // Italian national format with 2 decimals`
    setlocale(LC_MONETARY, 'it_IT');
    echo money_format('%.2n', $number)."\n";
    // L. 1.234,56

    // Using a negative number
    $number = -1234.5672;

    // US national format, using () for negative numbers
    // and 10 digits for left precision
    setlocale(LC_MONETARY, 'en_US');
    echo money_format('%(#10n', $number)."\n";
    // ($        1,234.57)

    // Similar format as above, adding the use of 2 digits of right
    // precision and '*' as a fill character
    echo money_format('%=*(#10.2n', $number)."\n";
    // ($********1,234.57)

    // Let's justify to the left, with 14 positions of width, 8 digits of
    // left precision, 2 of right precision, withouth grouping character
    // and using the international format for the de_DE locale.
    setlocale(LC_MONETARY, 'de_DE');
    echo money_format('%=*^-14#8.2i', 1234.56)."\n";
    // DEM 1234,56****

    // Let's add some blurb before and after the conversion specification
    setlocale(LC_MONETARY, 'en_GB');
```

```
    $fmt = 'The final value is %i (after a 10%% discount)';
    echo money_format($fmt, 1234.56)."\n";
    // The final value is  GBP 1,234.56 (after a 10% discount)

?>
```

See also: **setlocale()**, **number_format()**,**sprintf()**, **printf()** and **sscanf()**.

nl_langinfo

(PHP 4 >= 4.1.0)

nl_langinfo -- Query language and locale information

Description

string **nl_langinfo** (int item)

Warning
This function is currently not documented; only the argument list is available.

nl2br

(PHP 3, PHP 4)

nl2br -- Inserts HTML line breaks before all newlines in a string

Description

string **nl2br** (string string)

Returns *string* with '<br />' inserted before all newlines.

> **Note:** Starting with PHP 4.0.5, **nl2br()** is now XHTML compliant. All versions before 4.0.5 will return *string* with '<br>' inserted before newlines instead of '<br />'.

See also **htmlspecialchars()**, **htmlentities()**, **wordwrap()**, and **str_replace()**.

number_format

(PHP 3, PHP 4)

number_format -- Format a number with grouped thousands

Description

string **number_format** (float number [, int decimals [, string dec_point [, string thousands_sep]]])

number_format() returns a formatted version of *number*. This function accepts either one, two or four parameters (not three):

If only one parameter is given, *number* will be formatted without decimals, but with a comma (",") between every group of thousands.

If two parameters are given, *number* will be formatted with *decimals* decimals with a dot (".") in front, and a comma (",") between every group of thousands.

If all four parameters are given, *number* will be formatted with *decimals* decimals, *dec_point* instead of a dot (".") before the decimals and *thousands_sep* instead of a comma (",") between every group of thousands.

> **Note:** Only the first character of *thousands_sep* is used. For example, if you use `foo` as *thousands_sep* on the number `1000`, **number_format()** will return `1f000`.

Example 1. number_format() Example

For instance, French notation usually use two decimals, comma (',') as decimal separator, and space (' ') as thousand separator. This is achieved with this line :

```
<?php

    $number = 1234.56;

    // english notation (default)
    $english_format_number = number_format($number);
    // 1,234

    // French notation
    $nombre_format_francais = number_format($number, 2, ',', ' ');
    // 1 234,56

    $number = 1234.5678;

    // english notation without thousands seperator
    $english_format_number = number_format($number, 2, '.', '');
    // 1234.57

?>
```

See also: **sprintf()**, **printf()** and **sscanf()**.

ord

(PHP 3, PHP 4)

ord -- Return ASCII value of character

Description

int **ord** (string string)

Returns the ASCII value of the first character of *string*. This function complements **chr()**.

Example 1. ord() example

```
if (ord($str) == 10) {
    echo "The first character of \$str is a line feed.\n";
}
```

You can find an ASCII-table over here: http://www.asciitable.com.

See also **chr()**.

parse_str

(PHP 3, PHP 4)

parse_str -- Parses the string into variables

Description

void **parse_str** (string str [, array arr])

Parses *str* as if it were the query string passed via an URL and sets variables in the current scope. If the second parameter *arr* is present, variables are stored in this variable as an array elements instead.

> **Note:** Support for the optional second parameter was added in PHP 4.0.3.
>
> **Note:** To get the current *QUERY_STRING*, you may use the variable $_SERVER['QUERY_STRING']. Also, you may want to read the section on variables from outside of PHP.

Example 1. Using parse_str()

```
<?php
$str = "first=value&arr[]=foo+bar&arr[]=baz";
parse_str($str);
echo $first;  // value
```

```
echo $arr[0]; // foo bar
echo $arr[1]; // baz

parse_str($str, $output);
echo $output['first'];  // value
echo $output['arr'][0]; // foo bar
echo $output['arr'][1]; // baz

?>
```

See also **parse_url()**, **pathinfo()**, **set_magic_quotes_runtime()**, and **urldecode()**.

print

(PHP 3, PHP 4)

print -- Output a string

Description

print (string arg)

Outputs *arg*. Returns **TRUE** on success or **FALSE** on failure.

print() is not actually a real function (it is a language construct) so you are not required to use parentheses with it.

Example 1. print() examples

```
<?php
print("Hello World");

print "print() also works without parentheses.";

print "This spans
multiple lines. The newlines will be
output as well";

print "This spans\nmultiple lines. The newlines will be\noutput as well.";

print "escaping characters is done \"Like this\".";

// You can use variables inside of an print statement
$foo = "foobar";
$bar = "barbaz";

print "foo is $foo"; // foo is foobar

// Using single quotes will print the variable name, not the value
print 'foo is $foo'; // foo is $foo

// If you are not using any other characters, you can just print variables
print $foo;          // foobar

print <<<END
This uses the "here document" syntax to output
multiple lines with $variable interpolation. Note
that the here document terminator must appear on a
line with just a semicolon no extra whitespace!
END;
?>
```

See also **echo()**, **printf()**, and **flush()**.

printf

(PHP 3, PHP 4)

printf -- Output a formatted string

Description

void **printf** (string format [, mixed args])

Produces output according to *format*, which is described in the documentation for **sprintf()**.

See also **print()**, **sprintf()**, **sscanf()**, **fscanf()**, and **flush()**.

quoted_printable_decode

(PHP 3>= 3.0.6, PHP 4)

quoted_printable_decode -- Convert a quoted-printable string to an 8 bit string

Description

string **quoted_printable_decode** (string str)

This function returns an 8-bit binary string corresponding to the decoded quoted printable string. This function is similar to **imap_qprint()**, except this one does not require the IMAP module to work.

quotemeta

(PHP 3, PHP 4)

quotemeta -- Quote meta characters

Description

string **quotemeta** (string str)

Returns a version of str with a backslash character (\) before every character that is among these:
`. \\ + * ? [ ^ ] ( $ )`

See also **addslashes()**, **htmlentities()**, **htmlspecialchars()**, **nl2br()**, and **stripslashes()**.

rtrim

(PHP 3, PHP 4)

rtrim -- Strip whitespace from the end of a string

Description

string **rtrim** (string str [, string charlist])

> **Note:** The second parameter was added in PHP 4.1.0

This function returns a string with whitespace stripped from the end of *str*. Without the second parameter, **rtrim()** will strip these characters:

- " " (ASCII 32 (0x20)), an ordinary space.
- "\t" (ASCII 9 (0x09)), a tab.
- "\n" (ASCII 10 (0x0A)), a new line (line feed).
- "\r" (ASCII 13 (0x0D)), a carriage return.
- "\0" (ASCII 0 (0x00)), the NUL-byte.
- "\xoB" (ASCII 11 (0x0B)), a vertical tab.

You can also specify the characters you want to strip, by means of the *charlist* parameter. Simply list all characters that you want to be stripped. With .. you can specify a range of characters.

Example 1. Usage example of rtrim()

```
<?php
```

```
$text = "\t\tThese are a few words :) ...  ";
$trimmed = rtrim($text);
// $trimmed = "\t\tThese are a few words :) ..."
$trimmed = rtrim($text," \t.");
// $trimmed = "\t\tThese are a few words :)"
$clean = rtrim($binary,"\0x00..\0x1F");
// trim the ASCII control characters at the end of $binary
// (from 0 to 31 inclusive)

?>
```

See also **trim()** and **ltrim()**.

setlocale

(PHP 3, PHP 4)

setlocale -- Set locale information

Description

string **setlocale** (mixed category, string locale [, string ...])

string **setlocale** (mixed category, array locale)

Category is a named constant (or string) specifying the category of the functions affected by the locale setting:

- LC_ALL for all of the below
- LC_COLLATE for string comparison, see **strcoll()**
- LC_CTYPE for character classification and conversion, for example **strtoupper()**
- LC_MONETARY for **localeconv()**
- LC_NUMERIC for decimal separator (See also **localeconv()**)
- LC_TIME for date and time formatting with **strftime()**

If *locale* is the empty string "", the locale names will be set from the values of environment variables with the same names as the above categories, or from "LANG".

If *locale* is zero or "0", the locale setting is not affected, only the current setting is returned.

If *locale* is an array or followed by additional parameters then each array element or parameter is tried to be set as new locale until success. This is usefull if a locale is known under different names on different systems or for providing a fallback for a possibly not available locale.

> **Note:** Passing multiple locales is not available before PHP 4.3.0

Setlocale returns the new current locale, or **FALSE** if the locale functionality is not implemented in the platform, the specified locale does not exist or the category name is invalid. An invalid category name also causes a warning message.

> **Note:** The return value of **setlocale()** depends on the system that PHP is running. It returns exactly what the system setlocale function returns.

Example 1. setlocale() Examples

```
<?php
/* Set locale to Dutch */
setlocale (LC_ALL, 'nl_NL');

/* Output: vrijdag 22 december 1978 */
echo strftime ("%A %e %B %Y", mktime (0, 0, 0, 12, 22, 1978));

/* try different possible locale names for german as of PHP 4.3.0 */
$loc_de = setlocale (LC_ALL, 'de_DE@euro', 'de_DE', 'de', 'ge');
echo "Preferred locale for german on this system is '$loc_de'";
?>
```

sha1_file

(PHP 4 >= 4.3.0)

sha1_file -- Calculate the sha1 hash of a file

Description

string **sha1_file** (string filename)

Calculates the sha1 hash of *filename* using the US Secure Hash Algorithm 1, and returns that hash. The hash is a 40-character hexadecimal number. Upon failure, **FALSE** is returned.

See also **sha1()**, **crc32()**, and **md5_file()**

sha1

(PHP 4 >= 4.3.0)

sha1 -- Calculate the sha1 hash of a string

Description

string **sha1** (string str)

Calculates the sha1 hash of *str* using the US Secure Hash Algorithm 1, and returns that hash. The hash is a 40-character hexadecimal number.

See also **sha1_file()**, **crc32()**, and **md5()**

similar_text

(PHP 3>= 3.0.7, PHP 4)

similar_text -- Calculate the similarity between two strings

Description

int **similar_text** (string first, string second [, float percent])

This calculates the similarity between two strings as described in Oliver [1993]. Note that this implementation does not use a stack as in Oliver's pseudo code, but recursive calls which may or may not speed up the whole process. Note also that the complexity of this algorithm is O(N**3) where N is the length of the longest string.

By passing a reference as third argument, **similar_text()** will calculate the similarity in percent for you. It returns the number of matching chars in both strings.

soundex

(PHP 3, PHP 4)

soundex -- Calculate the soundex key of a string

Description

string **soundex** (string str)

Calculates the soundex key of *str*.

Soundex keys have the property that words pronounced similarly produce the same soundex key, and can thus be used to simplify searches in databases where you know the pronunciation but not the spelling. This soundex function returns a string 4 characters long, starting with a letter.

This particular soundex function is one described by Donald Knuth in "The Art Of Computer Programming, vol. 3: Sorting And

Searching", Addison-Wesley (1973), pp. 391-392.

Example 1. Soundex Examples

```
soundex("Euler") == soundex("Ellery") == 'E460';
soundex("Gauss") == soundex("Ghosh") == 'G200';
soundex("Hilbert") == soundex("Heilbronn") == 'H416';
soundex("Knuth") == soundex("Kant") == 'K530';
soundex("Lloyd") == soundex("Ladd") == 'L300';
soundex("Lukasiewicz") == soundex("Lissajous") == 'L222';
```

See also **levenshtein()**, **metaphone()**, and **similar_text()**.

sprintf

(PHP 3, PHP 4)

sprintf -- Return a formatted string

Description

string **sprintf** (string format [, mixed args])

Returns a string produced according to the formatting string *format*.

The format string is composed of zero or more directives: ordinary characters (excluding %) that are copied directly to the result, and *conversion specifications*, each of which results in fetching its own parameter. This applies to both **sprintf()** and **printf()**.

Each conversion specification consists of a percent sign (%), followed by one or more of these elements, in order:

1. An optional *padding specifier* that says what character will be used for padding the results to the right string size. This may be a space character or a 0 (zero character). The default is to pad with spaces. An alternate padding character can be specified by prefixing it with a single quote ('). See the examples below.
2. An optional *alignment specifier* that says if the result should be left-justified or right-justified. The default is right-justified; a - character here will make it left-justified.
3. An optional number, a *width specifier* that says how many characters (minimum) this conversion should result in.
4. An optional *precision specifier* that says how many decimal digits should be displayed for floating-point numbers. This option has no effect for other types than **float**. (Another function useful for formatting numbers is **number_format()**.)
5. A *type specifier* that says what type the argument data should be treated as. Possible types:

   % - a literal percent character. No argument is required.
   b - the argument is treated as an integer, and presented as a binary number.
   c - the argument is treated as an integer, and presented as the character with that ASCII value.
   d - the argument is treated as an integer, and presented as a (signed) decimal number.
   u - the argument is treated as an integer, and presented as an unsigned decimal number.
   f - the argument is treated as a **float**, and presented as a floating-point number.
   o - the argument is treated as an integer, and presented as an octal number.
   s - the argument is treated as and presented as a string.
   x - the argument is treated as an integer and presented as a hexadecimal number (with lowercase letters).
   X - the argument is treated as an integer and presented as a hexadecimal number (with uppercase letters).

As of PHP version 4.0.6 the format string supports argument numbering/swapping. Here is an example:

Example 1. Argument swapping

```
$format = "There are %d monkeys in the %s";
printf($format,$num,$location);
```

This might output, "There are 5 monkeys in the tree". But imagine we are creating a format string in a separate file, commonly because we would like to internationalize it and we rewrite it as:

Example 2. Argument swapping

```
$format = "The %s contains %d monkeys";
```

```
printf($format,$num,$location);
```

We now have a problem. The order of the placeholders in the format string does not match the order of the arguments in the code. We would like to leave the code as is and simply indicate in the format string which arguments the placeholders refer to. We would write the format string like this instead:

Example 3. Argument swapping

```
$format = "The %2\$s contains %1\$d monkeys";
printf($format,$num,$location);
```

An added benefit here is that you can repeat the placeholders without adding more arguments in the code. For example:

Example 4. Argument swapping

```
$format = "The %2\$s contains %1\$d monkeys.
           That's a nice %2\$s full of %1\$d monkeys.";
printf($format, $num, $location);
```

See also **printf()**, **sscanf()**, **fscanf()**, **vsprintf()**, and **number_format()**.

Examples

Example 5. sprintf(): zero-padded integers

```
$isodate = sprintf("%04d-%02d-%02d", $year, $month, $day);
```

Example 6. sprintf(): formatting currency

```
$money1 = 68.75;
$money2 = 54.35;
$money = $money1 + $money2;
// echo $money will output "123.1";
$formatted = sprintf("%01.2f", $money);
// echo $formatted will output "123.10"
```

sscanf

(PHP 4 >= 4.0.1)

sscanf -- Parses input from a string according to a format

Description

mixed **sscanf** (string str, string format [, string var1])

The function **sscanf()** is the input analog of **printf()**. **sscanf()** reads from the string *str* and interprets it according to the specified *format*. If only two parameters were passed to this function, the values parsed will be returned as an array.

Any whitespace in the format string matches any whitespace in the input string. This means that even a tab \t in the format string can match a single space character in the input string.

Example 1. sscanf() Example

```
// getting the serial number
$serial = sscanf("SN/2350001","SN/%d");
// and the date of manufacturing
$mandate = "January 01 2000";
list($month, $day, $year) = sscanf($mandate,"%s %d %d");
echo "Item $serial was manufactured on: $year-".substr($month,0,3)."-$day\n";
```

If optional parameters are passed, the function will return the number of assigned values. The optional parameters must be passed by reference.

Example 2. sscanf() - using optional parameters

```
// get author info and generate DocBook entry
$auth = "24\tLewis Carroll";
$n = sscanf($auth,"%d\t%s %s", &$id, &$first, &$last);
echo "<author id='$id'>
    <firstname>$first</firstname>
    <surname>$last</surname>
```

```
</author>\n";
```

See also **fscanf()**, **printf()**, and **sprintf()**.

str_pad

(PHP 4 >= 4.0.1)

str_pad -- Pad a string to a certain length with another string

Description

string **str_pad** (string input, int pad_length [, string pad_string [, int pad_type]])

This functions returns the *input* string padded on the left, the right, or both sides to the specified padding length. If the optional argument *pad_string* is not supplied, the *input* is padded with spaces, otherwise it is padded with characters from *pad_string* up to the limit.

Optional argument *pad_type* can be STR_PAD_RIGHT, STR_PAD_LEFT, or STR_PAD_BOTH. If *pad_type* is not specified it is assumed to be STR_PAD_RIGHT.

If the value of *pad_length* is negative or less than the length of the input string, no padding takes place.

Example 1. str_pad() example

```
$input = "Alien";
print str_pad($input, 10);                      // produces "Alien     "
print str_pad($input, 10, "-=", STR_PAD_LEFT);  // produces "-=-=-Alien"
print str_pad($input, 10, "_", STR_PAD_BOTH);   // produces "__Alien___"
```

str_repeat

(PHP 4)

str_repeat -- Repeat a string

Description

string **str_repeat** (string input, int multiplier)

Returns *input_str* repeated *multiplier* times. *multiplier* has to be greater than or equal to 0. If the *multiplier* is set to 0, the function will return an empty string.

Example 1. str_repeat() example

```
echo str_repeat("-=", 10);
```

This will output "-=-=-=-=-=-=-=-=-=-=".

See also for, **str_pad()**, and **substr_count()**.

str_replace

(PHP 3>= 3.0.6, PHP 4)

str_replace -- Replace all occurrences of the search string with the replacement string

Description

mixed **str_replace** (mixed search, mixed replace, mixed subject)

This function returns a string or an array with all occurences of *search* in *subject* replaced with the given *replace* value. If you

don't need fancy replacing rules, you should always use this function instead of **ereg_replace()** or **preg_replace()**.

In PHP 4.0.5 and later, every parameter to **str_replace()** can be an array.

If *subject* is an array, then the search and replace is performed with every entry of *subject*, and the return value is an array as well.

If *search* and *replace* are arrays, then **str_replace()** takes a value from each array and uses them to do search and replace on *subject*. If *replace* has fewer values than *search*, then an empty string is used for the rest of replacement values. If *search* is an array and *replace* is a string; then this replacement string is used for every value of *search*.

Example 1. str_replace() example

```
$bodytag = str_replace("%body%", "black", "<body text=%body%>");
```

This function is binary safe.

> **Note: str_replace()** was added in PHP 3.0.6, but was buggy up until PHP 3.0.8.

See also **ereg_replace()**, **preg_replace()**, and **strtr()**.

str_rot13

(PHP 4 >= 4.2.0)

str_rot13 -- Perform the rot13 transform on a string

Description

string **str_rot13** (string str)

This function performs the ROT13 encoding on the *str* argument and returns the resulting string. The ROT13 encoding simply shifts every letter by 13 places in the alphabet while leaving non-alpha characters untouched. Encoding and decoding are done by the same function, passing an encoded string as argument will return the original version.

str_shuffle

(PHP 4 >= 4.3.0)

str_shuffle -- Randomly shuffles a string

Description

string **str_shuffle** (string str)

str_shuffle() shuffles a string. One permutation of all possible is created.

Example 1. str_shuffle() example

```
<?php
$str = 'abcdef';
$shuffled = str_shuffle($str);

// This will print something like: bfdaec
print $shuffled;
?>
```

See also **shuffle()** and **rand()**.

str_word_count

(PHP 4 >= 4.3.0)

str_word_count -- Return information about words used in a string

Description

mixed **str_word_count** (string string [, int format])

Counts the number of words inside *string*. If the optional *format* is not specified, then the return value will be an integer representing the number of words found. In the event the *format* is specified, the return value will be an array, content of which is dependent on the *format*. The possible value for the *format* and the resultant outputs are listed below.

- 1 - returns an array containing all the words found inside the *string*.
- 2 - returns an associative array, where the key is the numeric position of the word inside the *string* and the value is the actual word itself.

For the purpose of this function, 'word' is defined as a locale dependent string containing alphabetic characters, which also may contain, but not start with "'" and "-" characters.

strcasecmp

(PHP 3>= 3.0.2, PHP 4)

strcasecmp -- Binary safe case-insensitive string comparison

Description

int **strcasecmp** (string str1, string str2)

Returns < 0 if *str1* is less than *str2*; > 0 if *str1* is greater than *str2*, and 0 if they are equal.

Example 1. strcasecmp() example

```
$var1 = "Hello";
$var2 = "hello";
if (!strcasecmp($var1, $var2)) {
    echo '$var1 is equal to $var2 in a case-insensitive string comparison';
}
```

See also **ereg()**, **strcmp()**, **substr()**, **stristr()**, **strncasecmp()**, and **strstr()**.

strchr

(PHP 3, PHP 4)

strchr -- Find the first occurrence of a character

Description

string **strchr** (string haystack, string needle)

This function is an alias for **strstr()**, and is identical in every way.

strcmp

(PHP 3, PHP 4)

strcmp -- Binary safe string comparison

Description

int **strcmp** (string str1, string str2)

Returns < 0 if *str1* is less than *str2*; > 0 if *str1* is greater than *str2*, and 0 if they are equal.

Note that this comparison is case sensitive.

See also **ereg()**, **strcasecmp()**, **substr()**, **stristr()**, **strncasecmp()**, **strncmp()**, and **strstr()**.

strcoll

(PHP 4 >= 4.0.5)

strcoll -- Locale based string comparison

Description

int **strcoll** (string str1, string str2)

Returns < 0 if *str1* is less than *str2*; > 0 if *str1* is greater than *str2*, and 0 if they are equal. **strcoll()** uses the current locale for doing the comparisons. If the current locale is C or POSIX, this function is equivalent to **strcmp()**.

Note that this comparison is case sensitive, and unlike **strcmp()** this function is not binary safe.

> **Note: strcoll()** was added in PHP 4.0.5, but was not enabled for win32 until 4.2.3.

See also **ereg()**, **strcmp()**, **strcasecmp()**, **substr()**, **stristr()**, **strncasecmp()**, **strncmp()**, **strstr()**, and **setlocale()**.

strcspn

(PHP 3>= 3.0.3, PHP 4)

strcspn -- Find length of initial segment not matching mask

Description

int **strcspn** (string str1, string str2)

Returns the length of the initial segment of *str1* which does *not* contain any of the characters in *str2*.

See also **strspn()**.

strip_tags

(PHP 3>= 3.0.8, PHP 4)

strip_tags -- Strip HTML and PHP tags from a string

Description

string **strip_tags** (string str [, string allowable_tags])

This function tries to return a string with all HTML and PHP tags stripped from a given *str*. It errors on the side of caution in case of incomplete or bogus tags. It uses the same tag stripping state machine as the **fgetss()** function.

You can use the optional second parameter to specify tags which should not be stripped.

> **Note:** *allowable_tags* was added in PHP 3.0.13 and PHP 4.0b3.

Example 1. strip_tags() example

```
$string = strip_tags($string, '<a><b><i><u>');
```

Warning
This function does not modify any attributes on the tags that you allow using *allowable_tags*, including the `style` and `onmouseover` attributes that a mischievous user may abuse when posting text that will be shown to other users.

stripcslashes

(PHP 4)

stripcslashes -- Un-quote string quoted with **addcslashes()**

Description

string **stripcslashes** (string str)

Returns a string with backslashes stripped off. Recognizes C-like `\n`, `\r` ..., octal and hexadecimal representation.

See also **addcslashes()**.

stripslashes

(PHP 3, PHP 4)

stripslashes -- Un-quote string quoted with **addslashes()**

Description

string **stripslashes** (string str)

Returns a string with backslashes stripped off. (`\'` becomes `'` and so on.) Double backslashes are made into a single backslash.

See also **addslashes()**.

stristr

(PHP 3>= 3.0.6, PHP 4)

stristr -- Case-insensitive **strstr()**

Description

string **stristr** (string haystack, string needle)

Returns all of *haystack* from the first occurrence of *needle* to the end. *needle* and *haystack* are examined in a case-insensitive manner.

If *needle* is not found, returns **FALSE**.

If *needle* is not a string, it is converted to an integer and applied as the ordinal value of a character.

See also **strchr()**, **strrchr()**, **substr()**, and **ereg()**.

strlen

(PHP 3, PHP 4)

strlen -- Get string length

Description

int **strlen** (string str)

Returns the length of *string*.

strnatcasecmp

(PHP 4)

strnatcasecmp -- Case insensitive string comparisons using a "natural order" algorithm

Description

int **strnatcasecmp** (string str1, string str2)

This function implements a comparison algorithm that orders alphanumeric strings in the way a human being would. The behaviour of this function is similar to **strnatcmp()**, except that the comparison is not case sensitive. For more infomation see: Martin Pool's Natural Order String Comparison page.

Similar to other string comparison functions, this one returns < 0 if *str1* is less than *str2*; > 0 if *str1* is greater than *str2*, and 0 if they are equal.

See also **ereg()**, **strcasecmp()**, **substr()**, **stristr()**, **strcmp()**, **strncmp()**, **strncasecmp()**, **strnatcmp()**, and **strstr()**.

strnatcmp

(PHP 4)

strnatcmp -- String comparisons using a "natural order" algorithm

Description

int **strnatcmp** (string str1, string str2)

This function implements a comparison algorithm that orders alphanumeric strings in the way a human being would, this is described as a "natural ordering". An example of the difference between this algorithm and the regular computer string sorting algorithms (used in **strcmp()**) can be seen below:

```
$arr1 = $arr2 = array("img12.png","img10.png","img2.png","img1.png");
echo "Standard string comparison\n";
usort($arr1,"strcmp");
print_r($arr1);
echo "\nNatural order string comparison\n";
usort($arr2,"strnatcmp");
print_r($arr2);
```

The code above will generate the following output:

```
Standard string comparison
Array
(
    [0] => img1.png
    [1] => img10.png
    [2] => img12.png
    [3] => img2.png
)

Natural order string comparison
Array
(
    [0] => img1.png
    [1] => img2.png
    [2] => img10.png
    [3] => img12.png
)
```

For more information see: Martin Pool's Natural Order String Comparison page.

Similar to other string comparison functions, this one returns < 0 if *str1* is less than *str2*; > 0 if *str1* is greater than *str2*, and 0 if they are equal.

Note that this comparison is case sensitive.

See also **ereg()**, **strcasecmp()**, **substr()**, **stristr()**, **strcmp()**, **strncmp()**, **strncasecmp()**, **strnatcasecmp()**, **strstr()**, **natsort()** and **natcasesort()**.

strncasecmp

(PHP 4 >= 4.0.2)

strncasecmp -- Binary safe case-insensitive string comparison of the first n characters

Description

int **strncasecmp** (string str1, string str2, int len)

This function is similar to **strcasecmp()**, with the difference that you can specify the (upper limit of the) number of characters (*len*) from each string to be used in the comparison. If any of the strings is shorter than *len*, then the length of that string will be used for the comparison.

Returns < 0 if *str1* is less than *str2*; > 0 if *str1* is greater than *str2*, and 0 if they are equal.

See also **ereg()**, **strcasecmp()**, **strcmp()**, **substr()**, **stristr()**, and **strstr()**.

strncmp

(PHP 4)

strncmp -- Binary safe string comparison of the first n characters

Description

int **strncmp** (string str1, string str2, int len)

This function is similar to **strcmp()**, with the difference that you can specify the (upper limit of the) number of characters (*len*) from each string to be used in the comparison. If any of the strings is shorter than *len*, then the length of that string will be used for the comparison.

Returns < 0 if *str1* is less than *str2*; > 0 if *str1* is greater than *str2*, and 0 if they are equal.

Note that this comparison is case sensitive.

See also **ereg()**, **strncasecmp()**, **strcasecmp()**, **substr()**, **stristr()**, **strcmp()**, and **strstr()**.

strpos

(PHP 3, PHP 4)

strpos -- Find position of first occurrence of a string

Description

int **strpos** (string haystack, string needle [, int offset])

Returns the numeric position of the first occurrence of *needle* in the *haystack* string. Unlike the **strrpos()**, this function can take a full string as the *needle* parameter and the entire string will be used.

If *needle* is not found, returns **FALSE**.

> **Note:** It is easy to mistake the return values for "character found at position 0" and "character not found". Here's how to detect the difference:

```
// in PHP 4.0b3 and newer:
$pos = strpos($mystring, "b");
if ($pos === false) { // note: three equal signs
    // not found...
}

// in versions older than 4.0b3:
$pos = strpos($mystring, "b");
if (!is_integer($pos)) {
    // not found...
}
```

If *needle* is not a string, it is converted to an integer and applied as the ordinal value of a character.

The optional *offset* parameter allows you to specify which character in *haystack* to start searching. The position returned is still relative to the the beginning of *haystack*.

See also **strrpos()**, **strrchr()**, **substr()**, **stristr()**, and **strstr()**.

strrchr

(PHP 3, PHP 4)

strrchr -- Find the last occurrence of a character in a string

Description

string **strrchr** (string haystack, string needle)

This function returns the portion of *haystack* which starts at the last occurrence of *needle* and goes until the end of *haystack*.

Returns **FALSE** if *needle* is not found.

If *needle* contains more than one character, the first is used.

If *needle* is not a string, it is converted to an integer and applied as the ordinal value of a character.

Example 1. strrchr() example

```
// get last directory in $PATH
$dir = substr(strrchr($PATH, ":"), 1);

// get everything after last newline
$text = "Line 1\nLine 2\nLine 3";
$last = substr(strrchr($text, 10), 1 );
```

See also **strchr()**, **substr()**, **stristr()**, and **strstr()**.

strrev

(PHP 3, PHP 4)

strrev -- Reverse a string

Description

string **strrev** (string string)

Returns *string*, reversed.

Example 1. Reversing a string with strrev()

```
<php
echo strrev("Hello world!"); // outputs "!dlrow olleH"
?>
```

strrpos

(PHP 3, PHP 4)

strrpos -- Find position of last occurrence of a char in a string

Description

int **strrpos** (string haystack, char needle)

Returns the numeric position of the last occurrence of *needle* in the *haystack* string. Note that the needle in this case can only be a single character. If a string is passed as the needle, then only the first character of that string will be used.

If *needle* is not found, returns **FALSE**.

> **Note:** It is easy to mistake the return values for "character found at position 0" and "character not found". Here's how to detect the difference:

```
// in PHP 4.0b3 and newer:
$pos = strrpos($mystring, "b");
if ($pos === false) { // note: three equal signs
    // not found...
}

// in versions older than 4.0b3:
$pos = strrpos($mystring, "b");
if (is_string($pos) && !$pos) {
    // not found...
}
```

If *needle* is not a string, it is converted to an integer and applied as the ordinal value of a character.

See also **strpos()**, **strrchr()**, **substr()**, **stristr()**, and **strstr()**.

strspn

(PHP 3>= 3.0.3, PHP 4)

strspn -- Find length of initial segment matching mask

Description

int **strspn** (string str1, string str2)

Returns the length of the initial segment of *str1* which consists entirely of characters in *str2*.

The line of code:

```
$var = strspn("42 is the answer, what is the question ...", "1234567890");
```

will assign 2 to `$var`, because the string "42" will be the longest segment containing characters from "1234567890".

See also **strcspn()**.

strstr

(PHP 3, PHP 4)

strstr -- Find first occurrence of a string

Description

string **strstr** (string haystack, string needle)

Returns part of *haystack* string from the first occurrence of *needle* to the end of *haystack*.

If *needle* is not found, returns **FALSE**.

If *needle* is not a string, it is converted to an integer and applied as the ordinal value of a character.

> **Note:** This function is case-sensitive. For case-insensitive searches, use **stristr()**.

Example 1. strstr() example

```
$email = 'user@example.com';
$domain = strstr($email, '@');
print $domain; // prints @example.com
```

> **Note:** If you only want to determine if a particular *needle* occurs within *haystack*, use the faster and less memory intensive function **strpos()** instead.

See also **ereg()**, **preg_match()**, **strchr()**, **stristr()**, **strpos()**, **strrchr()**, and **substr()**.

strtok

(PHP 3, PHP 4)

strtok -- Tokenize string

Description

string **strtok** (string arg1, string arg2)

strtok() splits a string (*arg1*) into smaller strings (tokens), with each token being delimited by any character from *arg2*. That is, if you have a string like "This is an example string" you could tokenize this string into its individual words by using the space character as the token.

Example 1. strtok() example

```
$string = "This is\tan example\nstring";
/* Use tab and newline as tokenizing characters as well  */
$tok = strtok($string," \n\t");
while ($tok) {
    echo "Word=$tok<br>";
    $tok = strtok(" \n\t");
}
```

Note that only the first call to strtok uses the string argument. Every subsequent call to strtok only needs the token to use, as it keeps track of where it is in the current string. To start over, or to tokenize a new string you simply call strtok with the string argument again to initialize it. Note that you may put multiple tokens in the token parameter. The string will be tokenized when any one of the characters in the argument are found.

The behavior when an empty part was found changed with PHP 4.1.0. The old behavior returned an empty string, while the new, correct, behavior simply skips the part of the string:

Example 2. Old strtok() behavior

```
$first_token  = strtok('/something', '/');
    $second_token = strtok('/');
    var_dump ($first_token, $second_token);

/* Output:
    string(0) ""
    string(9) "something"
*/
```

Example 3. New strtok() behavior

```
$first_token  = strtok('/something', '/');
    $second_token = strtok('/');
    var_dump ($first_token, $second_token);

/* Output:
    string(9) "something"
    bool(false)
*/
```

Also be careful that your tokens may be equal to "0". This evaluates to FALSE in conditional expressions.

See also **split()** and **explode()**.

strtolower

(PHP 3, PHP 4)

strtolower -- Make a string lowercase

Description

string **strtolower** (string str)

Returns *string* with all alphabetic characters converted to lowercase.

Note that 'alphabetic' is determined by the current locale. This means that in i.e. the default "C" locale, characters such as umlaut-A (Ä) will not be converted.

Example 1. strtolower() example

```
$str = "Mary Had A Little Lamb and She LOVED It So";
$str = strtolower($str);
print $str; # Prints mary had a little lamb and she loved it so
```

See also **strtoupper()**, **ucfirst()**, and **ucwords()**.

strtoupper

(PHP 3, PHP 4)

strtoupper -- Make a string uppercase

Description

string **strtoupper** (string string)

Returns *string* with all alphabetic characters converted to uppercase.

Note that 'alphabetic' is determined by the current locale. For instance, in the default "C" locale characters such as umlaut-a (ä) will not be converted.

Example 1. strtoupper() example

```
$str = "Mary Had A Little Lamb and She LOVED It So";
$str = strtoupper($str);
print $str; # Prints MARY HAD A LITTLE LAMB AND SHE LOVED IT SO
```

See also **strtolower()**, **ucfirst()**, and **ucwords()**.

strtr

(PHP 3, PHP 4)

strtr -- Translate certain characters

Description

string **strtr** (string str, string from, string to)

string **strtr** (string str, array replace_pairs)

This function returns a copy of *str*, translating all occurrences of each character in *from* to the corresponding character in *to* and returning the result.

If *from* and *to* are different lengths, the extra characters in the longer of the two are ignored.

Example 1. strtr() example

```
$addr = strtr($addr, "äåö", "aao");
```

strtr() can be called with only two arguments. If called with two arguments it behaves in a new way: *from* then has to be an array that contains string -> string pairs that will be replaced in the source string. **strtr()** will always look for the longest possible match first and will *NOT* try to replace stuff that it has already worked on.

Examples:

```
$trans = array("hello" => "hi", "hi" => "hello");
echo strtr("hi all, I said hello", $trans) . "\n";
```

This will show: "hello all, I said hi",

> **Note:** This optional *to* and *from* parameters were added in PHP 4.0.0

See also **ereg_replace()**.

substr_count

(PHP 4)

substr_count -- Count the number of substring occurrences

Description

int **substr_count** (string haystack, string needle)

substr_count() returns the number of times the *needle* substring occurs in the *haystack* string.

Example 1. substr_count() example

```
<?php
print substr_count("This is a test", "is"); // prints out 2
?>
```

See also **count_chars()**, **strpos()**, **substr()**, and **strstr()**.

substr_replace

(PHP 4)

substr_replace -- Replace text within a portion of a string

Description

string **substr_replace** (string string, string replacement, int start [, int length])

substr_replace() replaces a copy of *string* delimited by the *start* and (optionally) *length* parameters with the string given in *replacement*. The result is returned.

If *start* is positive, the replacing will begin at the *start*'th offset into *string*.

If *start* is negative, the replacing will begin at the *start*'th character from the end of *string*.

If *length* is given and is positive, it represents the length of the portion of *string* which is to be replaced. If it is negative, it represents the number of characters from the end of *string* at which to stop replacing. If it is not given, then it will default to strlen(*string*); i.e. end the replacing at the end of *string*.

Example 1. substr_replace() example

```
<?php
$var = 'ABCDEFGH:/MNRPQR/';
echo "Original: $var<hr>\n";

/* These two examples replace all of $var with 'bob'. */
echo substr_replace($var, 'bob', 0) . "<br>\n";
echo substr_replace($var, 'bob', 0, strlen($var)) . "<br>\n";

/* Insert 'bob' right at the beginning of $var. */
echo substr_replace($var, 'bob', 0, 0) . "<br>\n";

/* These next two replace 'MNRPQR' in $var with 'bob'. */
echo substr_replace($var, 'bob', 10, -1) . "<br>\n";
echo substr_replace($var, 'bob', -7, -1) . "<br>\n";

/* Delete 'MNRPQR' from $var. */
echo substr_replace($var, '', 10, -1) . "<br>\n";
?>
```

See also **str_replace()** and **substr()**.

substr

(PHP 3, PHP 4)

substr -- Return part of a string

Description

string **substr** (string string, int start [, int length])

substr() returns the portion of *string* specified by the *start* and *length* parameters.

If *start* is non-negative, the returned string will start at the *start*'th position in *string*, counting from zero. For instance, in the string 'abcdef', the character at position 0 is 'a', the character at position 2 is 'c', and so forth.

Example 1. Basic substr() usage

```
<?php
$rest = substr("abcdef", 1);    // returns "bcdef"
$rest = substr("abcdef", 1, 3); // returns "bcd"
$rest = substr("abcdef", 0, 4); // returns "abcd"
$rest = substr("abcdef", 0, 8); // returns "abcdef"

// Accessing via curly braces is another option
$string = 'abcdef';
echo $string{0};               // returns a
echo $string{3};               // returns d
?>
```

If *start* is negative, the returned string will start at the *start*'th character from the end of *string*.

Example 2. Using a negative *start*

```
<?php
$rest = substr("abcdef", -1);    // returns "f"
$rest = substr("abcdef", -2);    // returns "ef"
$rest = substr("abcdef", -3, 1); // returns "d"
?>
```

If *length* is given and is positive, the string returned will contain at most *length* characters beginning from *start* (depending on the length of *string*). If *string* is less than *start* characters long, **FALSE** will be returned.

If *length* is given and is negative, then that many characters will be omitted from the end of *string* (after the start position has been calculated when a *start* is negative). If *start* denotes a position beyond this truncation, an empty string will be returned.

Example 3. Using a negative *length*

```
<?php
$rest = substr("abcdef", 0, -1);  // returns "abcde"
$rest = substr("abcdef", 2, -1);  // returns "cde"
$rest = substr("abcdef", 4, -4);  // returns ""
$rest = substr("abcdef", -3, -1); // returns "de"
?>
```

See also **strrchr()** and **ereg()**.

trim

(PHP 3, PHP 4)

trim -- Strip whitespace from the beginning and end of a string

Description

string **trim** (string str [, string charlist])

> **Note:** The optional *charlist* parameter was added in PHP 4.1.0

This function returns a string with whitespace stripped from the beginning and end of *str*. Without the second parameter, **trim()** will strip these characters:

- " " (ASCII 32 (0x20)), an ordinary space.
- "\t" (ASCII 9 (0x09)), a tab.

- "\n" (ASCII 10 (0x0A)), a new line (line feed).
- "\r" (ASCII 13 (0x0D)), a carriage return.
- "\0" (ASCII 0 (0x00)), the NUL-byte.
- "\x0B" (ASCII 11 (0x0B)), a vertical tab.

You can also specify the characters you want to strip, by means of the *charlist* parameter. Simply list all characters that you want to be stripped. With .. you can specify a range of characters.

Example 1. Usage example of trim()

```
<?php

$text = "\t\tThese are a few words :) ...  ";
$trimmed = trim($text);
// $trimmed = "These are a few words :) ..."
$trimmed = trim($text," \t.");
// $trimmed = "These are a few words :)"
$clean = trim($binary,"\0x00..\0x1F");
// trim the ASCII control characters at the beginning and end of $binary
// (from 0 to 31 inclusive)

?>
```

See also ltrim() and rtrim().

ucfirst

(PHP 3, PHP 4)

ucfirst -- Make a string's first character uppercase

Description

string **ucfirst** (string str)

Returns a string with the first character of *str* capitalized, if that character is alphabetic.

Note that 'alphabetic' is determined by the current locale. For instance, in the default "C" locale characters such as umlaut-a (ä) will not be converted.

Example 1. ucfirst() example

```
$foo = 'hello world!';
$foo = ucfirst($foo);             // Hello world!

$bar = 'HELLO WORLD!';
$bar = ucfirst($bar);             // HELLO WORLD!
$bar = ucfirst(strtolower($bar)); // Hello world!
```

See also strtolower(), strtoupper(), and ucwords().

ucwords

(PHP 3>= 3.0.3, PHP 4)

ucwords -- Uppercase the first character of each word in a string

Description

string **ucwords** (string str)

Returns a string with the first character of each word in *str* capitalized, if that character is alphabetic.

Example 1. ucwords() example

```
$foo = 'hello world!';
```

```
$foo = ucwords($foo);             // Hello World!

$bar = 'HELLO WORLD!';
$bar = ucwords($bar);             // HELLO WORLD!
$bar = ucwords(strtolower($bar)); // Hello World!
```

> **Note:** The definition of a word is any string of characters that is immediately after a whitespace (These are: space, form-feed, newline, carriage return, horizontal tab, and vertical tab).

See also **strtoupper()**, **strtolower()** and **ucfirst()**.

vprintf

(PHP 4 >= 4.1.0)

vprintf -- Output a formatted string

Description

void **vprintf** (string format, array args)

Display array values as a formatted string according to *format* (which is described in the documentation for **sprintf()**).

Operates as **printf()** but accepts an array of arguments, rather than a variable number of arguments.

See also **printf()**, **sprintf()**, **vsprintf()**

vsprintf

(PHP 4 >= 4.1.0)

vsprintf -- Return a formatted string

Description

string **vsprintf** (string format, array args)

Return array values as a formatted string according to *format* (which is described in the documentation for **sprintf()**).

Operates as **sprintf()** but accepts an array of arguments, rather than a variable number of arguments.

See also **sprintf()** and **vprintf()**

wordwrap

(PHP 4 >= 4.0.2)

wordwrap -- Wraps a string to a given number of characters using a string break character.

Description

string **wordwrap** (string str [, int width [, string break [, int cut]]])

Returns a string with *str* wrapped at the column number specified by the (optional) *width* parameter. The line is broken using the (optional) *break* parameter.

wordwrap() will automatically wrap at column 75 and break using '\n' (newline) if *width* or *break* are not given.

If the *cut* is set to 1, the string is always wrapped at the specified width. So if you have a word that is larger than the given width, it is broken apart. (See second example).

> **Note:** The optional *cut* parameter was added in PHP 4.0.3

Example 1. wordwrap() example

```
$text = "The quick brown fox jumped over the lazy dog.";
$newtext = wordwrap( $text, 20 );

echo "$newtext\n";
```

This example would display:

```
The quick brown fox

jumped over the

lazy dog.
```

Example 2. wordwrap() example

```
$text = "A very long woooooooooooord.";
$newtext = wordwrap( $text, 8, "\n", 1);

echo "$newtext\n";
```

This example would display:

```
A very

long

wooooooo

ooooord.
```

See also **nl2br()**.

C. Sybase functions

Introduction

Requirements

Installation

To enable Sybase-DB support configure PHP `--with-sybase[=DIR]`. DIR is the Sybase home directory, defaults to `/home/sybase`. To enable Sybase-CT support configure PHP `--with-sybase-ct[=DIR]`. DIR is the Sybase home directory, defaults to `/home/sybase`.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. Sybase configuration options

Name	Default	Changeable
sybase.allow_persistent	"On"	PHP_INI_SYSTEM
sybase.max_persistent	"-1"	PHP_INI_SYSTEM
sybase.max_links	"-1"	PHP_INI_SYSTEM
sybase.interface_file	"/usr/sybase/interfaces"	PHP_INI_SYSTEM
sybase.min_error_severity	"10"	PHP_INI_ALL
sybase.min_message_severity	"10"	PHP_INI_ALL
sybase.compatability_mode	"Off"	PHP_INI_SYSTEM
magic_quotes_sybase	"Off"	PHP_INI_ALL

Here is a short explanation of the configuration directives.

sybase.allow_persistent **boolean**

Whether to allow persistent Sybase connections.

sybase.max_persistent **integer**

The maximum number of persistent Sybase connections per process. -1 means no limit.

sybase.max_links **integer**

The maximum number of Sybase connections per process, including persistent connections. -1 means no limit.

sybase.min_error_severity **integer**

Minimum error severity to display.

sybase.min_message_severity **integer**

Minimum message severity to display.

sybase.compatability_mode **boolean**

Compatability mode with old versions of PHP 3.0. If on, this will cause PHP to automatically assign types to results according to their Sybase type, instead of treating them all as strings. This compatability mode will probably not stay around forever, so try applying whatever necessary changes to your code, and turn it off.

magic_quotes_sybase **boolean**

If *magic_quotes_sybase* is on, a single-quote is escaped with a single-quote instead of a backslash if magic_quotes_gpc or magic_quotes_runtime are enabled.

Note: Note that when *magic_quotes_sybase* is ON it completely overrides *magic_quotes_gpc* . In this case even when *magic_quotes_gpc* is enabled neither double quotes, backslashes or NUL's will be escaped.

Table 2. Sybase-CT configuration options

Name	Default	Changeable
sybct.allow_persistent	"On"	PHP_INI_SYSTEM
sybct.max_persistent	"-1"	PHP_INI_SYSTEM
sybct.max_links	"-1"	PHP_INI_SYSTEM
sybct.min_server_severity	"10"	PHP_INI_ALL
sybct.min_client_severity	"10"	PHP_INI_ALL
sybct.hostname	NULL	PHP_INI_ALL

Here is a short explanation of the configuration directives.

sybct.allow_persistent **boolean**

Whether to allow persistent Sybase-CT connections. The default is on.

sybct.max_persistent **integer**

The maximum number of persistent Sybase-CT connections per process. The default is -1 meaning unlimited.

sybct.max_links **integer**

The maximum number of Sybase-CT connections per process, including persistent connections. The default is -1 meaning unlimited.

sybct.min_server_severity **integer**

Server messages with severity greater than or equal to sybct.min_server_severity will be reported as warnings. This value can also be set from a script by calling **sybase_min_server_severity()**. The default is 10 which reports errors of information severity or greater.

sybct.min_client_severity **integer**

Client library messages with severity greater than or equal to sybct.min_client_severity will be reported as warnings. This value can also be set from a script by calling **sybase_min_client_severity()**. The default is 10 which effectively disables reporting.

sybct.hostname **string**

The name of the host you claim to be connecting from, for display by sp_who. The default is none.

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Resource Types

Predefined Constants

This extension has no constants defined.

Table of Contents

sybase_affected_rows

(PHP 3>= 3.0.6, PHP 4)

sybase_affected_rows -- get number of affected rows in last query

Description

int **sybase_affected_rows** ([int link_identifier])

Returns: The number of affected rows by the last query.

sybase_affected_rows() returns the number of rows affected by the last INSERT, UPDATE or DELETE query on the server associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

This command is not effective for SELECT statements, only on statements which modify records. To retrieve the number of rows returned from a SELECT, use **sybase_num_rows()**.

Note: This function is only available using the CT library interface to Sybase, and not the DB library.

sybase_close

(PHP 3, PHP 4)

sybase_close -- close Sybase connection

Description

bool **sybase_close** (int link_identifier)

Returns: **TRUE** on success, **FALSE** on error

sybase_close() closes the link to a Sybase database that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

Note that this isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

sybase_close() will not close persistent links generated by **sybase_pconnect()**.

See also: **sybase_connect()**, **sybase_pconnect()**.

sybase_connect

(PHP 3, PHP 4)

sybase_connect -- open Sybase server connection

Description

int **sybase_connect** (string servername, string username, string password [, string charset])

Returns: A positive Sybase link identifier on success, or **FALSE** on error.

sybase_connect() establishes a connection to a Sybase server. The servername argument has to be a valid servername that is defined in the 'interfaces' file.

In case a second call is made to **sybase_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling **sybase_close()**.

See also **sybase_pconnect()**, **sybase_close()**.

sybase_data_seek

(PHP 3, PHP 4)

sybase_data_seek -- move internal row pointer

Description

bool **sybase_data_seek** (int result_identifier, int row_number)

Returns: **TRUE** on success, **FALSE** on failure

sybase_data_seek() moves the internal row pointer of the Sybase result associated with the specified result identifier to pointer to the specifyed row number. The next call to **sybase_fetch_row()** would return that row.

See also: **sybase_data_seek()**.

sybase_fetch_array

(PHP 3, PHP 4)

sybase_fetch_array -- fetch row as array

Description

array **sybase_fetch_array** (int result)

Returns: An array that corresponds to the fetched row, or FALSE if there are no more rows.

sybase_fetch_array() is an extended version of **sybase_fetch_row()**. In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

An important thing to note is that using **sybase_fetch_array()** is NOT significantly slower than using **sybase_fetch_row()**, while it provides a significant added value.

For further details, also see **sybase_fetch_row()**.

sybase_fetch_field

(PHP 3, PHP 4)

sybase_fetch_field -- get field information

Description

object **sybase_fetch_field** (int result [, int field_offset])

Returns an object containing field information.

sybase_fetch_field() can be used in order to obtain information about fields in a certain query result. If the field offset isn't specified, the next field that wasn't yet retreived by **sybase_fetch_field()** is retreived.

The properties of the object are:

- name - column name. if the column is a result of a function, this property is set to computed#N, where #N is a serial number.
- column_source - the table from which the column was taken
- max_length - maximum length of the column
- numeric - 1 if the column is numeric
- type - datatype of the column

See also **sybase_field_seek()**

sybase_fetch_object

(PHP 3, PHP 4)

sybase_fetch_object -- fetch row as object

Description

int **sybase_fetch_object** (int result)

Returns: An object with properties that correspond to the fetched row, or FALSE if there are no more rows.

sybase_fetch_object() is similar to **sybase_fetch_array()**, with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

Speed-wise, the function is identical to **sybase_fetch_array()**, and almost as quick as **sybase_fetch_row()** (the difference is insignificant).

See also: **sybase_fetch_array()** and **sybase_fetch_row()**.

sybase_fetch_row

(PHP 3, PHP 4)

sybase_fetch_row -- get row as enumerated array

Description

array **sybase_fetch_row** (int result)

Returns: An array that corresponds to the fetched row, or FALSE if there are no more rows.

sybase_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to **sybase_fetch_row()** would return the next row in the result set, or FALSE if there are no more rows.

See also: sybase_fetch_array(), sybase_fetch_object(), sybase_data_seek(), **sybase_fetch_lengths()**, and sybase_result().

sybase_field_seek

(PHP 3, PHP 4)

sybase_field_seek -- set field offset

Description

int **sybase_field_seek** (int result, int field_offset)

Seeks to the specified field offset. If the next call to sybase_fetch_field() won't include a field offset, this field would be returned.

See also: sybase_fetch_field().

sybase_free_result

(PHP 3, PHP 4)

sybase_free_result -- free result memory

Description

bool **sybase_free_result** (int result)

sybase_free_result() only needs to be called if you are worried about using too much memory while your script is running. All result memory will automatically be freed when the script ends. You may call **sybase_free_result()** with the result identifier as an argument and the associated result memory will be freed.

sybase_get_last_message

(PHP 3, PHP 4)

sybase_get_last_message -- Returns the last message from the server

Description

string **sybase_get_last_message** (void)

sybase_get_last_message() returns the last message reported by the server.

sybase_min_client_severity

(PHP 3, PHP 4)

sybase_min_client_severity -- Sets minimum client severity

Description

void **sybase_min_client_severity** (int severity)

sybase_min_client_severity() sets the minimum client severity level.

Note: This function is only available using the CT library interface to Sybase, and not the DB library.

See also: **sybase_min_server_severity()**.

sybase_min_error_severity

(PHP 3, PHP 4)

sybase_min_error_severity -- Sets minimum error severity

Description

void **sybase_min_error_severity** (int severity)

sybase_min_error_severity() sets the minimum error severity level.

See also: **sybase_min_message_severity()**.

sybase_min_message_severity

(PHP 3, PHP 4)

sybase_min_message_severity -- Sets minimum message severity

Description

void **sybase_min_message_severity** (int severity)

sybase_min_message_severity() sets the minimum message severity level.

See also: **sybase_min_error_severity()**.

sybase_min_server_severity

(PHP 3, PHP 4)

sybase_min_server_severity -- Sets minimum server severity

Description

void **sybase_min_server_severity** (int severity)

sybase_min_server_severity() sets the minimum server severity level.

Note: This function is only available using the CT library interface to Sybase, and not the DB library.

See also: **sybase_min_client_severity()**.

sybase_num_fields

(PHP 3, PHP 4)

sybase_num_fields -- get number of fields in result

Description

int **sybase_num_fields** (int result)

sybase_num_fields() returns the number of fields in a result set.

See also: **sybase_db_query()**, sybase_query(), sybase_fetch_field(), sybase_num_rows().

sybase_num_rows

(PHP 3, PHP 4)

sybase_num_rows -- get number of rows in result

Description

int **sybase_num_rows** (int result)

sybase_num_rows() returns the number of rows in a result set.

See also: **sybase_db_query()**, sybase_query() and, sybase_fetch_row().

sybase_pconnect

(PHP 3, PHP 4)

sybase_pconnect -- open persistent Sybase connection

Description

int **sybase_pconnect** (string servername, string username, string password [, string charset])

Returns: A positive Sybase persistent link identifier on success, or FALSE on error

sybase_pconnect() acts very much like sybase_connect() with two major differences.

First, when connecting, the function would first try to find a (persistent) link that's already open with the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (sybase_close() will not close links established by **sybase_pconnect()()**).

This type of links is therefore called 'persistent'.

sybase_query

(PHP 3, PHP 4)

sybase_query -- send Sybase query

Description

int **sybase_query** (string query, int link_identifier)

Returns: A positive Sybase result identifier on success, or FALSE on error.

sybase_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if

sybase_connect() was called, and use it.

See also: **sybase_db_query()**, sybase_select_db(), and sybase_connect().

sybase_result

(PHP 3, PHP 4)

sybase_result -- get result data

Description

string **sybase_result** (int result, int row, mixed field)

Returns: The contents of the cell at the row and offset in the specified Sybase result set.

sybase_result() returns the contents of one cell from a Sybase result set. The field argument can be the field's offset, or the field's name, or the field's table dot field's name (tablename.fieldname). If the column name has been aliased ('select foo as bar from...'), use the alias instead of the column name.

When working on large result sets, you should consider using one of the functions that fetch an entire row (specified below). As these functions return the contents of multiple cells in one function call, they're MUCH quicker than sybase_result(). Also, note that specifying a numeric offset for the field argument is much quicker than specifying a fieldname or tablename.fieldname argument.

Recommended high-performance alternatives: sybase_fetch_row(), sybase_fetch_array(), and sybase_fetch_object().

sybase_select_db

(PHP 3, PHP 4)

sybase_select_db -- select Sybase database

Description

bool **sybase_select_db** (string database_name, int link_identifier)

Returns: **TRUE** on success, **FALSE** on error

sybase_select_db() sets the current active database on the server that's associated with the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if sybase_connect() was called, and use it.

Every subsequent call to sybase_query() will be made on the active database.

See also: sybase_connect(), sybase_pconnect(), and sybase_query()

CI. Tokenizer functions

Introduction

Warning
This extension is *EXPERIMENTAL*. The behaviour of this extension -- including the names of its functions and anything else documented about this extension -- may change without notice in a future release of PHP. Use this extension at your own risk.

See also the appendix about tokens.

Requirements

No external libraries are needed to build this extension.

Installation

Beginning with PHP 4.3.0 these functions are enabled by default. For older versions you have to configure and compile PHP with `--enable-tokenizer`. You can disable tokenizer support with `--disable-tokenizer`.

The windows version of PHP has built in support for this extension. You do not need to load any additional extension in order to use these functions.

> **Note:** Builtin support for tokenizer is available with PHP 4.3.0.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

T_INCLUDE (integer)

T_INCLUDE_ONCE (integer)

T_EVAL (integer)

T_REQUIRE (integer)

T_REQUIRE_ONCE (integer)

T_LOGICAL_OR (integer)

T_LOGICAL_XOR (integer)

T_LOGICAL_AND (integer)

T_PRINT (integer)

T_PLUS_EQUAL (integer)

T_MINUS_EQUAL (integer)

T_MUL_EQUAL (integer)

T_DIV_EQUAL (integer)

T_CONCAT_EQUAL (integer)

T_MOD_EQUAL (integer)

T_AND_EQUAL (integer)

T_OR_EQUAL (integer)

T_XOR_EQUAL (integer)

T_SL_EQUAL (integer)

T_SR_EQUAL (integer)

T_BOOLEAN_OR (integer)

T_BOOLEAN_AND (integer)

T_IS_EQUAL (integer)

T_IS_NOT_EQUAL (integer)

T_IS_IDENTICAL (integer)

T_IS_NOT_IDENTICAL (integer)

T_IS_SMALLER_OR_EQUAL (integer)

T_IS_GREATER_OR_EQUAL (integer)

T_SL (integer)

T_SR (integer)

T_INC (integer)

T_DEC (integer)

T_INT_CAST (integer)

T_DOUBLE_CAST (integer)

T_STRING_CAST (integer)

T_ARRAY_CAST (integer)

T_OBJECT_CAST (integer)

T_BOOL_CAST (integer)

T_UNSET_CAST (integer)

T_NEW (integer)

T_EXIT (integer)

T_IF (integer)

T_ELSEIF (integer)

T_ELSE (integer)

T_ENDIF (integer)

T_LNUMBER (integer)

T_DNUMBER (integer)

T_STRING (integer)

T_STRING_VARNAME (integer)

T_VARIABLE (integer)

T_NUM_STRING (integer)

T_INLINE_HTML (integer)

T_CHARACTER (integer)

T_BAD_CHARACTER (integer)

T_ENCAPSED_AND_WHITESPACE (integer)

T_CONSTANT_ENCAPSED_STRING (integer)

T_ECHO (integer)

T_DO (integer)

T_WHILE (integer)

T_ENDWHILE (integer)

T_FOR (integer)

T_ENDFOR (integer)

T_FOREACH (integer)

T_ENDFOREACH (integer)

T_DECLARE (integer)

T_ENDDECLARE (integer)

T_AS (integer)

T_SWITCH (integer)

T_ENDSWITCH (integer)

T_CASE (integer)

T_DEFAULT (integer)

T_BREAK (integer)

T_CONTINUE (integer)

T_OLD_FUNCTION (integer)

T_FUNCTION (integer)

T_CONST (integer)

T_RETURN (integer)

T_USE (integer)

T_GLOBAL (integer)

T_STATIC (integer)

T_VAR (integer)

T_UNSET (integer)

T_ISSET (integer)

T_EMPTY (integer)

T_CLASS (integer)

T_EXTENDS (integer)

T_OBJECT_OPERATOR (integer)

T_DOUBLE_ARROW (integer)

T_LIST (integer)

T_ARRAY (integer)

T_LINE (integer)

T_FILE (integer)

T_COMMENT (integer)

T_ML_COMMENT (integer)

T_OPEN_TAG (integer)

T_OPEN_TAG_WITH_ECHO (integer)

T_CLOSE_TAG (integer)

T_WHITESPACE (integer)

T_START_HEREDOC (integer)

T_END_HEREDOC (integer)

T_DOLLAR_OPEN_CURLY_BRACES (integer)

T_CURLY_OPEN (integer)

T_PAAMAYIM_NEKUDOTAYIM (integer)

T_DOUBLE_COLON (integer)

Table of Contents

token_get_all

(PHP 4 >= 4.2.0)

token_get_all -- Split given source in tokens

Description

array **token_get_all** (string source)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

token_name

(PHP 4 >= 4.2.0)

token_name -- Get the name of a given token

Description

string **token_name** (int type)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

CII. URL Functions

Introduction

Dealing with URL strings: encoding, decoding and parsing.

Requirements

No external libraries are needed to build this extension.

Installation

There is no installation needed to use these functions; they are part of the PHP core.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Table of Contents

base64_decode

(PHP 3, PHP 4)

base64_decode -- Decodes data encoded with MIME base64

Description

string **base64_decode** (string encoded_data)

base64_decode() decodes *encoded_data* and returns the original data. The returned data may be binary.

See also: **base64_encode()**, RFC 2045 section 6.8.

base64_encode

(PHP 3, PHP 4)

base64_encode -- Encodes data with MIME base64

Description

string **base64_encode** (string data)

base64_encode() returns *data* encoded with base64. This encoding is designed to make binary data survive transport through transport layers that are not 8-bit clean, such as mail bodies.

Base64-encoded data takes about 33% more space than the original data.

See also: **base64_decode()**, **chunk_split()**, RFC 2045 section 6.8.

get_meta_tags

(PHP 3>= 3.0.4, PHP 4)

get_meta_tags -- Extracts all meta tag content attributes from a file and returns an array

Description

array **get_meta_tags** (string filename [, int use_include_path])

Opens *filename* and parses it line by line for <meta> tags in the file. This can be a local file or an URL. The parsing stops at </head>.

Setting *use_include_path* to 1 will result in PHP trying to open the file along the standard include path as per the include_path directive. This is used for local files, not URLs.

Example 1. What get_meta_tags() parses

```
<meta name="author" content="name">
<meta name="keywords" content="php documentation">
<meta name="DESCRIPTION" content="a php manual">
<meta name="geo.position" content="49.33;-86.59">
</head> <!-- parsing stops here -->
```

(pay attention to line endings - PHP uses a native function to parse the input, so a Mac file won't work on Unix).

The value of the name property becomes the key, the value of the content property becomes the value of the returned array, so you can easily use standard array functions to traverse it or access single values. Special characters in the value of the name property are substituted with '_', the rest is converted to lower case. If two meta tags have the same name, only the last one is returned.

Example 2. What get_meta_tags() returns

```
<?php
// Assuming the above tags are at example.com
$tags = get_meta_tags('http://www.example.com/');

// Notice how the keys are all lowercase now, and
// how . was replaced by _ in the key.
print $tags['author'];       // name
print $tags['keywords'];     // php documentation
print $tags['description'];  // a php manual
print $tags['geo_position']; // 49.33;-86.59
?>
```

> **Note:** As of PHP 4.0.5, **get_meta_tags()** supports unquoted html attributes.

See also **htmlentities()** and **urlencode()**.

parse_url

(PHP 3, PHP 4)

parse_url -- Parse a URL and return its components

Description

array **parse_url** (string url)

This function returns an associative array returning any of the various components of the URL that are present. This includes the

- *scheme* - e.g. http
- *host*
- *port*
- *user*
- *pass*
- *path*

- *query* - after the question mark ?
- *fragment* - after the hashmark #

This function is **not** meant to validate the given URL, it only breaks it up into the above listed parts. Partial urls are also accepted, **parse_url()** tries its best to parse them correctly.

Example 1. Using parse_url()

```
$ php -r 'print_r( parse_url("http://username:password@hostname/path?arg=value#anchor"));'
Array
(
    [scheme] => http
    [host] => hostname
    [user] => username
    [pass] => password
    [path] => /path
    [query] => arg=value
    [fragment] => anchor
)

$ php -r 'print_r( parse_url("http://invalid_host..name/"));'
Array
(
    [scheme] => http
    [host] => invalid_host..name
    [path] => /
)
```

See also **pathinfo()**, **parse_str()**, **dirname()**, and **basename()**.

rawurldecode

(PHP 3, PHP 4)

rawurldecode -- Decode URL-encoded strings

Description

string **rawurldecode** (string str)

Returns a string in which the sequences with percent (%) signs followed by two hex digits have been replaced with literal characters. For example, the string

```
foo%20bar%40baz
```

decodes into

```
foo bar@baz
```

.

> **Note: rawurldecode()** does not decode plus symbols ('+') into spaces. **urldecode()** does.

See also **rawurlencode()**, **urldecode()**, **urlencode()**.

rawurlencode

(PHP 3, PHP 4)

rawurlencode -- URL-encode according to RFC 1738

Description

string **rawurlencode** (string str)

Returns a string in which all non-alphanumeric characters except

```
-_.
```

have been replaced with a percent (%) sign followed by two hex digits. This is the encoding described in RFC 1738 for protecting literal characters from being interpreted as special URL delimiters, and for protecting URL's from being mangled by transmission media with character conversions (like some email systems). For example, if you want to include a password in an FTP URL:

Example 1. rawurlencode() example 1

```
echo '<a href="ftp://user:', rawurlencode('foo @+%/'),
     '@ftp.my.com/x.txt">';
```

Or, if you pass information in a PATH_INFO component of the URL:

Example 2. rawurlencode() example 2

```
echo '<a href="http://x.com/department_list_script/',
    rawurlencode('sales and marketing/Miami'), '">';
```

See also **rawurldecode()**, **urldecode()**, **urlencode()** and RFC 1738

urldecode

(PHP 3, PHP 4)

urldecode -- Decodes URL-encoded string

Description

string **urldecode** (string str)

Decodes any %## encoding in the given string. The decoded string is returned.

Example 1. urldecode() example

```
$a = explode('&', $QUERY_STRING);
$i = 0;
while ($i < count($a)) {
    $b = split('=', $a[$i]);
    echo 'Value for parameter ', htmlspecialchars(urldecode($b[0])),
         ' is ', htmlspecialchars(urldecode($b[1])), "<br />\n";
    $i++;
}
```

See also **urlencode()**, **rawurlencode()**, **rawurldecode()**.

urlencode

(PHP 3, PHP 4)

urlencode -- URL-encodes string

Description

string **urlencode** (string str)

Returns a string in which all non-alphanumeric characters except -_. have been replaced with a percent (%) sign followed by two hex digits and spaces encoded as plus (+) signs. It is encoded the same way that the posted data from a WWW form is encoded, that is the same way as in `application/x-www-form-urlencoded` media type. This differs from the RFC1738 encoding (see **rawurlencode()**) in that for historical reasons, spaces are encoded as plus (+) signs. This function is convenient when encoding a string to be used in a query part of an URL, as a convenient way to pass variables to the next page:

Example 1. urlencode() example

```
echo '<a href="mycgi?foo=', urlencode($userinput), '">';
```

Note: Be careful about variables that may match HTML entities. Things like &, &copy and &pound are parsed by the browser and the actual entity is used instead of the desired variable name. This is an obvious hassle that the W3C has been telling people about for years. The reference is here: http://www.w3.org/TR/html4/appendix/notes.html#h-B.2.2 PHP supports changing the argument separator to the W3C-suggested semi-colon through the arg_separator .ini directive. Unfortunately most user agents do not send form data in this semi-colon separated format. A more portable way around this is to use & instead of & as the separator. You don't need to change PHP's arg_separator for this. Leave it as &, but simply encode your URLs using **htmlentities()**(urlencode($data)).

Example 2. urlencode/htmlentities() example

```
echo '<a href="mycgi?foo=', htmlentities(urlencode($userinput)), '">';
```

See also **urldecode()**, **htmlentities()**, **rawurldecode()**, **rawurlencode()**.

CIII. Variable Functions

Introduction

For information on how variables behave, see the Variables entry in the Language Reference section of the manual.

Requirements

No external libraries are needed to build this extension.

Installation

There is no installation needed to use these functions; they are part of the PHP core.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. Variables Configuration Options

Name	Default	Changeable
unserialize_callback_func	""	PHP_INI_ALL

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Here is a short explanation of the configuration directives.

`unserialize_callback_func` **string**

The unserialize callback function will called (with the undefind class' name as parameter), if the unserializer finds an undefined class which should be instanciated. A warning appears if the specified function is not defined, or if the function doesn't include/implement the missing class. So only set this entry, if you really want to implement such a callback-function.

See also **unserialize()**.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Table of Contents

doubleval

doubleval -- Alias of **floatval()**

Description

This function is an alias of **floatval()**.

> **Note:** This alias is a left-over from a function-renaming. In older versions of PHP you'll need to use this alias of the **floatval()** function, because **floatval()** wasn't yet available in that version.

empty

(PHP 3, PHP 4)

empty -- Determine whether a variable is set

Description

bool **empty** (mixed var)

> **Note: empty()** is a language construct.

This is the opposite of `(boolean) var`, except that no warning is generated when the variable is not set. See converting to boolean for more information.

```
$var = 0;

if (empty($var)) {  // evaluates true
    echo '$var is either 0 or not set at all';
}

if (!isset($var)) { // evaluates false
    echo '$var is not set at all';
}
```

Note that this is meaningless when used on anything which isn't a variable; i.e. **empty (addslashes ($name))** has no meaning since it would be checking whether something which isn't a variable is a variable with a **FALSE** value.

See also **isset()** and **unset()**.

floatval

(PHP 4 >= 4.2.0)

floatval -- Get float value of a variable

Description

float **floatval** (mixed var)

Returns the **float** value of *var*.

var may be any scalar type. You cannot use **floatval()** on arrays or objects.

```
$var = '122.34343The';
$float_value_of_var = floatval ($var);
print $float_value_of_var; // prints 122.34343
```

See also **intval()**, **strval()**, **settype()** and Type juggling.

get_defined_vars

(PHP 4 >= 4.0.4)

get_defined_vars -- Returns an array of all defined variables

Description

array **get_defined_vars** (void)

This function returns an multidimensional array containing a list of all defined variables, be them environment, server or user-defined variables.

```
$b = array(1,1,2,3,5,8);

$arr = get_defined_vars();

// print $b
print_r($arr["b"]);

// print path to the PHP interpreter (if used as a CGI)
// e.g. /usr/local/bin/php
echo $arr["_"];

// print the command-line paramaters if any
print_r($arr["argv"]);

// print all the server vars
print_r($arr["_SERVER"]);

// print all the available keys for the arrays of variables
print_r(array_keys(get_defined_vars()));
```

See also **get_defined_functions()** and **get_defined_constants()**.

get_resource_type

(PHP 4 >= 4.0.2)

get_resource_type -- Returns the resource type

Description

string **get_resource_type** (resource handle)

This function returns a string representing the type of the **resource** passed to it. If the paramater is not a valid **resource**, it generates an error.

```
$c = mysql_connect();
echo get_resource_type($c)."\n";
// prints: mysql link

$fp = fopen("foo","w");
echo get_resource_type($fp)."\n";
// prints: file

$doc = new_xmldoc("1.0");
echo get_resource_type($doc->doc)."\n";
// prints: domxml document
```

gettype

(PHP 3, PHP 4)

gettype -- Get the type of a variable

Description

string **gettype** (mixed var)

Returns the type of the PHP variable *var*.

Warning
Never use **gettype()** to test for a certain type, since the returned string may be subject to change in a future version. In addition, it is slow too, as it involves string comparision . Instead, use the `is_*` functions.

Possibles values for the returned string are:

- "**boolean**" (since PHP 4)
- "**integer**"
- "**double**" (for historical reasons "double" is returned in case of a **float**, and not simply "float")
- "**string**"
- "**array**"
- "**object**"
- "**resource**" (since PHP 4)
- "NULL" (since PHP 4)
- "user function" (PHP 3 only, deprecated)
- "unknown type"

For PHP 4, you should use **function_exists()** and **method_exists()** to replace the prior usage of **gettype()** on a function.

See also **settype()**, **is_array()**, **is_bool()**, **is_float()**, **is_integer()**, **is_null()**, **is_numeric()**, **is_object()**, **is_resource()**, **is_scalar()**, and **is_string()**.

import_request_variables

(PHP 4 >= 4.1.0)

import_request_variables -- Import GET/POST/Cookie variables into the global scope

Description

bool **import_request_variables** (string types [, string prefix])

Imports GET/POST/Cookie variables into the global scope. It is useful if you disabled register_globals, but would like to see some variables in the global scope.

Using the *types* parameter, you can specify which request variables to import. You can use 'G', 'P' and 'C' characters respectively for GET, POST and Cookie. These characters are not case sensitive, so you can also use any combination of 'g', 'p' and 'c'. POST includes the POST uploaded file information. Note that the order of the letters matters, as when using "gp", the POST variables will overwrite GET variables with the same name. Any other letters than GPC are discarded.

The *prefix* parameter is used as a variable name prefix, prepended before all variable's name imported into the global scope. So if you have a GET value named "userid", and provide a prefix "pref_", then you'll get a global variable named $pref_userid.

If you're interested in importing other variables into the global scope, such as SERVER, consider using **extract()**.

> **Note:** Although the *prefix* parameter is optional, you will get an E_NOTICE level error if you specify no prefix, or specify an empty string as a prefix. This is a possible security hazard. Notice level errors are not displayed using the default error reporting level.

```
// This will import GET and POST vars
// with an "rvar_" prefix
import_request_variables("gP", "rvar_");

print $rvar_foo;
```

See also $_REQUEST, register_globals, Predefined Variables, and **extract()**.

intval

(PHP 3, PHP 4)

intval -- Get integer value of a variable

Description

int **intval** (mixed var [, int base])

Returns the **integer** value of *var*, using the specified base for the conversion (the default is base 10).

var may be any scalar type. You cannot use **intval()** on **array**s or **object**s.

> **Note:** The *base* argument for **intval()** has no effect unless the *var* argument is a string.

See also **floatval()**, **strval()**, **settype()** and Type juggling.

is_array

(PHP 3, PHP 4)

is_array -- Finds whether a variable is an array

Description

bool **is_array** (mixed var)

Returns **TRUE** if *var* is an **array**, **FALSE** otherwise.

See also **is_float()**, **is_int()**, **is_integer()**, **is_string()**, and **is_object()**.

is_bool

(PHP 4)

is_bool -- Finds out whether a variable is a boolean

Description

bool **is_bool** (mixed var)

Returns **TRUE** if the *var* parameter is a **boolean**.

See also **is_array()**, **is_float()**, **is_int()**, **is_integer()**, **is_string()**, and **is_object()**.

is_callable

(PHP 4 >= 4.0.6)

is_callable -- Find out whether the argument is a valid callable construct

Description

bool **is_callable** (mixed var [, bool syntax_only [, string callable_name]])

Warning
This function is currently not documented; only the argument list is available.

is_double

is_double -- Alias of **is_float()**

Description

This function is an alias of **is_float()**.

is_float

(PHP 3, PHP 4)

is_float -- Finds whether a variable is a float

Description

bool **is_float** (mixed var)

Returns **TRUE** if *var* is a **float**, **FALSE** otherwise.

> **Note:** To test if a variable is a number or a numeric string (such as form input, which is always a string), you must use **is_numeric()**.

See also **is_bool()**, **is_int()**, **is_integer()**, **is_numeric()**, **is_string()**, **is_array()**, and **is_object()**,

is_int

(PHP 3, PHP 4)

is_int -- Find whether a variable is an integer

Description

bool **is_int** (mixed var)

Returns **TRUE** if *var* is an **integer** **FALSE** otherwise.

> **Note:** To test if a variable is a number or a numeric string (such as form input, which is always a string), you must use

is_numeric().

See also is_bool(), is_float(), is_integer(), is_numeric(), is_string(), is_array(), and is_object().

is_integer

is_integer -- Alias of is_int()

Description

This function is an alias of is_int().

is_long

is_long -- Alias of is_int()

Description

This function is an alias of is_int().

is_null

(PHP 4 >= 4.0.4)

is_null -- Finds whether a variable is NULL

Description

bool **is_null** (mixed var)

Returns TRUE if *var* is null, FALSE otherwise.

See the NULL type when a variable is considered to be NULL and when not.

See also NULL, is_bool(), is_numeric(), is_float(), is_int(), is_string(), is_object(), is_array(), is_integer(), and is_real()

is_numeric

(PHP 4)

is_numeric -- Finds whether a variable is a number or a numeric string

Description

bool **is_numeric** (mixed var)

Returns TRUE if *var* is a number or a numeric string, FALSE otherwise.

See also is_bool(), is_float(), is_int(), is_string(), is_object(), is_array(), and is_integer().

is_object

(PHP 3, PHP 4)

is_object -- Finds whether a variable is an object

Description

bool **is_object** (mixed var)

Returns **TRUE** if *var* is an **object**, **FALSE** otherwise.

See also **is_bool()**, **is_int()**, **is_integer()**, **is_float()**, **is_string()**, and **is_array()**.

is_real

is_real -- Alias of **is_float()**

Description

This function is an alias of **is_float()**.

is_resource

(PHP 4)

is_resource -- Finds whether a variable is a resource

Description

bool **is_resource** (mixed var)

is_resource() returns **TRUE** if the variable given by the *var* parameter is a **resource**, otherwise it returns **FALSE**.

See the documentation on the **resource**-type for more information.

is_scalar

(PHP 4 >= 4.0.5)

is_scalar -- Finds whether a variable is a scalar

Description

bool **is_scalar** (mixed var)

is_scalar() returns **TRUE** if the variable given by the *var* parameter is a scalar, otherwise it returns **FALSE**.

Scalar variables are those containing an **integer**, **float**, **string** or **boolean**. Types **array**, **object** and **resource** or not scalar.

```
function show_var($var) {
    if (is_scalar($var)) {
        echo $var;
    } else {
        var_dump($var);
    }
}
$pi = 3.1416;
$proteins = array("hemoglobin", "cytochrome c oxidase", "ferredoxin");

show_var($pi);
// prints: 3.1416

show_var($proteins)
// prints:
// array(3) {
//   [0]=>
//   string(10) "hemoglobin"
//   [1]=>
//   string(20) "cytochrome c oxidase"
//   [2]=>
//   string(10) "ferredoxin"
// }
```

> **Note: is_scalar()** does not consider **resource** type values to be scalar as resources are abstract datatypes which are currently based on integers. This implementation detail should not be relied upon, as it may change.

See also **is_bool()**, **is_numeric()**, **is_float()**, **is_int()**, **is_real()**, **is_string()**, **is_object()**, **is_array()**, and **is_integer()**.

is_string

(PHP 3, PHP 4)

is_string -- Finds whether a variable is a string

Description

bool **is_string** (mixed var)

Returns **TRUE** if *var* is a **string**, **FALSE** otherwise.

See also **is_bool()**, **is_int()**, **is_integer()**, **is_float()**, **is_real()**, **is_object()**, and **is_array()**.

isset

(PHP 3, PHP 4)

isset -- Determine whether a variable is set

Description

bool **isset** (mixed var [, mixed var [, ...]])

> **Note: isset()** is a language construct.

Returns **TRUE** if *var* exists; **FALSE** otherwise.

If a variable has been unset with **unset()**, it will no longer be **isset()**. **isset()** will return **FALSE** if testing a variable that has been set to **NULL**. Also note that a **NULL** byte ("\0") is not equivalent to the PHP **NULL** constant.

> **Warning: isset()** only works with variables as passing anything else will result in a parse error. For checking if constants are set use the **defined()** function.

```
<?php

$var = '';

// This will evaluate to &true; so the text will be printed.
if (isset($var)) {
    print "This var is set set so I will print.";
}

// In the next examples we'll use var_dump to output
// the return value of isset().

$a = "test";
$b = "anothertest";

var_dump( isset($a) );      // TRUE
var_dump( isset ($a, $b) ); // TRUE

unset ($a);

var_dump( isset ($a) );     // FALSE
var_dump( isset ($a, $b) ); // FALSE

$foo = NULL;
var_dump( isset ($foo) );   // FALSE

?>
```

This also work for elements in arrays:

```
<?php

$a = array ('test' => 1, 'hello' => NULL);

var_dump( isset ($a['test']) );            // TRUE
var_dump( isset ($a['foo']) );             // FALSE
var_dump( isset ($a['hello']) );           // FALSE

// The key 'hello' equals NULL so is considered unset
// If you want to check for NULL key values then try:
var_dump( array_key_exists('hello', $a) ); // TRUE
```

```
?>
```

See also **empty()**, **unset()**, **defined()**, **array_key_exists()** and the error control **@** operator.

print_r

(PHP 4)

print_r -- Prints human-readable information about a variable

Description

bool **print_r** (mixed expression)

print_r() displays information about a variable in a way that's readable by humans. If given a **string**, **integer** or **float**, the value itself will be printed. If given an **array**, values will be presented in a format that shows keys and elements. Similar notation is used for **object**s.

Remember that **print_r()** will move the array pointer to the end. Use **reset()** to bring it back to beginning.

> **Tip:** As with anything that outputs its result directly to the browser, you can use the output-control functions to capture the output of this function, and save it in a **string** (for example).

```
<pre>
<?php
    $a = array ('a' => 'apple', 'b' => 'banana', 'c' => array ('x','y','z'));
    print_r ($a);
?>
</pre>
```

Which will output:

```
<pre>
Array
(
    [a] => apple
    [b] => banana
    [c] => Array
        (
            [0] => x
            [1] => y
            [2] => z
        )
)
</pre>
```

> **Note:** Prior to PHP 4.0.4, **print_r()** will continue forever if given an **array** or **object** that contains a direct or indirect reference to itself. An example is `print_r($GLOBALS)` because `$GLOBALS` is itself a global variable that contains a reference to itself.

See also **ob_start()**, **var_dump()**, and **var_export()**.

serialize

(PHP 3>= 3.0.5, PHP 4)

serialize -- Generates a storable representation of a value

Description

string **serialize** (mixed value)

serialize() returns a string containing a byte-stream representation of *value* that can be stored anywhere.

This is useful for storing or passing PHP values around without losing their type and structure.

To make the serialized string into a PHP value again, use **unserialize()**. **serialize()** handles all types, except the **resource**-type. You can even **serialize()** arrays that contain references to itself. References inside the array/object you are **serialize()**ing will also be stored.

When serializing objects, PHP will attempt to call the member function __**sleep()** prior to serialization. This is to allow the object

to do any last minute clean-up, etc. prior to being serialized. Likewise, when the object is restored using **unserialize()** the **__wakeup()** member function is called.

> **Note:** In PHP 3, object properties will be serialized, but methods are lost. PHP 4 removes that limitation and restores both properties and methods. Please see the Serializing Objects section of Classes and Objects for more information.

Example 1. serialize() example

```
// $session_data contains a multi-dimensional array with session
// information for the current user.  We use serialize() to store
// it in a database at the end of the request.

$conn = odbc_connect ("webdb", "php", "chicken");
$stmt = odbc_prepare ($conn,
      "UPDATE sessions SET data = ? WHERE id = ?");
$sqldata = array (serialize($session_data), $PHP_AUTH_USER);
if (!odbc_execute ($stmt, &$sqldata)) {
    $stmt = odbc_prepare($conn,
     "INSERT INTO sessions (id, data) VALUES(?, ?)");
    if (!odbc_execute($stmt, &$sqldata)) {
    /* Something went wrong.  Bitch, whine and moan. */
    }
}
```

See Also: **unserialize()**.

settype

(PHP 3, PHP 4)

settype -- Set the type of a variable

Description

bool **settype** (mixed var, string type)

Set the type of variable *var* to *type*.

Possibles values of *type* are:

- "boolean" (or, since PHP 4.2.0, "bool")
- "integer" (or, since PHP 4.2.0, "int")
- "float" (only possible since PHP 4.2.0, for older versions use the deprecated variant "double")
- "string"
- "array"
- "object"
- "null" (since PHP 4.2.0)

Returns **TRUE** on success or **FALSE** on failure.

Example 1. settype() example

```
$foo = "5bar"; // string
$bar = true;   // boolean

settype($foo, "integer"); // $foo is now 5   (integer)
settype($bar, "string");  // $bar is now "1" (string)
```

See also **gettype()**, type-casting and type-juggling.

strval

(PHP 3, PHP 4)

strval -- Get string value of a variable

Description

string **strval** (mixed var)

Returns the **string** value of *var*. See the documentation on **string** for more information on converting to string.

var may be any scalar type. You cannot use **strval()** on arrays or objects.

See also **floatval()**, **intval()**, **settype()** and Type juggling.

unserialize

(PHP 3>= 3.0.5, PHP 4)

unserialize -- Creates a PHP value from a stored representation

Description

mixed **unserialize** (string str [, string callback])

unserialize() takes a single serialized variable (see **serialize()**) and converts it back into a PHP value. The converted value is returned, and can be an **integer**, **float**, **string**, **array** or **object**. In case the passed string is not unserializeable, **FALSE** is returned.

> **unserialize_callback_func directive:** It's possible to set a callback-function which will be called, if an undefined class should be instantiated during unserializing. (to prevent getting an incomplete **object** "__PHP_Incomplete_Class".) Use your `php.ini`, **ini_set()** or `.htaccess` to define 'unserialize_callback_func'. Everytime an undefined class should be instantiated, it'll be called. To disable this feature just empty this setting. Also note that the directive unserialize_callback_func directive became available in PHP 4.2.0.
>
> **Note:** The *callback* parameter was added in PHP 4.2.0

If the variable being unserialized is an object, after successfully reconstructing the object PHP will automatically attempt to call the **__wakeup()** member function (if it exists).

Example 1. unserialize_callback_func example

```
<?php
$serialized_object='O:1:"a":1:{s:5:"value";s:3:"100";}';

// unserialize_callback_func directive available as of PHP 4.2.0
ini_set('unserialize_callback_func','mycallback'); // set your callback_function

function mycallback($classname) {
    // just include a file containing your classdefinition
    // you get $classname to figure out which classdefinition is required
}
?>
```

> **Note:** In PHP 3, methods are not preserved when unserializing a serialized object. PHP 4 removes that limitation and restores both properties and methods. Please see the Serializing Objects section of Classes and Objects or more information.

Example 2. unserialize() example

```
<?php
// Here, we use unserialize() to load session data to the
// $session_data array from the string selected from a database.
// This example complements the one described with serialize().

$conn = odbc_connect ("webdb", "php", "chicken");
$stmt = odbc_prepare ($conn, "SELECT data FROM sessions WHERE id = ?");
$sqldata = array ($PHP_AUTH_USER);
if (!odbc_execute ($stmt, &$sqldata) || !odbc_fetch_into ($stmt, &$tmp)) {
    // if the execute or fetch fails, initialize to empty array
    $session_data = array();
} else {
    // we should now have the serialized data in $tmp[0].
    $session_data = unserialize ($tmp[0]);
    if (!is_array ($session_data)) {
        // something went wrong, initialize to empty array
        $session_data = array();
    }
}
?>
```

See Also: **serialize()**.

unset

(PHP 3, PHP 4)

unset -- Unset a given variable

Description

void **unset** (mixed var [, mixed var [, ...]])

> **Note: unset()** is a language construct.

unset() destroys the specified variables. Note that in PHP 3, **unset()** will always return **TRUE** (actually, the integer value 1). In PHP 4, however, **unset()** is no longer a true function: it is now a statement. As such no value is returned, and attempting to take the value of **unset()** results in a parse error.

Example 1. unset() example

```
// destroy a single variable
unset ($foo);

// destroy a single element of an array
unset ($bar['quux']);

// destroy more than one variable
unset ($foo1, $foo2, $foo3);
```

The behavior of **unset()** inside of a function can vary depending on what type of variable you are attempting to destroy.

If a globalized variable is **unset()** inside of a function, only the local variable is destroyed. The variable in the calling environment will retain the same value as before **unset()** was called.

```
function destroy_foo() {
    global $foo;
    unset($foo);
}

$foo = 'bar';
destroy_foo();
echo $foo;
```

The above example would output:

```
bar
```

If a variable that is PASSED BY REFERENCE is **unset()** inside of a function, only the local variable is destroyed. The variable in the calling environment will retain the same value as before **unset()** was called.

```
function foo(&$bar) {
    unset($bar);
    $bar = "blah";
}

$bar = 'something';
echo "$bar\n";

foo($bar);
echo "$bar\n";
```

The above example would output:

```
something
something
```

If a static variable is **unset()** inside of a function, **unset()** destroyes the variable and all its references.

```
function foo() {
    static $a;
    $a++;
    echo "$a\n";
    unset($a);
}
```

```
foo();
foo();
foo();
```

The above example would output:

```
1
2
3
```

If you would like to **unset()** a global variable inside of a function, you can use the $GLOBALS array to do so:

```
function foo() {
    unset($GLOBALS['bar']);
}

$bar = "something";
foo();
```

See also **isset()** and **empty()**.

var_dump

(PHP 3>= 3.0.5, PHP 4)

var_dump -- Dumps information about a variable

Description

void **var_dump** (mixed expression [, mixed expression [, ...]])

This function returns structured information about one or more expressions that includes its type and value. Arrays are explored recursively with values indented to show structure.

> **Tip:** As with anything that outputs its result directly to the browser, you can use the output-control functions to capture the output of this function, and save it in a **string** (for example).

Compare **var_dump()** to **print_r()**.

Example 1. var_dump() example

```
<pre>
<?php
$a = array (1, 2, array ("a", "b", "c"));
var_dump ($a);

/* output:
array(3) {
  [0]=>
  int(1)
  [1]=>
  int(2)
  [2]=>
  array(3) {
    [0]=>
    string(1) "a"
    [1]=>
    string(1) "b"
    [2]=>
    string(1) "c"
  }
}

*/

$b = 3.1;
$c = TRUE;
var_dump($b,$c);

/* output:
float(3.1)
bool(true)

*/
?>
</pre>
```

var_export

(PHP 4 >= 4.2.0)

var_export -- Outputs or returns a string representation of a variable

Description

mixed **var_export** (mixed expression [, bool return])

This function returns structured information about the variable that is passed to this function. It is similar to **var_dump()** with the exception that the returned representation is valid PHP code.

You can also return the variable representation by using **TRUE** as second parameter to this function.

Compare **var_export()** to **var_dump()**.

```
<pre>
<?php
$a = array (1, 2, array ("a", "b", "c"));
var_export ($a);

/* output:
array (
  0 => 1,
  1 => 2,
  2 =>
  array (
    0 => 'a',
    1 => 'b',
    2 => 'c',
  ),
)
*/

$b = 3.1;
$v = var_export($b, TRUE);
echo $v;

/* output:
3.1
*/
?>
</pre>
```

CIV. vpopmail functions

Introduction

Warning
This extension is *EXPERIMENTAL*. The behaviour of this extension -- including the names of its functions and anything else documented about this extension -- may change without notice in a future release of PHP. Use this extension at your own risk.

Table of Contents

vpopmail_add_alias_domain_ex

(4.0.5 - 4.2.3 only)

vpopmail_add_alias_domain_ex -- Add alias to an existing virtual domain

Description

bool **vpopmail_add_alias_domain_ex** (string olddomain, string newdomain)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

vpopmail_add_alias_domain

(4.0.5 - 4.2.3 only)

vpopmail_add_alias_domain -- Add an alias for a virtual domain

Description

bool **vpopmail_add_alias_domain** (string domain, string aliasdomain)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

vpopmail_add_domain_ex

(4.0.5 - 4.2.3 only)

vpopmail_add_domain_ex -- Add a new virtual domain

Description

bool **vpopmail_add_domain_ex** (string domain, string passwd [, string quota [, string bounce [, bool apop]]])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

vpopmail_add_domain

(4.0.5 - 4.2.3 only)

vpopmail_add_domain -- Add a new virtual domain

Description

bool **vpopmail_add_domain** (string domain, string dir, int uid, int gid)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

vpopmail_add_user

(4.0.5 - 4.2.3 only)

vpopmail_add_user -- Add a new user to the specified virtual domain

Description

bool **vpopmail_add_user** (string user, string domain, string password [, string gecos [, bool apop]])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

vpopmail_alias_add

(4.1.0 - 4.2.3 only)

vpopmail_alias_add -- insert a virtual alias

Description

bool **vpopmail_alias_add** (string user, string domain, string alias)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

vpopmail_alias_del_domain

(4.1.0 - 4.2.3 only)

vpopmail_alias_del_domain -- deletes all virtual aliases of a domain

Description

bool **vpopmail_alias_del_domain** (string domain)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

vpopmail_alias_del

(4.1.0 - 4.2.3 only)

vpopmail_alias_del -- deletes all virtual aliases of a user

Description

bool **vpopmail_alias_del** (string user, string domain)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

vpopmail_alias_get_all

(4.1.0 - 4.2.3 only)

vpopmail_alias_get_all -- get all lines of an alias for a domain

Description

array **vpopmail_alias_get_all** (string domain)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

vpopmail_alias_get

(4.1.0 - 4.2.3 only)

vpopmail_alias_get -- get all lines of an alias for a domain

Description

array **vpopmail_alias_get** (string alias, string domain)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

vpopmail_auth_user

(4.0.5 - 4.2.3 only)

vpopmail_auth_user -- Attempt to validate a username/domain/password. Returns true/false

Description

bool **vpopmail_auth_user** (string user, string domain, string password [, string apop])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

vpopmail_del_domain_ex

(4.0.5 - 4.2.3 only)

vpopmail_del_domain_ex -- Delete a virtual domain

Description

bool **vpopmail_del_domain_ex** (string domain)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

vpopmail_del_domain

(4.0.5 - 4.2.3 only)

vpopmail_del_domain -- Delete a virtual domain

Description

bool **vpopmail_del_domain** (string domain)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

vpopmail_del_user

(4.0.5 - 4.2.3 only)

vpopmail_del_user -- Delete a user from a virtual domain

Description

bool **vpopmail_del_user** (string user, string domain)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

vpopmail_error

(4.0.5 - 4.2.3 only)

vpopmail_error -- Get text message for last vpopmail error. Returns string

Description

string **vpopmail_error** (void)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

vpopmail_passwd

(4.0.5 - 4.2.3 only)

vpopmail_passwd -- Change a virtual user's password

Description

bool **vpopmail_passwd** (string user, string domain, string password)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

vpopmail_set_user_quota

(4.0.5 - 4.2.3 only)

vpopmail_set_user_quota -- Sets a virtual user's quota

Description

bool **vpopmail_set_user_quota** (string user, string domain, string quota)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

CV. W32api functions

Introduction

This extension is a generic extension API to DLLs. This was originally written to allow access to the Win32 API from PHP, although you can also access other functions exported via other DLLs.

Currently supported types are generic PHP types (strings, booleans, floats, integers and nulls) and types you define with **w32api_deftype()**.

Warning

This extension is *EXPERIMENTAL*. The behaviour of this extension -- including the names of its functions and anything else documented about this extension -- may change without notice in a future release of PHP. Use this extension at your own risk.

Requirements

This extension will only work on Windows systems.

Installation

There is no installation needed to use these functions; they are part of the PHP core.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension defines one resource type, used for user defined types. The name of this resource is `"dynaparm"`.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

DC_MICROSOFT (integer)

DC_BORLAND (integer)

DC_CALL_CDECL (integer)

DC_CALL_STD (integer)

DC_RETVAL_MATH4 (integer)

DC_RETVAL_MATH8 (integer)

DC_CALL_STD_BO (integer)

DC_CALL_STD_MS (integer)

DC_CALL_STD_M8 (integer)

DC_FLAG_ARGPTR (integer)

Examples

This example gets the amount of time the system has been running and displays it in a message box.

Example 1. Get the uptime and display it in a message box

```
<?php
// Define constants needed, taken from
// Visual Studio/Tools/Winapi/WIN32API.txt
define("MB_OK", 0);

// Load the extension in
dl("php_w32api.dll");

// Register the GetTickCount function from kernel32.dll
w32api_register_function("kernel32.dll",
                         "GetTickCount",
                         "long");

// Register the MessageBoxA function from User32.dll
w32api_register_function("User32.dll",
                         "MessageBoxA",
                         "long");

// Get uptime information
$ticks = GetTickCount();

// Convert it to a nicely displayable text
$secs  = floor($ticks / 1000);
$mins  = floor($secs / 60);
$hours = floor($mins / 60);

$str = sprintf("You have been using your computer for:".
               "\r\n %d Milliseconds, or \r\n %d Seconds".
               "or \r\n %d mins or\r\n %d hours %d mins.",
               $ticks,
               $secs,
               $mins,
               $hours,
               $mins - ($hours*60));

// Display a message box with only an OK button and the uptime text
MessageBoxA(NULL,
            $str,
            "Uptime Information",
            MB_OK);
?>
```

Table of Contents

w32api_deftype

(4.2.0 - 4.2.3 only)

w32api_deftype -- Defines a type for use with other w32api_functions

Description

bool **w32api_deftype** (string typename, string member1_type, string member1_name [, string ... [, string ...]])

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

If you would like to define a type for a w32api call, you need to call **w32api_deftype()**. This function takes 2n+1 arguments, where n is the number of members the type has. The first argument is the name of the type. After that is the type of the member followed by the members name (in pairs). A member type can be a user defined type. All the type names are case sensitive. Built in type names should be provided in lowercase. Returns **TRUE** on success or **FALSE** on failure.

w32api_init_dtype

(4.2.0 - 4.2.3 only)

w32api_init_dtype -- Creates an instance of the data type typename and fills it with the values passed

Description

resource **w32api_init_dtype** (string typename, mixed value [, mixed ...])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

This function creates an instance of the data type named *typename*, filling in the values of the data type. The *typename* parameter is case sensitive. You should give the values in the same order as you defined the data type with **w32api_deftype()**. The type of the resource returned is dynaparm.

w32api_invoke_function

(4.2.0 - 4.2.3 only)

w32api_invoke_function -- Invokes function funcname with the arguments passed after the function name

Description

mixed **w32api_invoke_function** (string funcname, mixed argument [, mixed ...])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

w32api_invoke_function() tries to find the previously registered function, named *funcname*, passing the parameters you provided. The return type is the one you set when you registered the function, the value is the one returned by the function itself. Any of the arguments can be of any PHP type or **w32api_deftype()** defined type, as needed.

w32api_register_function

(4.2.0 - 4.2.3 only)

w32api_register_function -- Registers function function_name from library with PHP

Description

bool **w32api_register_function** (string libary, string function_name, string return_type)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

This function tries to find the *function_name* function in *libary*, and tries to import it into PHP. The function will be registered with the given *return_type*. This type can be a generic PHP type, or a type defined with **w32api_deftype()**. All type names are

case sensitive. Built in type names should be provided in lowercase. Returns **TRUE** on success or **FALSE** on failure.

w32api_set_call_method

(4.2.0 - 4.2.3 only)

w32api_set_call_method -- Sets the calling method used

Description

void **w32api_set_call_method** (int method)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

This function sets the method call type. The parameter can be one of the constants **DC_CALL_CDECL** or **DC_CALL_STD**. The extension default is **DC_CALL_STD**.

CVI. WDDX Functions

Introduction

These functions are intended for work with WDDX.

Requirements

In order to use WDDX, you will need to install the expat library (which comes with Apache 1.3.7 or higher).

Installation

After installing expat compile PHP with `--enable-wddx`.

The windows version of PHP has built in support for this extension. You do not need to load any additional extension in order to use these functions.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Examples

All the functions that serialize variables use the first element of an array to determine whether the array is to be serialized into an array or structure. If the first element has string key, then it is serialized into a structure, otherwise, into an array.

Example 1. Serializing a single value

```
<?php
print wddx_serialize_value("PHP to WDDX packet example", "PHP packet");
?>
```

This example will produce:

```
<wddxPacket version='1.0'><header comment='PHP packet'/><data>
<string>PHP to WDDX packet example</string></data></wddxPacket>
```

Example 2. Using incremental packets

```
<?php
$pi = 3.1415926;
$packet_id = wddx_packet_start("PHP");
wddx_add_vars($packet_id, "pi");

/* Suppose $cities came from database */
$cities = array("Austin", "Novato", "Seattle");
wddx_add_vars($packet_id, "cities");

$packet = wddx_packet_end($packet_id);
print $packet;
?>
```

This example will produce:

```
<wddxPacket version='1.0'><header comment='PHP'/><data><struct>
<var name='pi'><number>3.1415926</number></var><var name='cities'>
<array length='3'><string>Austin</string><string>Novato</string>
<string>Seattle</string></array></var></struct></data></wddxPacket>
```

> **Note:** If you want to serialize non-ASCII characters you have to set the appropriate locale before doing so (see **setlocale()**).

Table of Contents

wddx_add_vars

(PHP 3>= 3.0.7, PHP 4)

wddx_add_vars -- Add variables to a WDDX packet with the specified ID

Description

bool **wddx_add_vars** (int packet_id, mixed name_var [, mixed ...])

wddx_add_vars() is used to serialize passed variables and add the result to the packet specified by the *packet_id*. The variables to be serialized are specified in exactly the same way as **wddx_serialize_vars()**.

wddx_deserialize

(PHP 3>= 3.0.7, PHP 4)

wddx_deserialize -- Deserializes a WDDX packet

Description

mixed **wddx_deserialize** (string packet)

wddx_deserialize() takes a *packet* string and deserializes it. It returns the result which can be string, number, or array. Note that structures are deserialized into associative arrays.

wddx_packet_end

(PHP 3>= 3.0.7, PHP 4)

wddx_packet_end -- Ends a WDDX packet with the specified ID

Description

string **wddx_packet_end** (int packet_id)

wddx_packet_end() ends the WDDX packet specified by the *packet_id* and returns the string with the packet.

wddx_packet_start

(PHP 3>= 3.0.7, PHP 4)

wddx_packet_start -- Starts a new WDDX packet with structure inside it

Description

int **wddx_packet_start** ([string comment])

Use **wddx_packet_start()** to start a new WDDX packet for incremental addition of variables. It takes an optional *comment* string and returns a packet ID for use in later functions. It automatically creates a structure definition inside the packet to contain the variables.

wddx_serialize_value

(PHP 3>= 3.0.7, PHP 4)

wddx_serialize_value -- Serialize a single value into a WDDX packet

Description

string **wddx_serialize_value** (mixed var [, string comment])

wddx_serialize_value() is used to create a WDDX packet from a single given value. It takes the value contained in *var*, and an optional *comment* string that appears in the packet header, and returns the WDDX packet.

wddx_serialize_vars

(PHP 3>= 3.0.7, PHP 4)

wddx_serialize_vars -- Serialize variables into a WDDX packet

Description

string **wddx_serialize_vars** (mixed var_name [, mixed ...])

wddx_serialize_vars() is used to create a WDDX packet with a structure that contains the serialized representation of the passed variables.

wddx_serialize_vars() takes a variable number of arguments, each of which can be either a string naming a variable or an array containing strings naming the variables or another array, etc.

Example 1. wddx_serialize_vars() example

```
<?php
$a = 1;
$b = 5.5;
$c = array("blue", "orange", "violet");
$d = "colors";

$clvars = array("c", "d");
print wddx_serialize_vars("a", "b", $clvars);
?>
```

The above example will produce:

```
<wddxPacket version='1.0'><header/><data><struct><var name='a'><number>1</number></var>
<var name='b'><number>5.5</number></var><var name='c'><array length='3'>
<string>blue</string><string>orange</string><string>violet</string></array></var>
<var name='d'><string>colors</string></var></struct></data></wddxPacket>
```

CVII. XML parser functions

Introduction

XML (eXtensible Markup Language) is a data format for structured document interchange on the Web. It is a standard defined by The World Wide Web consortium (W3C). Information about XML and related technologies can be found at http://www.w3.org/XML/.

This PHP extension implements support for James Clark's expat in PHP. This toolkit lets you parse, but not validate, XML documents. It supports three source character encodings also provided by PHP: `US-ASCII`, `ISO-8859-1` and `UTF-8`. `UTF-16` is not supported.

This extension lets you create XML parsers and then define *handlers* for different XML events. Each XML parser also has a few parameters you can adjust.

Requirements

This extension uses expat, which can be found at http://www.jclark.com/xml/. The Makefile that comes with expat does not build a library by default, you can use this make rule for that:

```
libexpat.a: $(OBJS)
    ar -rc $@ $(OBJS)
    ranlib $@
```

A source RPM package of expat can be found at http://sourceforge.net/projects/expat/.

Installation

These functions are enabled by default, using the bundled expat library. You can disable XML support with `--disable-xml`. If you compile PHP as a module for Apache 1.3.9 or later, PHP will automatically use the bundled expat library from Apache. In order you dont't want to use the bundled expat library configure PHP `--with-expat-dir=DIR`, where DIR should point to the base installation directory of expat.

The windows version of `PHP` has built in support for this extension. You do not need to load any additional extension in order to use these functions.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

XML_ERROR_NONE (integer)

XML_ERROR_NO_MEMORY (integer)

XML_ERROR_SYNTAX (integer)

XML_ERROR_NO_ELEMENTS (integer)

XML_ERROR_INVALID_TOKEN (integer)

XML_ERROR_UNCLOSED_TOKEN (integer)

XML_ERROR_PARTIAL_CHAR (integer)

XML_ERROR_TAG_MISMATCH (integer)

XML_ERROR_DUPLICATE_ATTRIBUTE (integer)

XML_ERROR_JUNK_AFTER_DOC_ELEMENT (integer)

XML_ERROR_PARAM_ENTITY_REF (integer)

XML_ERROR_UNDEFINED_ENTITY (integer)

XML_ERROR_RECURSIVE_ENTITY_REF (integer)

XML_ERROR_ASYNC_ENTITY (integer)

XML_ERROR_BAD_CHAR_REF (integer)

XML_ERROR_BINARY_ENTITY_REF (integer)

XML_ERROR_ATTRIBUTE_EXTERNAL_ENTITY_REF (integer)

XML_ERROR_MISPLACED_XML_PI (integer)

XML_ERROR_UNKNOWN_ENCODING (integer)

XML_ERROR_INCORRECT_ENCODING (integer)

XML_ERROR_UNCLOSED_CDATA_SECTION (integer)

XML_ERROR_EXTERNAL_ENTITY_HANDLING (integer)

XML_OPTION_CASE_FOLDING (integer)

XML_OPTION_TARGET_ENCODING (integer)

XML_OPTION_SKIP_TAGSTART (integer)

XML_OPTION_SKIP_WHITE (integer)

Event Handlers

The XML event handlers defined are:

Table 1. Supported XML handlers

PHP function to set handler	Event description
xml_set_element_handler()	Element events are issued whenever the XML parser encounters start or end tags. There are separate handlers for start tags and end tags.
xml_set_character_data_handler()	Character data is roughly all the non-markup contents of XML documents, including whitespace between tags. Note that the XML parser does not add or remove any whitespace, it is up to the application (you) to decide whether whitespace is significant.
xml_set_processing_instruction_handler()	PHP programmers should be familiar with processing instructions (PIs) already. <?php ?> is a processing instruction, where *php* is called the "PI target". The handling of these are application-specific, except that all PI targets starting with "XML" are reserved.
xml_set_default_handler()	What goes not to another handler goes to the default handler. You will get things like the XML and document type declarations in the default handler.
xml_set_unparsed_entity_decl_handler()	This handler will be called for declaration of an unparsed (NDATA) entity.
xml_set_notation_decl_handler()	This handler is called for declaration of a notation.
xml_set_external_entity_ref_handler()	This handler is called when the XML parser finds a reference to an external parsed general entity. This can be a reference to a file or URL, for example. See the external entity example for a demonstration.

Case Folding

The element handler functions may get their element names *case-folded*. Case-folding is defined by the XML standard as "a process applied to a sequence of characters, in which those identified as non-uppercase are replaced by their uppercase equivalents". In other words, when it comes to XML, case-folding simply means uppercasing.

By default, all the element names that are passed to the handler functions are case-folded. This behaviour can be queried and controlled per XML parser with the **xml_parser_get_option()** and **xml_parser_set_option()** functions, respectively.

Error Codes

The following constants are defined for XML error codes (as returned by **xml_parse()**):

XML_ERROR_NONE
XML_ERROR_NO_MEMORY
XML_ERROR_SYNTAX
XML_ERROR_NO_ELEMENTS
XML_ERROR_INVALID_TOKEN
XML_ERROR_UNCLOSED_TOKEN
XML_ERROR_PARTIAL_CHAR
XML_ERROR_TAG_MISMATCH
XML_ERROR_DUPLICATE_ATTRIBUTE
XML_ERROR_JUNK_AFTER_DOC_ELEMENT
XML_ERROR_PARAM_ENTITY_REF
XML_ERROR_UNDEFINED_ENTITY
XML_ERROR_RECURSIVE_ENTITY_REF
XML_ERROR_ASYNC_ENTITY
XML_ERROR_BAD_CHAR_REF
XML_ERROR_BINARY_ENTITY_REF
XML_ERROR_ATTRIBUTE_EXTERNAL_ENTITY_REF
XML_ERROR_MISPLACED_XML_PI
XML_ERROR_UNKNOWN_ENCODING
XML_ERROR_INCORRECT_ENCODING
XML_ERROR_UNCLOSED_CDATA_SECTION
XML_ERROR_EXTERNAL_ENTITY_HANDLING

Character Encoding

PHP's XML extension supports the Unicode character set through different *character encodings*. There are two types of character encodings, *source encoding* and *target encoding*. PHP's internal representation of the document is always encoded with `UTF-8`.

Source encoding is done when an XML document is parsed. Upon creating an XML parser, a source encoding can be specified (this encoding can not be changed later in the XML parser's lifetime). The supported source encodings are `ISO-8859-1`, `US-ASCII` and `UTF-8`. The former two are single-byte encodings, which means that each character is represented by a single byte. `UTF-8` can encode characters composed by a variable number of bits (up to 21) in one to four bytes. The default source encoding used by PHP is `ISO-8859-1`.

Target encoding is done when PHP passes data to XML handler functions. When an XML parser is created, the target encoding is set to the same as the source encoding, but this may be changed at any point. The target encoding will affect character data as well as tag names and processing instruction targets.

If the XML parser encounters characters outside the range that its source encoding is capable of representing, it will return an error.

If PHP encounters characters in the parsed XML document that can not be represented in the chosen target encoding, the problem characters will be "demoted". Currently, this means that such characters are replaced by a question mark.

Examples

Here are some example PHP scripts parsing XML documents.

XML Element Structure Example

This first example displays the stucture of the start elements in a document with indentation.

Example 1. Show XML Element Structure

```
$file = "data.xml";
$depth = array();

function startElement($parser, $name, $attrs) {
    global $depth;
    for ($i = 0; $i < $depth[$parser]; $i++) {
        print "  ";
    }
    print "$name\n";
    $depth[$parser]++;
}

function endElement($parser, $name) {
    global $depth;
    $depth[$parser]--;
}

$xml_parser = xml_parser_create();
xml_set_element_handler($xml_parser, "startElement", "endElement");
if (!($fp = fopen($file, "r"))) {
    die("could not open XML input");
}

while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML error: %s at line %d",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
xml_parser_free($xml_parser);
```

XML Tag Mapping Example

Example 2. Map XML to HTML

This example maps tags in an XML document directly to HTML tags. Elements not found in the "map array" are ignored. Of course, this example will only work with a specific XML document type.

```
$file = "data.xml";
$map_array = array(
    "BOLD"     => "B",
    "EMPHASIS" => "I",
    "LITERAL"  => "TT"
);

function startElement($parser, $name, $attrs) {
    global $map_array;
    if ($htmltag = $map_array[$name]) {
        print "<$htmltag>";
    }
}

function endElement($parser, $name) {
    global $map_array;
    if ($htmltag = $map_array[$name]) {
        print "</$htmltag>";
    }
}

function characterData($parser, $data) {
    print $data;
}

$xml_parser = xml_parser_create();
// use case-folding so we are sure to find the tag in $map_array
xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, true);
xml_set_element_handler($xml_parser, "startElement", "endElement");
xml_set_character_data_handler($xml_parser, "characterData");
if (!($fp = fopen($file, "r"))) {
    die("could not open XML input");
}

while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML error: %s at line %d",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
xml_parser_free($xml_parser);
```

XML External Entity Example

This example highlights XML code. It illustrates how to use an external entity reference handler to include and parse other documents, as well as how PIs can be processed, and a way of determining "trust" for PIs containing code.

XML documents that can be used for this example are found below the example (`xmltest.xml` and `xmltest2.xml`.)

Example 3. External Entity Example

```
<?php
$file = "xmltest.xml";

function trustedFile($file) {
    // only trust local files owned by ourselves
    if (!eregi("^([a-z]+)://", $file)
        && fileowner($file) == getmyuid()) {
            return true;
    }
    return false;
}

function startElement($parser, $name, $attribs) {
    print "&lt;<font color=\"#0000cc\">$name</font>";
    if (sizeof($attribs)) {
        while (list($k, $v) = each($attribs)) {
            print " <font color=\"#009900\">$k</font>=\"<font
                  color=\"#990000\">$v</font>\"";
        }
    }
    print "&gt;";
}

function endElement($parser, $name) {
    print "&lt;/<font color=\"#0000cc\">$name</font>&gt;";
}

function characterData($parser, $data) {
    print "<b>$data</b>";
}

function PIHandler($parser, $target, $data) {
```

```
    switch (strtolower($target)) {
        case "php":
            global $parser_file;
            // If the parsed document is "trusted", we say it is safe
            // to execute PHP code inside it.  If not, display the code
            // instead.
            if (trustedFile($parser_file[$parser])) {
                eval($data);
            } else {
                printf("Untrusted PHP code: <i>%s</i>",
                       htmlspecialchars($data));
            }
            break;
    }
}

function defaultHandler($parser, $data) {
    if (substr($data, 0, 1) == "&" && substr($data, -1, 1) == ";") {
        printf('<font color="#aa00aa">%s</font>',
                htmlspecialchars($data));
    } else {
        printf('<font size="-1">%s</font>',
                htmlspecialchars($data));
    }
}

function externalEntityRefHandler($parser, $openEntityNames, $base, $systemId,
                                   $publicId) {
    if ($systemId) {
        if (!list($parser, $fp) = new_xml_parser($systemId)) {
            printf("Could not open entity %s at %s\n", $openEntityNames,
                   $systemId);
            return false;
        }
        while ($data = fread($fp, 4096)) {
            if (!xml_parse($parser, $data, feof($fp))) {
                printf("XML error: %s at line %d while parsing entity %s\n",
                       xml_error_string(xml_get_error_code($parser)),
                       xml_get_current_line_number($parser), $openEntityNames);
                xml_parser_free($parser);
                return false;
            }
        }
        xml_parser_free($parser);
        return true;
    }
    return false;
}

function new_xml_parser($file) {
    global $parser_file;

    $xml_parser = xml_parser_create();
    xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, 1);
    xml_set_element_handler($xml_parser, "startElement", "endElement");
    xml_set_character_data_handler($xml_parser, "characterData");
    xml_set_processing_instruction_handler($xml_parser, "PIHandler");
    xml_set_default_handler($xml_parser, "defaultHandler");
    xml_set_external_entity_ref_handler($xml_parser, "externalEntityRefHandler");

    if (!($fp = @fopen($file, "r"))) {
        return false;
    }
    if (!is_array($parser_file)) {
        settype($parser_file, "array");
    }
    $parser_file[$xml_parser] = $file;
    return array($xml_parser, $fp);
}

if (!(list($xml_parser, $fp) = new_xml_parser($file))) {
    die("could not open XML input");
}

print "<pre>";
while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML error: %s at line %d\n",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
print "</pre>";
print "parse complete\n";
xml_parser_free($xml_parser);

?>
```

Example 4. xmltest.xml

```
<?xml version='1.0'?>
<!DOCTYPE chapter SYSTEM "/just/a/test.dtd" [
<!ENTITY plainEntity "FOO entity">
<!ENTITY systemEntity SYSTEM "xmltest2.xml">
```

```
]>
<chapter>
 <TITLE>Title &plainEntity;</TITLE>
 <para>
  <informaltable>
   <tgroup cols="3">
    <tbody>
     <row><entry>a1</entry><entry morerows="1">b1</entry><entry>c1</entry></row>
     <row><entry>a2</entry><entry>c2</entry></row>
     <row><entry>a3</entry><entry>b3</entry><entry>c3</entry></row>
    </tbody>
   </tgroup>
  </informaltable>
 </para>
 &systemEntity;
 <section id="about">
  <title>About this Document</title>
  <para>
   <!-- this is a comment -->
   <?php print 'Hi!  This is PHP version '.phpversion(); ?>
  </para>
 </section>
</chapter>
```

This file is included from `xmltest.xml`:

Example 5. xmltest2.xml

```
<?xml version="1.0"?>
<!DOCTYPE foo [
<!ENTITY testEnt "test entity">
]>
<foo>
   <element attrib="value"/>
   &testEnt;
   <?php print "This is some more PHP code being executed."; ?>
</foo>
```

Table of Contents

utf8_decode

(PHP 3>= 3.0.6, PHP 4)

utf8_decode -- Converts a string with ISO-8859-1 characters encoded with UTF-8 to single-byte ISO-8859-1.

Description

string **utf8_decode** (string data)

This function decodes *data*, assumed to be `UTF-8` encoded, to `ISO-8859-1`.

See **utf8_encode()** for an explaination of UTF-8 encoding.

utf8_encode

(PHP 3>= 3.0.6, PHP 4)

utf8_encode -- encodes an ISO-8859-1 string to UTF-8

Description

string **utf8_encode** (string data)

This function encodes the string *data* to UTF-8, and returns the encoded version. UTF-8 is a standard mechanism used by Unicode for encoding *wide character* values into a byte stream. UTF-8 is transparent to plain ASCII characters, is self-synchronized (meaning it is possible for a program to figure out where in the bytestream characters start) and can be used with normal string comparison functions for sorting and such. PHP encodes UTF-8 characters in up to four bytes, like this:

Table 1. UTF-8 encoding

bytes	bits	representation
1	7	0bbbbbbb
2	11	110bbbbb 10bbbbbb
3	16	1110bbbb 10bbbbbb 10bbbbbb
4	21	11110bbb 10bbbbbb 10bbbbbb 10bbbbbb

Each *b* represents a bit that can be used to store character data.

xml_error_string

(PHP 3>= 3.0.6, PHP 4)

xml_error_string -- get XML parser error string

Description

string **xml_error_string** (int code)

code

An error code from **xml_get_error_code()**.

Returns a string with a textual description of the error code *code*, or **FALSE** if no description was found.

xml_get_current_byte_index

(PHP 3>= 3.0.6, PHP 4)

xml_get_current_byte_index -- get current byte index for an XML parser

Description

int **xml_get_current_byte_index** (resource parser)

parser

A reference to the XML parser to get byte index from.

This function returns **FALSE** if *parser* does not refer to a valid parser, or else it returns which byte index the parser is currently at in its data buffer (starting at 0).

xml_get_current_column_number

(PHP 3>= 3.0.6, PHP 4)

xml_get_current_column_number -- Get current column number for an XML parser

Description

int **xml_get_current_column_number** (resource parser)

parser

A reference to the XML parser to get column number from.

This function returns **FALSE** if *parser* does not refer to a valid parser, or else it returns which column on the current line (as given by **xml_get_current_line_number()**) the parser is currently at.

xml_get_current_line_number

(PHP 3>= 3.0.6, PHP 4)

xml_get_current_line_number -- get current line number for an XML parser

Description

int **xml_get_current_line_number** (resource parser)

parser

A reference to the XML parser to get line number from.

This function returns **FALSE** if *parser* does not refer to a valid parser, or else it returns which line the parser is currently at in its data buffer.

xml_get_error_code

(PHP 3>= 3.0.6, PHP 4)

xml_get_error_code -- get XML parser error code

Description

int **xml_get_error_code** (resource parser)

parser

A reference to the XML parser to get error code from.

This function returns **FALSE** if *parser* does not refer to a valid parser, or else it returns one of the error codes listed in the error codes section.

xml_parse_into_struct

(PHP 3>= 3.0.8, PHP 4)

xml_parse_into_struct -- Parse XML data into an array structure

Description

int **xml_parse_into_struct** (resource parser, string data, array &values [, array &index])

This function parses an XML file into 2 parallel array structures, one (*index*) containing pointers to the location of the appropriate values in the *values* array. These last two parameters must be passed by reference.

Below is an example that illustrates the internal structure of the arrays being generated by the function. We use a simple `note` tag embeded inside a `para` tag, and then we parse this an print out the structures generated:

```
$simple = "<para><note>simple note</note></para>";
$p = xml_parser_create();
xml_parse_into_struct($p,$simple,$vals,$index);
xml_parser_free($p);
echo "Index array\n";
print_r($index);
echo "\nVals array\n";
print_r($vals);
```

When we run that code, the output will be:

```
Index array
Array
(
    [PARA] => Array
        (
            [0] => 0
            [1] => 2
        )

    [NOTE] => Array
        (
            [0] => 1
        )

)

Vals array
Array
(
    [0] => Array
        (
            [tag] => PARA
            [type] => open
            [level] => 1
        )

    [1] => Array
        (
            [tag] => NOTE
            [type] => complete
            [level] => 2
            [value] => simple note
        )

    [2] => Array
        (
            [tag] => PARA
            [type] => close
            [level] => 1
        )

)
```

Event-driven parsing (based on the expat library) can get complicated when you have an XML document that is complex. This function does not produce a DOM style object, but it generates structures amenable of being transversed in a tree fashion. Thus, we can create objects representing the data in the XML file easily. Let's consider the following XML file representing a small database of aminoacids information:

Example 1. moldb.xml - small database of molecular information

```
<?xml version="1.0"?>
<moldb>

    <molecule>
        <name>Alanine</name>
        <symbol>ala</symbol>
        <code>A</code>
        <type>hydrophobic</type>
    </molecule>

    <molecule>
        <name>Lysine</name>
        <symbol>lys</symbol>
        <code>K</code>
        <type>charged</type>
    </molecule>

</moldb>
```

And some code to parse the document and generate the appropriate objects:

Example 2. parsemoldb.php - parses moldb.xml into and array of molecular objects

```
<?php
```

```
class AminoAcid {
    var $name;  // aa name
    var $symbol;    // three letter symbol
    var $code;  // one letter code
    var $type;  // hydrophobic, charged or neutral

    function AminoAcid ($aa) {
        foreach ($aa as $k=>$v)
            $this->$k = $aa[$k];
    }
}

function readDatabase($filename) {
    // read the xml database of aminoacids
    $data = implode("",file($filename));
    $parser = xml_parser_create();
    xml_parser_set_option($parser,XML_OPTION_CASE_FOLDING,0);
    xml_parser_set_option($parser,XML_OPTION_SKIP_WHITE,1);
    xml_parse_into_struct($parser,$data,$values,$tags);
    xml_parser_free($parser);

    // loop through the structures
    foreach ($tags as $key=>$val) {
        if ($key == "molecule") {
            $molranges = $val;
            // each contiguous pair of array entries are the
            // lower and upper range for each molecule definition
            for ($i=0; $i < count($molranges); $i+=2) {
                    $offset = $molranges[$i] + 1;
                $len = $molranges[$i + 1] - $offset;
                $tdb[] = parseMol(array_slice($values, $offset, $len));
            }
        } else {
            continue;
        }
    }
    return $tdb;
}

function parseMol($mvalues) {
    for ($i=0; $i < count($mvalues); $i++)
        $mol[$mvalues[$i]["tag"]] = $mvalues[$i]["value"];
    return new AminoAcid($mol);
}

$db = readDatabase("moldb.xml");
echo "** Database of AminoAcid objects:\n";
print_r($db);

?>
```

After executing `parsemoldb.php`, the variable `$db` contains an array of **AminoAcid** objects, and the output of the script confirms that:

```
** Database of AminoAcid objects:
Array
(
    [0] => aminoacid Object
        (
            [name] => Alanine
            [symbol] => ala
            [code] => A
            [type] => hydrophobic
        )

    [1] => aminoacid Object
        (
            [name] => Lysine
            [symbol] => lys
            [code] => K
            [type] => charged
        )

)
```

xml_parse

(PHP 3>= 3.0.6, PHP 4)

xml_parse -- start parsing an XML document

Description

bool **xml_parse** (resource parser, string data [, bool is_final])

parser

A reference to the XML parser to use.

data

Chunk of data to parse. A document may be parsed piece-wise by calling **xml_parse()** several times with new data, as long as the *is_final* parameter is set and **TRUE** when the last data is parsed.

is_final (optional)

If set and **TRUE**, *data* is the last piece of data sent in this parse.

When the XML document is parsed, the handlers for the configured events are called as many times as necessary, after which this function returns **TRUE** or **FALSE**.

TRUE is returned if the parse was successful, **FALSE** if it was not successful, or if *parser* does not refer to a valid parser. For unsuccessful parses, error information can be retrieved with **xml_get_error_code()**, **xml_error_string()**, **xml_get_current_line_number()**, **xml_get_current_column_number()** and **xml_get_current_byte_index()**.

xml_parser_create_ns

(PHP 4 >= 4.0.5)

xml_parser_create_ns -- Create an XML parser

Description

resource **xml_parser_create_ns** ([string encoding [, string sep]])

Warning
This function is currently not documented; only the argument list is available.

xml_parser_create

(PHP 3>= 3.0.6, PHP 4)

xml_parser_create -- create an XML parser

Description

resource **xml_parser_create** ([string encoding])

encoding (optional)

Which character encoding the parser should use. The following character encodings are supported:

`ISO-8859-1` (default)
`US-ASCII`
`UTF-8`

This function creates an XML parser and returns a handle for use by other XML functions. Returns **FALSE** on failure.

xml_parser_free

(PHP 3>= 3.0.6, PHP 4)

xml_parser_free -- Free an XML parser

Description

bool **xml_parser_free** (resource parser)

parser

A reference to the XML parser to free.

This function returns **FALSE** if *parser* does not refer to a valid parser, or else it frees the parser and returns **TRUE**.

xml_parser_get_option

(PHP 3>= 3.0.6, PHP 4)

xml_parser_get_option -- get options from an XML parser

Description

mixed **xml_parser_get_option** (resource parser, int option)

parser

A reference to the XML parser to get an option from.

option

Which option to fetch. See **xml_parser_set_option()** for a list of options.

This function returns **FALSE** if *parser* does not refer to a valid parser, or if the option could not be set. Else the option's value is returned.

See **xml_parser_set_option()** for the list of options.

xml_parser_set_option

(PHP 3>= 3.0.6, PHP 4)

xml_parser_set_option -- set options in an XML parser

Description

bool **xml_parser_set_option** (resource parser, int option, mixed value)

parser

A reference to the XML parser to set an option in.

option

Which option to set. See below.

value

The option's new value.

This function returns **FALSE** if *parser* does not refer to a valid parser, or if the option could not be set. Else the option is set and **TRUE** is returned.

The following options are available:

Table 1. XML parser options

Option constant	Data type	Description
XML_OPTION_CASE_FOLDING	integer	Controls whether case-folding is enabled for this XML parser. Enabled by default.
XML_OPTION_TARGET_ENCODING	string	Sets which target encoding to use in this XML parser. By default, it is set to the same as the source encoding used by **xml_parser_create()**. Supported target encodings are `ISO-8859-1`, `US-ASCII` and `UTF-8`.

xml_set_character_data_handler

(PHP 3>= 3.0.6, PHP 4)

xml_set_character_data_handler -- set up character data handler

Description

bool **xml_set_character_data_handler** (resource parser, string handler)

Sets the character data handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when **xml_parse()** is called for *parser*.

The function named by *handler* must accept two parameters: ***handler*** (resource parser, string data)

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

data

The second parameter, *data*, contains the character data as a string.

If a handler function is set to an empty string, or **FALSE**, the handler in question is disabled.

TRUE is returned if the handler is set up, **FALSE** if *parser* is not a parser.

Note: Instead of a function name, an array containing an object reference and a method name can also be supplied.

xml_set_default_handler

(PHP 3>= 3.0.6, PHP 4)

xml_set_default_handler -- set up default handler

Description

bool **xml_set_default_handler** (resource parser, string handler)

Sets the default handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when **xml_parse()** is called for *parser*.

The function named by *handler* must accept two parameters: ***handler*** (resource parser, string data)

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

data

The second parameter, *data*, contains the character data. This may be the XML declaration, document type declaration, entities or other data for which no other handler exists.

If a handler function is set to an empty string, or **FALSE**, the handler in question is disabled.

TRUE is returned if the handler is set up, **FALSE** if *parser* is not a parser.

Note: Instead of a function name, an array containing an object reference and a method name can also be supplied.

xml_set_element_handler

(PHP 3>= 3.0.6, PHP 4)

xml_set_element_handler -- set up start and end element handlers

Description

bool **xml_set_element_handler** (resource parser, string start_element_handler, string end_element_handler)

Sets the element handler functions for the XML parser *parser*. *start_element_handler* and *end_element_handler* are strings containing the names of functions that must exist when **xml_parse()** is called for *parser*.

The function named by *start_element_handler* must accept three parameters: ***start_element_handler*** (resource parser, string name, array attribs)

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

name

The second parameter, *name*, contains the name of the element for which this handler is called. If case-folding is in effect for this parser, the element name will be in uppercase letters.

attribs

The third parameter, *attribs*, contains an associative array with the element's attributes (if any). The keys of this array are the attribute names, the values are the attribute values. Attribute names are case-folded on the same criteria as element names. Attribute values are *not* case-folded.

The original order of the attributes can be retrieved by walking through *attribs* the normal way, using **each()**. The first key in the array was the first attribute, and so on.

The function named by *end_element_handler* must accept two parameters: ***end_element_handler*** (resource parser, string name)

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

name

The second parameter, *name*, contains the name of the element for which this handler is called. If case-folding is in effect for this parser, the element name will be in uppercase letters.

If a handler function is set to an empty string, or **FALSE**, the handler in question is disabled.

TRUE is returned if the handlers are set up, **FALSE** if *parser* is not a parser.

Note: Instead of a function name, an array containing an object reference and a method name can also be supplied.

xml_set_end_namespace_decl_handler

(PHP 4 >= 4.0.5)

xml_set_end_namespace_decl_handler -- Set up character data handler

Description

bool **xml_set_end_namespace_decl_handler** (resource pind, string hdl)

Warning
This function is currently not documented; only the argument list is available.

Note: Instead of a function name, an array containing an object reference and a method name can also be supplied.

xml_set_external_entity_ref_handler

(PHP 3>= 3.0.6, PHP 4)

xml_set_external_entity_ref_handler -- set up external entity reference handler

Description

bool **xml_set_external_entity_ref_handler** (resource parser, string handler)

Sets the external entity reference handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when **xml_parse()** is called for *parser*.

The function named by *handler* must accept five parameters, and should return an integer value. If the value returned from the handler is **FALSE** (which it will be if no value is returned), the XML parser will stop parsing and **xml_get_error_code()** will return XML_ERROR_EXTERNAL_ENTITY_HANDLING. ***handler*** (resource parser, string open_entity_names, string base, string system_id, string public_id)

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

open_entity_names

The second parameter, *open_entity_names*, is a space-separated list of the names of the entities that are open for the parse of this entity (including the name of the referenced entity).

base

This is the base for resolving the system identifier (*system_id*) of the external entity. Currently this parameter will always be set to an empty string.

system_id

The fourth parameter, *system_id*, is the system identifier as specified in the entity declaration.

public_id

The fifth parameter, *public_id*, is the public identifier as specified in the entity declaration, or an empty string if none was specified; the whitespace in the public identifier will have been normalized as required by the XML spec.

If a handler function is set to an empty string, or **FALSE**, the handler in question is disabled.

TRUE is returned if the handler is set up, **FALSE** if *parser* is not a parser.

Note: Instead of a function name, an array containing an object reference and a method name can also be supplied.

xml_set_notation_decl_handler

(PHP 3>= 3.0.6, PHP 4)

xml_set_notation_decl_handler -- set up notation declaration handler

Description

bool **xml_set_notation_decl_handler** (resource parser, string handler)

Sets the notation declaration handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when **xml_parse()** is called for *parser*.

A notation declaration is part of the document's DTD and has the following format:

```
<!NOTATION
      name {system_id |
      public_id}>
```

See section 4.7 of the XML 1.0 spec for the definition of notation declarations.

The function named by *handler* must accept five parameters: ***handler*** (resource parser, string notation_name, string base, string system_id, string public_id)

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

notation_name

This is the notation's *name*, as per the notation format described above.

base

This is the base for resolving the system identifier (*system_id*) of the notation declaration. Currently this parameter will always be set to an empty string.

system_id

System identifier of the external notation declaration.

public_id

Public identifier of the external notation declaration.

If a handler function is set to an empty string, or **FALSE**, the handler in question is disabled.

TRUE is returned if the handler is set up, **FALSE** if *parser* is not a parser.

Note: Instead of a function name, an array containing an object reference and a method name can also be supplied.

xml_set_object

(PHP 4)

xml_set_object -- Use XML Parser within an object

Description

void **xml_set_object** (resource parser, object object)

This function allows to use *parser* inside *object*. All callback functions could be set with **xml_set_element_handler()** etc and assumed to be methods of *object*.

```
<?php
class xml  {
    var $parser;

    function xml()
    {
        $this->parser = xml_parser_create();

        xml_set_object($this->parser, &$this);
        xml_set_element_handler($this->parser, "tag_open", "tag_close");
        xml_set_character_data_handler($this->parser, "cdata");
    }

    function parse($data)
    {
        xml_parse($this->parser, $data);
    }

    function tag_open($parser, $tag, $attributes)
    {
        var_dump($parser, $tag, $attributes);
    }

    function cdata($parser, $cdata)
    {
        var_dump($parser, $cdata);
    }

    function tag_close($parser, $tag)
    {
        var_dump($parser, $tag);
    }

} // end of class xml

$xml_parser = new xml();
$xml_parser->parse("<A ID='hallo'>PHP</A>");
?>
```

xml_set_processing_instruction_handler

(PHP 3>= 3.0.6, PHP 4)

xml_set_processing_instruction_handler -- Set up processing instruction (PI) handler

Description

bool **xml_set_processing_instruction_handler** (resource parser, string handler)

Sets the processing instruction (PI) handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when **xml_parse()** is called for *parser*.

A processing instruction has the following format:

```
<?
        target
        data?>
```

You can put PHP code into such a tag, but be aware of one limitation: in an XML PI, the PI end tag (?>) can not be quoted, so this character sequence should not appear in the PHP code you embed with PIs in XML documents. If it does, the rest of the PHP code, as well as the "real" PI end tag, will be treated as character data.

The function named by *handler* must accept three parameters: ***handler*** (resource parser, string target, string data)

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

target

The second parameter, *target*, contains the PI target.

data

The third parameter, *data*, contains the PI data.

If a handler function is set to an empty string, or **FALSE**, the handler in question is disabled.

TRUE is returned if the handler is set up, **FALSE** if *parser* is not a parser.

Note: Instead of a function name, an array containing an object reference and a method name can also be supplied.

xml_set_start_namespace_decl_handler

(PHP 4 >= 4.0.5)

xml_set_start_namespace_decl_handler -- Set up character data handler

Description

bool **xml_set_start_namespace_decl_handler** (resource pind, string hdl)

Warning
This function is currently not documented; only the argument list is available.

Note: Instead of a function name, an array containing an object reference and a method name can also be supplied.

xml_set_unparsed_entity_decl_handler

(PHP 3>= 3.0.6, PHP 4)

xml_set_unparsed_entity_decl_handler -- Set up unparsed entity declaration handler

Description

bool **xml_set_unparsed_entity_decl_handler** (resource parser, string handler)

Sets the unparsed entity declaration handler function for the XML parser *parser*. *handler* is a string containing the name of a function that must exist when **xml_parse()** is called for *parser*.

This handler will be called if the XML parser encounters an external entity declaration with an NDATA declaration, like the following:

```
<!ENTITY name {publicId | systemId}
        NDATA notationName>
```

See section 4.2.2 of the XML 1.0 spec for the definition of notation declared external entities.

The function named by *handler* must accept six parameters: ***handler*** (resource parser, string entity_name, string base, string system_id, string public_id, string notation_name)

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

entity_name

The name of the entity that is about to be defined.

base

This is the base for resolving the system identifier (*systemId*) of the external entity. Currently this parameter will always be set to an empty string.

system_id

System identifier for the external entity.

public_id

Public identifier for the external entity.

notation_name

Name of the notation of this entity (see **xml_set_notation_decl_handler()**).

If a handler function is set to an empty string, or **FALSE**, the handler in question is disabled.

TRUE is returned if the handler is set up, **FALSE** if *parser* is not a parser.

Note: Instead of a function name, an array containing an object reference and a method name can also be supplied.

CVIII. XML-RPC functions

Introduction

These functions can be used to write XML-RPC servers and clients. You can find more information about XML-RPC at http://www.xmlrpc.com/, and more documentation on this extension and it's functions at http://xmlrpc-epi.sourceforge.net/.

Warning
This extension is *EXPERIMENTAL*. The behaviour of this extension -- including the names of its functions and anything else documented about this extension -- may change without notice in a future release of PHP. Use this extension at your own risk.

Requirements

No external libraries are needed to build this extension.

Installation

XML-RPC support in PHP is not enabled by default. You will need to use the `--with-xmlrpc[=DIR]` configuration option when compiling PHP to enable XML-RPC support. This extension is bundled into PHP as of 4.1.0.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. XML-RPC configuration options

Name	Default	Changeable
xmlrpc_errors	"0"	PHP_INI_SYSTEM
xmlrpc_error_number	"0"	PHP_INI_ALL

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Table of Contents

xmlrpc_decode_request

(PHP 4 >= 4.1.0)

xmlrpc_decode_request -- Decodes XML into native PHP types

Description

array **xmlrpc_decode_request** (string xml, string method [, string encoding])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

xmlrpc_decode

(PHP 4 >= 4.1.0)

xmlrpc_decode -- Decodes XML into native PHP types

Description

array **xmlrpc_decode** (string xml [, string encoding])

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

xmlrpc_encode_request

(PHP 4 >= 4.1.0)

xmlrpc_encode_request -- Generates XML for a method request

Description

string **xmlrpc_encode_request** (string method, mixed params)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

xmlrpc_encode

(PHP 4 >= 4.1.0)

xmlrpc_encode -- Generates XML for a PHP value

Description

string **xmlrpc_encode** (mixed value)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

xmlrpc_get_type

(PHP 4 >= 4.1.0)

xmlrpc_get_type -- Gets xmlrpc type for a PHP value. Especially useful for base64 and datetime strings

Description

string **xmlrpc_get_type** (mixed value)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

xmlrpc_parse_method_descriptions

(PHP 4 >= 4.1.0)

xmlrpc_parse_method_descriptions -- Decodes XML into a list of method descriptions

Description

array **xmlrpc_parse_method_descriptions** (string xml)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

xmlrpc_server_add_introspection_data

(PHP 4 >= 4.1.0)

xmlrpc_server_add_introspection_data -- Adds introspection documentation

Description

int **xmlrpc_server_add_introspection_data** (resource server, array desc)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

xmlrpc_server_call_method

(PHP 4 >= 4.1.0)

xmlrpc_server_call_method -- Parses XML requests and call methods

Description

mixed **xmlrpc_server_call_method** (resource server, string xml, mixed user_data [, array output_options])

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

xmlrpc_server_create

(PHP 4 >= 4.1.0)

xmlrpc_server_create -- Creates an xmlrpc server

Description

resource **xmlrpc_server_create** (void)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

xmlrpc_server_destroy

(PHP 4 >= 4.1.0)

xmlrpc_server_destroy -- Destroys server resources

Description

void **xmlrpc_server_destroy** (resource server)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

xmlrpc_server_register_introspection_callback

(PHP 4 >= 4.1.0)

xmlrpc_server_register_introspection_callback -- Register a PHP function to generate documentation

Description

bool **xmlrpc_server_register_introspection_callback** (resource server, string function)

Warning

This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning

This function is currently not documented; only the argument list is available.

xmlrpc_server_register_method

(PHP 4 >= 4.1.0)

xmlrpc_server_register_method -- Register a PHP function to resource method matching method_name

Description

bool **xmlrpc_server_register_method** (resource server, string method_name, string function)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

xmlrpc_set_type

(PHP 4 >= 4.1.0)

xmlrpc_set_type -- Sets xmlrpc type, base64 or datetime, for a PHP string value

Description

bool **xmlrpc_set_type** (string value, string type)

Warning
This function is *EXPERIMENTAL*. The behaviour of this function, the name of this function, and anything else documented about this function may change without notice in a future release of PHP. Use this function at your own risk.

Warning
This function is currently not documented; only the argument list is available.

CIX. XSLT functions

Introduction

This PHP extension provides a processor independent API to XSLT transformations. Currently this extension only supports the Sablotron library from the Ginger Alliance. Support is planned for other libraries, such as the Xalan library or the libxslt library.

XSLT (Extensible Stylesheet Language (XSL) Transformations) is a language for transforming XML documents into other XML documents. It is a standard defined by The World Wide Web Consortium (W3C). Information about XSLT and related technologies can be found at http://www.w3.org/TR/xslt.

> **Note:** This extension is different than the sablotron extension distributed with versions of PHP prior to PHP 4.1, currently only the new XSLT extension in PHP 4.1 is supported. If you need support for the old extension, please ask your questions on the PHP mailing lists.

Requirements

This extension uses Sablotron and expat, which can both be found at http://www.gingerall.com/. Binaries are provided as well as source.

Installation

On UNIX, run **configure** with the `--enable-xslt --with-xslt-sablot` options. The Sablotron library should be installed somewhere your compiler can find it.

Make sure you have the same libraries linked to the Sablotron library as those, which are linked with PHP. The configuration options: `--with-expat-dir=DIR --with-iconv-dir=DIR` are there to help you specify them. When asking for support, always mention these directives, and whether there are other versions of those libraries installed on your system somewhere. Naturally, provide all the version numbers.

> **JavaScript E-XSLT support:** If you compiled Sablotron with JavaScript support, you must specify the option: `--with-sablot-js=DIR`.

> **Note to Win32 Users:** In order to enable this module on a Windows environment, you must copy *sablot.dll* from the DLL folder of the PHP/Win32 binary package to the SYSTEM32 folder of your windows machine. (Ex: C:\WINNT\SYSTEM32 or C:\WINDOWS\SYSTEM32)

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

`XSLT_OPT_SILENT` (integer)

Drop all logging and error reporting. This is a generic option for all backends that may be added in the future.

`XSLT_SABOPT_PARSE_PUBLIC_ENTITIES` (integer)

Tell Sablotron to parse public entities. By default this has been turned off.

`XSLT_SABOPT_DISABLE_ADDING_META` (integer)

Do not add the meta tag "Content-Type" for HTML output. The default is set during compilation of Sablotron.

`XSLT_SABOPT_DISABLE_STRIPPING` (integer)

Suppress the whitespace stripping (on data files only).

`XSLT_SABOPT_IGNORE_DOC_NOT_FOUND` (integer)

Consider unresolved documents (the document() function) non-lethal.

`XSLT_ERR_UNSUPPORTED_SCHEME` (integer)

Error return code, for scheme handlers.

Table of Contents

xslt_set_error_handler -- Set an error handler for a XSLT processor
xslt_set_log -- Set the log file to write log messages to
xslt_set_sax_handler -- Set SAX handlers for a XSLT processor
xslt_set_sax_handlers -- Set the SAX handlers to be called when the XML document gets processed
xslt_set_scheme_handler -- Set Scheme handlers for a XSLT processor
xslt_set_scheme_handlers -- Set the scheme handlers for the XSLT processor

xslt_create

(PHP 4 >= 4.0.3)

xslt_create -- Create a new XSLT processor

Description

resource **xslt_create** (void)

Create and return a new XSLT processor resource for manipulation by the other XSLT functions.

xslt_errno

(PHP 4 >= 4.0.3)

xslt_errno -- Return a error number

Description

int **xslt_errno** (resource xh)

Return an error code describing the last error that occured on the passed XSLT processor.

xslt_error

(PHP 4 >= 4.0.3)

xslt_error -- Return a error string

Description

mixed **xslt_error** (resource xh)

Return a string describing the last error that occured on the passed XSLT processor.

Example 1. Handling errors using the xslt_error() and xslt_errno() functions.

```
<?php

$xh = xslt_create();
$result = xslt_process($xh, 'dog.xml', 'pets.xsl');
if (!$result) {
    die(sprintf("Cannot process XSLT document [%d]: %s",
                xslt_errno($xh), xslt_error($xh)));
}

print($result);

xslt_free($xh);
?>
```

xslt_free

(PHP 4 >= 4.0.3)

xslt_free -- Free XSLT processor

Description

void **xslt_free** (resource xh)

Free the XSLT processor identified by the given handle.

xslt_process

(PHP 4 >= 4.0.3)

xslt_process -- Perform an XSLT transformation

Description

mixed **xslt_process** (resource xh, string xmlcontainer, string xslcontainer [, string resultcontainer [, array arguments [, array parameters]]])

The **xslt_process()** function is the crux of the new XSLT extension. It allows you to perform an XSLT transformation using almost any type of input source - the containers. This is accomplished through the use of argument buffers -- a concept taken from the Sablotron XSLT processor (currently the only XSLT processor this extension supports). The input containers default to a filename 'containing' the document to be processed. The result container defaults to a filename for the transformed document. If the result container is not specified - i.e. *NULL* - than the result is returned.

Warning
This function has changed it's arguments, sinceversion 4.0.6. Do NOT provide the actual xml or xsl content as 2nd and 3rd argument, as this will create a segmentation fault, in Sablotron versions up to and including 0.95.

Containers can also be set via the *$arguments* array (see below).

The simplest type of transformation with the **xslt_process()** function is the transformation of an XML file with an XSLT file, placing the result in a third file containing the new XML (or HTML) document. Doing this with sablotron is really quite easy...

Example 1. Using the xslt_process() to transform an XML file and a XSL file to a new XML file

```
<?php

// Allocate a new XSLT processor
$xh = xslt_create();

// Process the document
if (xslt_process($xh, 'sample.xml', 'sample.xsl', 'result.xml')) {
    print "SUCCESS, sample.xml was transformed by sample.xsl into result.xml";
    print ", result.xml has the following contents\n<br>\n";
    print "<pre>\n";
    readfile('result.xml');
    print "</pre>\n";
}
else {
    print "Sorry, sample.xml could not be transformed by sample.xsl into";
    print "  result.xml the reason is that " . xslt_error($xh) . " and the ";
    print "error code is " . xslt_errno($xh);
}

xslt_free($xh);

?>
```

While this functionality is great, many times, especially in a web environment, you want to be able to print out your results directly. Therefore, if you omit the third argument to the **xslt_process()** function (or provide a NULL value for the argument), it will automatically return the value of the XSLT transformation, instead of writing it to a file...

Example 2. Using the xslt_process() to transform an XML file and a XSL file to a variable containing the resulting XML data

```
<?php

// Allocate a new XSLT processor
$xh = xslt_create();

// Process the document, returning the result into the $result variable
$result = xslt_process($xh, 'sample.xml', 'sample.xsl');
if ($result) {
    print "SUCCESS, sample.xml was transformed by sample.xsl into the \$result";
```

```
    print " variable, the \$result variable has the following contents\n<br>\n";
    print "<pre>\n";
    print $result;
    print "</pre>\n";
}
else {
    print "Sorry, sample.xml could not be transformed by sample.xsl into";
    print "  the \$result variable the reason is that " . xslt_error($xh) .
    print " and the error code is " . xslt_errno($xh);
}

xslt_free($xh);

?>
```

The above two cases are the two simplest cases there are when it comes to XSLT transformation and I'd dare say that they are the most common cases, however, sometimes you get your XML and XSLT code from external sources, such as a database or a socket. In these cases you'll have the XML and/or XSLT data in a variable -- and in production applications the overhead of dumping these to file may be too much. This is where XSLT's "argument" syntax, comes to the rescue. Instead of files as the XML and XSLT arguments to the **xslt_process()** function, you can specify "argument place holders" which are then subsituted by values given in the arguments array (5th parameter to the **xslt_process()** function). The following is an example of processing XML and XSLT into a result variable without the use of files at all.

Example 3. Using the xslt_process() to transform a variable containing XML data and a variable containing XSL data into a variable containing the resulting XML data

```
<?php
// $xml and $xsl contain the XML and XSL data

$arguments = array(
     '/_xml' => $xml,
     '/_xsl' => $xsl
);

// Allocate a new XSLT processor
$xh = xslt_create();

// Process the document
$result = xslt_process($xh, 'arg:/_xml', 'arg:/_xsl', NULL, $arguments);
if ($result) {
    print "SUCCESS, sample.xml was transformed by sample.xsl into the \$result";
    print " variable, the \$result variable has the following contents\n<br>\n";
    print "<pre>\n";
    print $result;
    print "</pre>\n";
}
else {
    print "Sorry, sample.xml could not be transformed by sample.xsl into";
    print "  the \$result variable the reason is that " . xslt_error($xh) .
    print " and the error code is " . xslt_errno($xh);
}
xslt_free($xh);
?>
```

Finally, the last argument to the **xslt_process()** function represents an array for any top-level parameters that you want to pass to the XSLT document. These parameters can then be accessed within your XSL files using the <xsl:param name="parameter_name"> instruction. The parameters must be UTF-8 encoded and their values will be interpreted as strings by the Sablotron processor. In other words - you cannot pass node-sets as parameters to the XSLT document.

> **Note:** Please note that *file://* is needed in front of path if you use Windows.

xslt_set_base

(PHP 4 >= 4.0.5)

xslt_set_base -- Set the base URI for all XSLT transformations

Description

void **xslt_set_base** (resource xh, string uri)

Sets the base URI for all XSLT transformations, the base URI is used with Xpath instructions to resolve document() and other commands which access external resources. It is also used to resolve URIs for the <xsl:include> and <xsl:import> elements.

As of 4.3, the default base URI is the directory of the executing script. In effect, it is the directory name value of the **__FILE__** constant. Prior to 4.3, the default base URI was less predictable.

> **Note:** Please note that *file://* is needed in front of path if you use Windows.

xslt_set_encoding

(PHP 4 >= 4.0.5)

xslt_set_encoding -- Set the encoding for the parsing of XML documents

Description

void **xslt_set_encoding** (resource xh, string encoding)

Set the output encoding for the XSLT transformations. When using the Sablotron backend this option is only available when you compile Sablotron with encoding support.

xslt_set_error_handler

(PHP 4 >= 4.0.4)

xslt_set_error_handler -- Set an error handler for a XSLT processor

Description

void **xslt_set_error_handler** (resource xh, mixed handler)

Set an error handler function for the XSLT processor given by *xh*, this function will be called whenever an error occurs in the XSLT transformation (this function is also called for notices).

xslt_set_log

(PHP 4 >= 4.0.6)

xslt_set_log -- Set the log file to write log messages to

Description

void **xslt_set_log** (resource xh, mixed log)

xh

A reference to the XSLT parser.

log

This parameter is either a boolean value which toggles logging on and off, or a string containing the logfile in which log errors too.

This function allows you to set the file in which you want XSLT log messages to, XSLT log messages are different than error messages, in that log messages are not actually error messages but rather messages related to the state of the XSLT processor. They are useful for debugging XSLT, when something goes wrong.

By default logging is disabled, in order to enable logging you must first call **xslt_set_log()** with a boolean parameter which enables logging, then if you want to set the log file to debug to, you must then pass it a string containing the filename:

Example 1. Using the XSLT Logging features

```
<?php

$xh = xslt_create();
xslt_set_log($xh, true);
xslt_set_log($xh, getcwd() . 'myfile.log');

$result = xslt_process($xh, 'dog.xml', 'pets.xsl');
print($result);

xslt_free($xh);
?>
```

Note: Please note that *file://* is needed in front of path if you use Windows.

xslt_set_sax_handler

(4.0.3 - 4.0.6 only)

xslt_set_sax_handler -- Set SAX handlers for a XSLT processor

Description

void **xslt_set_sax_handler** (resource xh, array handlers)

Set SAX handlers on the resource handle given by *xh*. SAX handlers should be a two dimensional array with the format (all top level elements are optional):

```
array(
[document] =>
    array(
        start document handler,
        end document handler
    ),
[element] =>
    array(
        start element handler,
        end element handler
    ),
[namespace] =>
    array(
        start namespace handler,
        end namespace handler
    ),
[comment] => comment handler,
[pi] => processing instruction handler,
[character] => character data handler
)
```

xslt_set_sax_handlers

(PHP 4 >= 4.0.6)

xslt_set_sax_handlers -- Set the SAX handlers to be called when the XML document gets processed

Description

void **xslt_set_sax_handlers** (resource processor, array handlers)

Warning
This function is currently not documented; only the argument list is available.

xslt_set_scheme_handler

(4.0.5 - 4.0.6 only)

xslt_set_scheme_handler -- Set Scheme handlers for a XSLT processor

Description

void **xslt_set_scheme_handler** (resource xh, array handlers)

Set Scheme handlers on the resource handle given by *xh*. Scheme handlers should be an array with the format (all elements are optional):

```
array(
[get_all] => get all handler,
[open] => open handler,
[get] => get handler,
[put] => put handler,
[close] => close handler
```

)

xslt_set_scheme_handlers

(PHP 4 >= 4.0.6)

xslt_set_scheme_handlers -- Set the scheme handlers for the XSLT processor

Description

void **xslt_set_scheme_handlers** (resource processor, array handlers)

Warning
This function is currently not documented; only the argument list is available.

CX. YAZ functions

Introduction

This extension offers a PHP interface to the YAZ toolkit that implements the Z39.50 Protocol for Information Retrieval. With this extension you can easily implement a Z39.50 origin (client) that searches or scans Z39.50 targets (servers) in parallel.

The module hides most of the complexity of Z39.50 so it should be fairly easy to use. It supports persistent stateless connections very similar to those offered by the various RDB APIs that are available for PHP. This means that sessions are stateless but shared amongst users, thus saving the connect and initialize phase steps in most cases.

YAZ is available at http://www.indexdata.dk/yaz/. You can find news information, example scripts, etc. for this extension at http://www.indexdata.dk/phpyaz/.

Installation

Compile YAZ (ANSI/NISO Z39.50 support) and install it. Build PHP with your favourite modules and add option `--with-yaz[=DIR]`. Your task is roughly the following:

```
gunzip -c php-4.3.X.tar.gz|tar xf -
gunzip -c yaz-1.9.Y.tar.gz|tar xf -
cd yaz-1.9.Y
./configure --prefix=/usr
make
make install
cd ../php-4.3.X.
./configure --with-yaz=/usr/bin
make
make install
```

Warning
The IMAP extension cannot be used in conjuction with the recode or YAZ extensions. This is due to the fact that they both share the same internal symbol.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

Table 1. YAZ configuration options

Name	Default	Changeable
yaz.max_links	"100"	PHP_INI_ALL
yaz.log_file	""	PHP_INI_ALL

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Examples

PHP/YAZ keeps track of connections with targets (Z-Associations). A resource represents a connection to a target.

Example 1. Parallel searching using YAZ()

The script below demonstrates the parallel searching feature of the API. When invoked with no arguments it prints a query form; else (arguments are supplied) it searches the targets as given in array `host`.

```
$num_hosts = count ($host);
if (empty($term) || count($host) == 0) {
    echo '<form method="get">
    <input type="checkbox"
    name="host[]" value="bagel.indexdata.dk/gils">
        GILS test
    <input type="checkbox"
    name="host[]" value="localhost:9999/Default">
        local test
    <input type="checkbox" checked="1"
    name="host[]" value="z3950.loc.gov:7090/voyager">
        Library of Congress
    <br>
    RPN Query:
    <input type="text" size="30" name="term">
    <input type="submit" name="action" value="Search">
    ';
} else {
    echo 'You searced for ' . htmlspecialchars($term) . '<br>';
    for ($i = 0; $i < $num_hosts; $i++) {
        $id[] = yaz_connect($host[$i]);
        yaz_range($id[$i], 1, 10);
        yaz_search($id[$i],"rpn",$term);
    }
    yaz_wait();
    for ($i = 0; $i < $num_hosts; $i++) {
        echo '<hr>' . $host[$i] . ":";
        $error = yaz_error($id[$i]);
        if (!empty($error)) {
            echo "Error: $error";
        } else {
            $hits = yaz_hits($id[$i]);
            echo "Result Count $hits";
        }
        echo '<dl>';
        for ($p = 1; $p <= 10; $p++) {
            $rec = yaz_record($id[$i],$p,"string");
            if (empty($rec)) continue;
            echo "<dt><b>$p</b></dt><dd>";
            echo ereg_replace("\n", "<br>\n",$rec);
            echo "</dd>";
        }
        echo '</dl>';
    }
}
```

Table of Contents

yaz_addinfo

(PHP 4 >= 4.0.1)

yaz_addinfo -- Returns additional error information

Description

string **yaz_addinfo** (resource id)

Returns additional error message for target (last request). An empty string is returned if last operation was a success or if no additional information was provided by the target.

yaz_ccl_conf

(PHP 4 >= 4.0.5)

yaz_ccl_conf -- Configure CCL parser

Description

int **yaz_ccl_conf** (resource id, array config)

This function configures the CCL query parser for a target with definitions of access points (CCL qualifiers) and their mapping to RPN. To map a specific CCL query to RPN afterwards call the **yaz_ccl_parse()** function. Each index of the array *config* is the name of a CCL field and the corresponding value holds a string that specifies a mapping to RPN. The mapping is a sequence of attribute-type, attribute-value pairs. Attribute-type and attribute-value is separated by an equal sign (=). Each pair is separated by white space.

Example 1. CCL configuration

In the example below, the CCL parser is configured to support three CCL fields: `ti`, `au` and `isbn`. Each field is mapped to their BIB-1 equivalent. It is assumed that variable `$id` is the connection ID.

```
$fields["ti"] = "1=4";
$fields["au"] = "1=1";
$fields["isbn"] = "1=7";
yaz_ccl_conf($id,$fields);
```

yaz_ccl_parse

(PHP 4 >= 4.0.5)

yaz_ccl_parse -- Invoke CCL Parser

Description

int **yaz_ccl_parse** (resource id, string query, array & result)

This function invokes a CCL parser. It converts a given CCL FIND query to an RPN query which may be passed to the **yaz_search()** function to perform a search. To define a set of valid CCL fields call **yaz_ccl_conf()** prior to this function. If the supplied *query* was successfully converted to RPN, this function returns **TRUE**, and the index `rpn` of the supplied array *result* holds a valid RPN query. If the query could not be converted (because of invalid syntax, unknown field, etc.) this function returns **FALSE** and three indexes are set in the resulting array to indicate the cause of failure: `errorcode`CCL error code (integer), `errorstring`CCL error string, and `errorpos`approximate position in query of failure (integer is character position).

Example 1. CCL Parsing

We'll try to search using CCL. In the example below, `$ccl` is a CCL query.

```
yaz_ccl_conf($id,$fields);  // see example for yaz_ccl_conf
if (!yaz_ccl_parse($id, $ccl, &$cclresult) {
    echo 'Error: ' . $cclresult["errorstring"];
} else {
    $rpn = $cclresult["rpn"];
    yaz_search($id,"rpn",$rpn);
}
```

yaz_close

(PHP 4 >= 4.0.1)

yaz_close -- Close YAZ connection

Description

int **yaz_close** (resource id)

Closes connection given by *id*. The *id* is a connection resource as returned by a previous **yaz_connect()** call.

yaz_connect

(PHP 4 >= 4.0.1)

yaz_connect -- Prepares for a connection to a Z39.50 target (server).

Description

resource **yaz_connect** (string zurl [, mixed options])

This function returns a connection resource on success; zero on failure.

yaz_connect() prepares for a connection to a Z39.50 target. The *zurl* argument takes the form host[:port][/database]. If port is omitted 210 is used. If database is omitted `Default` is used. This function is non-blocking and doesn't attempt to establish a socket - it merely prepares a connect to be performed later when **yaz_wait()** is called.

If the second argument, *options*, is given as a string it is treated as the Z39.50 V2 authentication string (OpenAuth).

If *options* is given as an array the contents of the array serves as options. Note that array options are only supported for PHP 4.1.0 and later.

yaz_connect() options

user

Username for authentication.

group

Group for authentication.

password

Password for authentication.

cookie

Cookie for session (YAZ proxy).

proxy

Proxy for connection (YAZ proxy).

persistent

A boolean. If **TRUE** the connection is persistent; If **FALSE** the connection is not persistent. By default connections are persistent.

piggyback

A boolean. If **TRUE** piggyback is enabled for searches; If **FALSE** piggyback is disabled. By default piggyback is enabled. Enabling piggyback is more efficient and usually saves a network-round-trip for first time fetches of records. However, a few Z39.50 targets doesn't support piggyback or they ignore element set names. For those, piggyback should be disabled.

Note: The use of a proxy often improves performance. A Z39.50 proxy is part of the free YAZ++ package.

yaz_database

(PHP 4 >= 4.0.6)

yaz_database -- Specifies the databases within a session

Description

int **yaz_database** (resource id, string databases)

This function specifies one or more databases to be used in search, retrieval, etc. - overriding databases specified in call to **yaz_connect()**. Multiple databases are separated by a plus sign +.

This function allows you to use different sets of databases within a session.

Returns **TRUE** on success; **FALSE** on error.

yaz_element

(PHP 4 >= 4.0.1)

yaz_element -- Specifies Element-Set Name for retrieval

Description

int **yaz_element** (resource id, string elementset)

This function sets the element set name for retrieval. Call this function before **yaz_search()** or **yaz_present()** to specify the element set name for records to be retrieved. Most servers support F (for full records) and B (for brief records).

Returns **TRUE** on success; **FALSE** on error.

yaz_errno

(PHP 4 >= 4.0.1)

yaz_errno -- Returns error number

Description

int **yaz_errno** (resource id)

Returns error for target (last request). The error code is either a Z39.50 diagnostic code (usually a Bib-1 diagnostic) or a client

side error code which is generated by PHP/YAZ itself, such as "Connect failed", "Init Rejected", etc.

yaz_errno() should be called after network activity for each target - (after **yaz_wait()** returns) to determine the success or failure of the last operation (e.g. search). To get a text description of the error, call **yaz_error()**.

yaz_error

(PHP 4 >= 4.0.1)

yaz_error -- Returns error description

Description

string **yaz_error** (resource id)

Returns error text message for target (last request). An empty string is returned if last operation was a success.

yaz_error() returns an english text message corresponding to the last error number as returned by **yaz_errno()**.

yaz_get_option

(PHP 5 CVS only)

yaz_get_option -- Returns value of option for connection

Description

string **yaz_get_option** (resource id, string name)

Returns value of option with the *name* specified. If option is unset, an empty string is returned.

See description of **yaz_set_option()** for available options.

yaz_hits

(PHP 4 >= 4.0.1)

yaz_hits -- Returns number of hits for last search

Description

int **yaz_hits** (resource id)

yaz_hits() returns number of hits for last search.

yaz_itemorder

(PHP 4 >= 4.0.5)

yaz_itemorder -- Prepares for Z39.50 Item Order with an ILL-Request package

Description

int **yaz_itemorder** (resource id, array args)

This function prepares for an Extended Services request using the Profile for the Use of Z39.50 Item Order Extended Service to Transport ILL (Profile/1). See this and the specification. The args parameter must be a hash array with information about the Item Order request to be sent. The key of the hash is the name of the corresponding ASN.1 tag path. For example, the ISBN below the Item-ID has the key item-id,ISBN.

The ILL-Request parameters are:

protocol-version-num
transaction-id,initial-requester-id,person-or-institution-symbol,person
transaction-id,initial-requester-id,person-or-institution-symbol,institution
transaction-id,initial-requester-id,name-of-person-or-institution,name-of-person
transaction-id,initial-requester-id,name-of-person-or-institution,name-of-institution
transaction-id,transaction-group-qualifier
transaction-id,transaction-qualifier
transaction-id,sub-transaction-qualifier
service-date-time,this,date
service-date-time,this,time
service-date-time,original,date
service-date-time,original,time
requester-id,person-or-institution-symbol,person
requester-id,person-or-institution-symbol,institution
requester-id,name-of-person-or-institution,name-of-person
requester-id,name-of-person-or-institution,name-of-institution
responder-id,person-or-institution-symbol,person
responder-id,person-or-institution-symbol,institution
responder-id,name-of-person-or-institution,name-of-person
responder-id,name-of-person-or-institution,name-of-institution
transaction-type
delivery-address,postal-address,name-of-person-or-institution,name-of-person
delivery-address,postal-address,name-of-person-or-institution,name-of-institution
delivery-address,postal-address,extended-postal-delivery-address
delivery-address,postal-address,street-and-number
delivery-address,postal-address,post-office-box
delivery-address,postal-address,city
delivery-address,postal-address,region
delivery-address,postal-address,country
delivery-address,postal-address,postal-code
delivery-address,electronic-address,telecom-service-identifier
delivery-address,electronic-address,telecom-service-addreess
billing-address,postal-address,name-of-person-or-institution,name-of-person
billing-address,postal-address,name-of-person-or-institution,name-of-institution
billing-address,postal-address,extended-postal-delivery-address
billing-address,postal-address,street-and-number
billing-address,postal-address,post-office-box
billing-address,postal-address,city
billing-address,postal-address,region
billing-address,postal-address,country
billing-address,postal-address,postal-code
billing-address,electronic-address,telecom-service-identifier
billing-address,electronic-address,telecom-service-addreess
ill-service-type
requester-optional-messages,can-send-RECEIVED
requester-optional-messages,can-send-RETURNED
requester-optional-messages,requester-SHIPPED
requester-optional-messages,requester-CHECKED-IN
search-type,level-of-service
search-type,need-before-date
search-type,expiry-date
search-type,expiry-flag
place-on-hold
client-id,client-name
client-id,client-status
client-id,client-identifier
item-id,item-type
item-id,call-number
item-id,author
item-id,title
item-id,sub-title
item-id,sponsoring-body
item-id,place-of-publication
item-id,publisher
item-id,series-title-number
item-id,volume-issue
item-id,edition
item-id,publication-date
item-id,publication-date-of-component

item-id,author-of-article
item-id,title-of-article
item-id,pagination
item-id,ISBN
item-id,ISSN
item-id,additional-no-letters
item-id,verification-reference-source
copyright-complicance
retry-flag
forward-flag
requester-note
forward-note

There are also a few parameters that are part of the Extended Services Request package and the ItemOrder package:

package-name
user-id
contact-name
contact-phone
contact-email
itemorder-item

yaz_present

(PHP 4 >= 4.0.5)

yaz_present -- Prepares for retrieval (Z39.50 present).

Description

int **yaz_present** (resource id)

This function prepares for retrieval of records after a successful search. The **yaz_range()** should be called prior to this function to specify the range of records to be retrieved.

yaz_range

(PHP 4 >= 4.0.1)

yaz_range -- Specifies the maximum number of records to retrieve

Description

int **yaz_range** (resource id, int start, int number)

This function should be called before either **yaz_search()** or **yaz_present()** to specify a range of records to be retrieved. The *start* specifies position of first record to be retrieved and *number* is the number records. Records in a result set are numbered 1, 2, ... $hits where $hits is the count returned by **yaz_hits()**.

Returns **TRUE** on success; **FALSE** on error.

yaz_record

(PHP 4 >= 4.0.1)

yaz_record -- Returns a record

Description

string **yaz_record** (resource id, int pos, string type)

Returns record at position *pos* or empty string if no record exists at given position.

The **yaz_record()** function inspects a record in the current result set at the position specified. If no database record exists at the given position an empty string is returned. The *type* specifies the form of the returned record.

If type is "string" the record is returned in a string representation suitable for printing (for XML and SUTRS). If *type* is "array" the record is returned as an array representation (for structured records). If *type* is "raw" the record is returned in its original raw form.

Records in a result set are numbered 1, 2, ... $hits where $hits is the count returned by **yaz_hits()**.

yaz_scan_result

(PHP 4 >= 4.0.5)

yaz_scan_result -- Returns Scan Response result

Description

array **yaz_scan_result** (resource id [, array & result])

yaz_scan_result() returns terms and associated information as received from the target in the last performed **yaz_scan()**. This function returns an array (0..n-1) where n is the number of terms returned. Each value is a pair where first item is term, second item is result-count. If the *result* is given it will be modified to hold additional information taken from the Scan Response: number (number of entries returned), stepsize (Step-size), position (position of term), status (Scan Status).

yaz_scan

(PHP 4 >= 4.0.5)

yaz_scan -- Prepares for a scan

Description

int **yaz_scan** (resource id, string type, string startterm [, array flags])

This function prepares for a Z39.50 Scan Request. Argument *id* specifies connection. Starting term point for the scan is given by *startterm*. The form in which is the starting term is specified is given by *type*. Currently type rpn is supported. The optional *flags* specifies additional information to control the behaviour of the scan request. Three indexes are currently read from the flags: number (number of terms requested), position (preferred position of term) and stepSize (preferred step size). To actually tranfer the Scan Request to the target and receive the Scan Response, **yaz_wait()** must be called. Upon completion of **yaz_wait()** call **yaz_error()** and **yaz_scan_result()** to handle the response.

The syntax of *startterm* is similar to the RPN query as described in **yaz_search()**. The startterm consists of zero or more @attr-operator specifications, then followed by exactly one token.

Example 1. PHP function that scans titles

```
function scan_titles($id, $startterm) {
  yaz_scan($id,"rpn", "@attr 1=4 " . $startterm);
  yaz_wait();
  $errno = yaz_errno($id);
  if ($errno == 0) {
    $ar = yaz_scan_result($id,&$options);
    echo 'Scan ok; ';
    while(list($key,$val)=each($options)) {
      echo "$key = $val  ";
    }
    echo '<br><table><tr><td>';
    while(list($key,list($k, $term, $tcount))=each($ar)) {
      if (empty($k)) continue;
      echo "<tr><td>$term</td><td>";
      echo $tcount;
      echo "</td></tr>";
    }
    echo '</table>';
  } else {
    echo "Scan failed. Error: " . yaz_error($id) . "<br>";
  }
}
```

yaz_schema

(PHP 4 >= 4.2.0)

yaz_schema -- Specifies schema for retrieval.

Description

int **yaz_schema** (resource id, string schema)

The schema must be specified as an OID (Object Identifier) in a raw dot-notation (like `1.2.840.10003.13.4`) or as one of the known registered schemas: `GILS-schema`, `Holdings`, `Zthes`, ... This function should be called before **yaz_search()** or **yaz_present()**.

yaz_search

(PHP 4 >= 4.0.1)

yaz_search -- Prepares for a search

Description

int **yaz_search** (resource id, string type, string query)

yaz_search() prepares for a search on the connection given by *`id`*. The *`type`* represents the query type - only `"rpn"` is supported now in which case the third argument specifies a Type-1 query in prefix query notation. Like **yaz_connect()** this function is non-blocking and only prepares for a search to be executed later when **yaz_wait()** is called.

The RPN query

The RPN query is a textual represenation of the Type-1 query as defined by the Z39.50 standard. However, in the text representation as used by YAZ a prefix notation is used, that is the operater precedes the operands. The query string is a sequence of tokens where white space is ignored is ignored unless surrounded by double quotes. Tokens beginning with an at-character (`@`) are considered operators, otherwise they are treated as search terms.

Table 1. RPN Operators

Construct	Description
`@and` query1 query2	intersection of query1 and query2
`@or` query1 query2	union of query1 and query2
`@not` query1 query2	query1 and not query2
`@set` name	result set reference
`@attrset` set query	specifies attribute-set for query. This construction is only allowed once - in the beginning of the whole query
`@attr` [set] type=value query	applies attribute to query. The type and value are integers specifying the attribute-type and attribute-value respectively. The set, if given, specifies the attribute-set.

Example 1. Query Examples

You can search for simple terms, like this

```
computer
```

which matches documents where "computer" occur. No attributes are specified.

The Query

```
"knuth donald"
```

matches documents where "knuth donald" occur (provided that the server supports phrase search).

This query applies two attributes for the same phrase.

```
@attr 1=1003 @attr 4=1 "knuth donald"
```

First attribute is type 1 (Bib-1 use), attribute value is 1003 (Author). Second attribute has is type 4 (structure), value 1 (phrase), so this should match documents where Donald Knuth is author.

This query

```
@and @or a b @not @or c d e
```

would in infix notation look like `(a or b) and ((c or d) not e)`.

Another, more complex, one:

```
@attrset gils @and @attr 1=4 art @attr 1=2000 company
```

The query as a whole uses the GILS attributeset. The query matches documents where `art` occur in the title (GILS,BIB-1) and in which `company` occur as Distributor (GILS).

You can find information about attributes at the Z39.50 Maintenance Agency site.

> **Note:** If you would like to use a more friendly notation, use the CCL parser - functions **yaz_ccl_conf()** and **yaz_ccl_parse()**.

yaz_set_option

(PHP 5 CVS only)

yaz_set_option -- Sets one or more options for connection

Description

string **yaz_set_option** (resource id, string name, string value)

Sets option *`name`* to *`value`*.

Table 1. PYP/YAZ Connection Options

Name	Description
implementationName	implementation name of target
implementationVersion	implementation version of target
implementationId	implementation ID of target
schema	schema for retrieval. By default, no schema is used. Setting this option is equivalent to using function **yaz_schema()**
preferredRecordSyntax	record syntax for retrieval. By default, no syntax is used. Setting this option is equivalent to using function **yaz_syntax()**
start	offset for first record to be retrieved via **yaz_search()** or **yaz_present()**. Records are numbered from zero and upwards. Setting this option in combination with option `count` has the same effect as calling **yaz_range()** except that records are numbered from 1 in **yaz_range()**
count	maximum number of records to be retrieved via **yaz_search()** or **yaz_present()**.
elementSetName	element-set-name for retrieval. Setting this option is equivalent to calling **yaz_element()**.

yaz_sort

(PHP 4 >= 4.1.0)

yaz_sort -- Sets sorting criteria

Description

int **yaz_sort** (resource id, string criteria)

This function sets sorting criteria and enables Z39.50 Sort. Call this function *before* **yaz_search()**. Using this function alone doesn't have any effect. When in conjunction with **yaz_search()**, a Z39.50 Sort will be sent after a search response has been received and before any records are retrieved with Z39.50 Present. The *`criteria`* takes the form

`field1 flags1 field2 flags2` ...

where field1 specifies primary attributes for sort, field2 seconds, etc.. The field specifies either numerical attribute combinations consisting of type=value pairs separated by comma (e.g. `1=4,2=1`) ; or the field may specify a plain string criteria (e.g. `title`. The

flags is a sequnce of the following characters which may not be separated by any white space.

Sort Flags

`a`

Sort ascending

`d`

Sort descending

`i`

Case insensitive sorting

`s`

Case sensitive sorting

Example 1. Sort Criterias

To sort on Bib1 attribute title, case insensitive, and ascending you'd use the following sort criteria:

```
1=4 ia
```

If the secondary sorting criteria should be author, case sensitive and ascending you'd use:

```
1=4 ia 1=1003 sa
```

yaz_syntax

(PHP 4 >= 4.0.1)

yaz_syntax -- Specifies the preferred record syntax for retrieval.

Description

int **yaz_syntax** (resource id, string syntax)

The syntax must be specified as an OID (Object Identifier) in a raw dot-notation (like `1.2.840.10003.5.10`) or as one of the known registered record syntaxes (sutrs, usmarc, grs1, xml, etc.). This function should be called before **yaz_search()** or **yaz_present()**.

yaz_wait

(PHP 4 >= 4.0.1)

yaz_wait -- Wait for Z39.50 requests to complete

Description

int **yaz_wait** ([array options])

This function carries out networked (blocked) activity for outstanding requests which have been prepared by the functions **yaz_connect()**, **yaz_search()**, **yaz_present()**, **yaz_scan()** and **yaz_itemorder()**. **yaz_wait()** returns when all targets have either completed all requests or aborted (in case of errors).

If the *`options`* array is given that holds options that change the behaviour of **yaz_wait()**.

`timeout`

Sets timeout in seconds. If a target hasn't responded within the timeout it is considered dead and **yaz_wait()** returns. The default value for timeout is 15 seconds.

CXI. YP/NIS Functions

Introduction

NIS (formerly called Yellow Pages) allows network management of important administrative files (e.g. the password file). For more information refer to the NIS manpage and The Linux NIS(YP)/NYS/NIS+ HOWTO. There is also a book called Managing NFS and NIS by Hal Stern.

> **Note:** This extension is not available on Windows platforms.

Requirements

None besides functions from standard Unix libraries which are always available (either `libc` or `libnsl`, configure will detect which one to use).

Installation

To get these functions to work, you have to configure PHP with `--enable-yp`.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

YPERR_BADARGS (integer)

YPERR_BADDB (integer)

YPERR_BUSY (integer)

YPERR_DOMAIN (integer)

YPERR_KEY (integer)

YPERR_MAP (integer)

YPERR_NODOM (integer)

YPERR_NOMORE (integer)

YPERR_PMAP (integer)

YPERR_RESRC (integer)

YPERR_RPC (integer)

YPERR_YPBIND (integer)

YPERR_YPERR (integer)

YPERR_YPSERV (integer)

YPERR_VERS (integer)

Table of Contents

yp_all

(PHP 4 >= 4.0.6)

yp_all -- Traverse the map and call a function on each entry

Description

void **yp_all** (string domain, string map, string callback)

Warning
This function is currently not documented; only the argument list is available.

yp_cat

(PHP 4 >= 4.0.6)

yp_cat -- Return an array containing the entire map

Description

array **yp_cat** (string domain, string map)

Warning
This function is currently not documented; only the argument list is available.

yp_err_string

(PHP 4 >= 4.0.6)

yp_err_string -- Returns the error string associated with the previous operation

Description

string **yp_err_string** (void)

yp_err_string() returns the error message associated with the previous operation. Useful to indicate what exactly went wrong.

Example 1. Example for NIS errors

```
<?php
    echo "Error: " . yp_err_string();
?>
```

See also **yp_errno()**.

yp_errno

(PHP 4 >= 4.0.6)

yp_errno -- Returns the error code of the previous operation

Description

int **yp_errno** (void)

yp_errno() returns the error code of the previous operation.

Possible errors are:

1 args to function are bad
2 RPC failure - domain has been unbound
3 can't bind to server on this domain
4 no such map in server's domain
5 no such key in map
6 internal yp server or client error
7 resource allocation failure
8 no more records in map database
9 can't communicate with portmapper
10 can't communicate with ypbind
11 can't communicate with ypserv
12 local domain name not set
13 yp database is bad
14 yp version mismatch
15 access violation
16 database busy

See also **yp_err_string()**.

yp_first

(PHP 3>= 3.0.7, PHP 4)

yp_first -- Returns the first key-value pair from the named map

Description

array **yp_first** (string domain, string map)

yp_first() returns the first key-value pair from the named map in the named domain, otherwise **FALSE**.

Example 1. Example for the NIS first

```
<?php
$entry = yp_first($domain, "passwd.byname");
$key = $entry ["key"];
$value = $entry ["value"];
echo "First entry in this map has key " . $key . " and value " . $value;
?>
```

See also **yp-get-default-domain()**

yp_get_default_domain

(PHP 3>= 3.0.7, PHP 4)

yp_get_default_domain -- Fetches the machine's default NIS domain

Description

int **yp_get_default_domain** (void)

yp_get_default_domain() returns the default domain of the node or **FALSE**. Can be used as the domain parameter for successive NIS calls.

A NIS domain can be described a group of NIS maps. Every host that needs to look up information binds itself to a certain domain. Refer to the documents mentioned at the beginning for more detailed information.

Example 1. Example for the default domain

```
<?php
$domain = yp_get_default_domain();
echo "Default NIS domain is: " . $domain;
?>
```

yp_master

(PHP 3>= 3.0.7, PHP 4)

yp_master -- Returns the machine name of the master NIS server for a map

Description

string **yp_master** (string domain, string map)

yp_master() returns the machine name of the master NIS server for a map.

Example 1. Example for the NIS master

```
<?php
$number = yp_master ($domain, $mapname);
echo "Master for this map is: " . $master;
?>
```

See also **yp-get-default-domain()**.

yp_match

(PHP 3>= 3.0.7, PHP 4)

yp_match -- Returns the matched line

Description

string **yp_match** (string domain, string map, string key)

yp_match() returns the value associated with the passed key out of the specified map or **FALSE**. This key must be exact.

Example 1. Example for NIS match

```
<?php
$entry = yp_match ($domain, "passwd.byname", "joe");
echo "Matched entry is: " . $entry;
?>
```

In this case this could be: joe:##joe:11111:100:Joe User:/home/j/joe:/usr/local/bin/bash

See also **yp-get-default-domain()**

yp_next

(PHP 3>= 3.0.7, PHP 4)

yp_next -- Returns the next key-value pair in the named map.

Description

array **yp_next** (string domain, string map, string key)

yp_next() returns the next key-value pair in the named map after the specified key or **FALSE**.

Example 1. Example for NIS next

```
<?php
$entry = yp_next ($domain, "passwd.byname", "joe");

if (!$entry) {
        echo "No more entries found\n";
    <!-- echo yp_errno() . ": " . yp_err_string(); -->
}

$key = key ($entry);

echo "The next entry after joe has key " . $key
      . " and value " . $entry[$key];
?>
```

See also **yp-get-default-domain()**.

yp_order

(PHP 3>= 3.0.7, PHP 4)

yp_order -- Returns the order number for a map

Description

int **yp_order** (string domain, string map)

yp_order() returns the order number for a map or **FALSE**.

Example 1. Example for the NIS order

```
<?php
    $number = yp_order($domain,$mapname);
    echo "Order number for this map is: " . $number;
?>
```

See also **yp-get-default-domain()**.

CXII. Zip File Functions (Read Only Access)

Introduction

This module enables you to transparently read ZIP compressed archives and the files inside them.

Requirements

This module uses the functions of the ZZIPlib library by Guido Draheim. You need ZZIPlib version >= 0.10.6.

Note that ZZIPlib only provides a subset of functions provided in a full implementation of the ZIP compression algorithm and can only read ZIP file archives. A normal ZIP utility is needed to create the ZIP file archives read by this library.

Installation

Zip support in PHP is not enabled by default. You will need to use the `--with-zip[=DIR]` configuration option when compiling PHP to enable zip support.

> **Note:** Zip support before PHP 4.1.0 is experimental. This section reflects the Zip extension as it exists in PHP 4.1.0 and later.

Runtime Configuration

This extension has no configuration directives defined in `php.ini`.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Examples

This example opens a ZIP file archive, reads each file in the archive and prints out its contents. The `test2.zip` archive used in this example is one of the test archives in the ZZIPlib source distribution.

Example 1. Zip Usage Example

```
<?php

$zip = zip_open("/tmp/test2.zip");

if ($zip) {

    while ($zip_entry = zip_read($zip)) {
        echo "Name:               " . zip_entry_name($zip_entry) . "\n";
        echo "Actual Filesize:    " . zip_entry_filesize($zip_entry) . "\n";
        echo "Compressed Size:    " . zip_entry_compressedsize($zip_entry) . "\n";
        echo "Compression Method: " . zip_entry_compressionmethod($zip_entry) . "\n";

        if (zip_entry_open($zip, $zip_entry, "r")) {
            echo "File Contents:\n";
            $buf = zip_entry_read($zip_entry, zip_entry_filesize($zip_entry));
            echo "$buf\n";

            zip_entry_close($zip_entry);
        }
        echo "\n";

    }

    zip_close($zip);

}

?>
```

Table of Contents

zip_read -- Read Next Entry in a Zip File Archive

zip_close

(PHP 4 >= 4.1.0)

zip_close -- Close a Zip File Archive

Description

void **zip_close** (resource zip)

Closes a zip file archive. The parameter *zip* must be a zip archive previously opened by **zip_open()**.

This function has no return value.

See also **zip_open()** and **zip_read()**.

zip_entry_close

(PHP 4 >= 4.1.0)

zip_entry_close -- Close a Directory Entry

Description

void **zip_entry_close** (resource zip_entry)

Closes a directory entry specified by *zip_entry*. The parameter *zip_entry* must be a valid directory entry opened by **zip_entry_open()**.

This function has no return value.

See also **zip_entry_open()** and **zip_entry_read()**.

zip_entry_compressedsize

(PHP 4 >= 4.1.0)

zip_entry_compressedsize -- Retrieve the Compressed Size of a Directory Entry

Description

int **zip_entry_compressedsize** (resource zip_entry)

Returns the compressed size of the directory entry specified by *zip_entry*. The parameter *zip_entry* is a valid directory entry returned by **zip_read()**.

See also **zip_open()** and **zip_read()**.

zip_entry_compressionmethod

(PHP 4 >= 4.1.0)

zip_entry_compressionmethod -- Retrieve the Compression Method of a Directory Entry

Description

string **zip_entry_compressionmethod** (resource zip_entry)

Returns the compression method of the directory entry specified by *zip_entry*. The parameter *zip_entry* is a valid directory entry

returned by **zip_read()**.

See also **zip_open()** and **zip_read()**.

zip_entry_filesize

(PHP 4 >= 4.1.0)

zip_entry_filesize -- Retrieve the Actual File Size of a Directory Entry

Description

int **zip_entry_filesize** (resource zip_entry)

Returns the actual size of the directory entry specified by *zip_entry*. The parameter *zip_entry* is a valid directory entry returned by **zip_read()**.

See also **zip_open()** and **zip_read()**.

zip_entry_name

(PHP 4 >= 4.1.0)

zip_entry_name -- Retrieve the Name of a Directory Entry

Description

string **zip_entry_name** (resource zip_entry)

Returns the name of the directory entry specified by *zip_entry*. The parameter *zip_entry* is a valid directory entry returned by **zip_read()**.

See also **zip_open()** and **zip_read()**.

zip_entry_open

(PHP 4 >= 4.1.0)

zip_entry_open -- Open a Directory Entry for Reading

Description

bool **zip_entry_open** (resource zip, resource zip_entry [, string mode])

Opens a directory entry in a zip file for reading. The parameter *zip* is a valid resource handle returned by **zip_open()**. The parameter *zip_entry* is a directory entry resource returned by **zip_read()**. The optional parameter *mode* can be any of the modes specified in the documentaion for **fopen()**.

> **Note:** Currently, *mode* is ignored and is always `"rb"`. This is due to the fact that zip support in PHP is read only access. Please see **fopen()** for an explanation of various modes, including `"rb"`.

Returns **TRUE** on success or **FALSE** on failure.

> **Note:** Unlike **fopen()** and other similar functions, the return value of **zip_entry_open()** only indicates the result of the operation and is not needed for reading or closing the directory entry.

See also **zip_entry_read()** and **zip_entry_close()**.

zip_entry_read

(PHP 4 >= 4.1.0)

zip_entry_read -- Read From an Open Directory Entry

Description

string **zip_entry_read** (resource zip_entry [, int length])

Reads up to *length* bytes from an open directory entry. If *length* is not specified, then **zip_entry_read()** will attempt to read 1024 bytes. The parameter *zip_entry* is a valid directory entry returned by **zip_read()**.

> **Note:** The *length* parameter should be the uncompressed length you wish to read.

Returns the data read, or **FALSE** if the end of the file is reached.

See also **zip_entry_open()**, **zip_entry_close()** and **zip_entry_filesize()**.

zip_open

(PHP 4 >= 4.1.0)

zip_open -- Open a Zip File Archive

Description

resource **zip_open** (string filename)

Opens a new zip archive for reading. The *filename* parameter is the filename of the zip archive to open.

Returns a resource handle for later use with **zip_read()** and **zip_close()** or returns **FALSE** if *filename* does not exist.

See also **zip_read()** and **zip_close()**.

zip_read

(PHP 4 >= 4.1.0)

zip_read -- Read Next Entry in a Zip File Archive

Description

resource **zip_read** (resource zip)

Reads the next entry in a zip file archive. The parameter *zip* must be a zip archive previously opened by **zip_open()**.

Returns a directory entry resource for later use with the **zip_entry_...()** functions.

See also **zip_open()**, **zip_close()**, **zip_entry_open()**, and **zip_entry_read()**.

CXIII. Zlib Compression Functions

Introduction

This module enables you to transparently read and write gzip (.gz) compressed files, through versions of most of the filesystem functions which work with gzip-compressed files (and uncompressed files, too, but not with sockets).

> **Note:** Version 4.0.4 introduced a fopen-wrapper for .gz-files, so that you can use a special 'zlib:' URL to access compressed files transparently using the normal f*() file access functions if you prepend the filename or path with a 'zlib:' prefix when calling **fopen()**.
>
> In version 4.3.0, this special prefix has been changed to 'zlib://' to prevent ambiguities with filenames containing ':'.
>
> This feature requires a C runtime library that provides the `fopencookie()` function. To my current knowledge the GNU libc is the only library that provides this feature.

Requirements

This module uses the functions of zlib by Jean-loup Gailly and Mark Adler. You have to use a zlib version >= 1.0.9 with this module.

Installation

Zlib support in PHP is not enabled by default. You will need to configure PHP `--with-zlib[=DIR]`

The windows version of PHP has built in support for this extension. You do not need to load any additional extension in order to use these functions.

> **Note:** Builtin support for zlib is available with PHP 4.3.0.

Runtime Configuration

The behaviour of these functions is affected by settings in `php.ini`.

The zlib extension offers the option to transparently compress your pages on-the-fly, if the requesting browser supports this. Therefore there are three options in the configuration file `php.ini`.

Table 1. Zlib Configuration Options

Name	Default	Changeable
zlib.output_compression	"Off"	PHP_INI_ALL
zlib.output_compression_level	"-1"	PHP_INI_ALL
zlib.output_handler	""	PHP_INI_ALL

For further details and definition of the PHP_INI_* constants see **ini_set()**.

Here is a short explanation of the configuration directives.

`zlib.output_compression` **boolean/integer**

Whether to transparently compress pages. If this option is set to "On" in `php.ini` or the Apache configuration, pages are compressed if the browser sends an "Accept-Encoding: gzip" or "deflate" header. "Content-Encoding: gzip" (respectively "deflate") and "Vary: Accept-Encoding" headers are added to the output.

You can use **ini_set()** to disable this in your script if the headers aren't already sent. If you output a "Content-Type: image/" header the compression is disabled, too (in order to circumvent a Netscape bug). You can reenable it, if you add "ini_set('zlib.output_compression', 'On')" after the header call which added the image content-type.

This option also accepts integer values instead of boolean "On"/"Off", using this you can set the output buffer size (default is 4KB).

> **Note:** output_handler must be empty if this is set 'On' ! Instead you must use zlib.output_handler.

`zlib.output_compression_level` **integer**

Compression level used for transparent output compression.

`zlib.output_handler` **string**

You cannot specify additional output handlers if zlib.output_compression is activated here. This setting does the same as output_handler but in a different order.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

FORCE_GZIP (integer)

FORCE_DEFLATE (integer)

Examples

This example opens a temporary file and writes a test string to it, then it prints out the content of this file twice.

Example 1. Small Zlib Example

```
<?php

$filename = tempnam ('/tmp', 'zlibtest').'.gz';
print "<html>\n<head></head>\n<body>\n<pre>\n";
$s = "Only a test, test, test, test, test, test, test, test!\n";

// open file for writing with maximum compression
$zp = gzopen($filename, "w9");

// write string to file
gzwrite($zp, $s);

// close file
gzclose($zp);

// open file for reading
$zp = gzopen($filename, "r");

// read 3 char
print gzread($zp, 3);

// output until end of the file and close it.
gzpassthru($zp);

print "\n";

// open file and print content (the 2nd time).
if (readgzfile($filename) != strlen($s)) {
        echo "Error with zlib functions!";
}
unlink($filename);
print "</pre>\n</h1></body>\n</html>\n";

?>
```

Table of Contents

gzclose

(PHP 3, PHP 4)

gzclose -- Close an open gz-file pointer

Description

int **gzclose** (resource zp)

The gz-file pointed to by zp is closed.

Returns **TRUE** on success or **FALSE** on failure.

The gz-file pointer must be valid, and must point to a file successfully opened by **gzopen()**.

gzcompress

(PHP 4 >= 4.0.1)

gzcompress -- Compress a string

Description

string **gzcompress** (string data [, int level])

This function returns a compressed version of the input *data* using the ZLIB data format, or **FALSE** if an error is encountered. The optional parameter *level* can be given as 0 for no compression up to 9 for maximum compression.

For details on the ZLIB compression algorithm see the document "ZLIB Compressed Data Format Specification version 3.3" (RFC 1950).

> **Note:** This is *not* the same as gzip compression, which includes some header data. See **gzencode()** for gzip compression.

See also **gzdeflate()**, **gzinflate()**, **gzuncompress()**, **gzencode()**.

gzdeflate

(PHP 4 >= 4.0.4)

gzdeflate -- Deflate a string

Description

string **gzdeflate** (string data [, int level])

This function returns a compressed version of the input *data* using the DEFLATE data format, or **FALSE** if an error is encountered. The optional parameter *level* can be given as 0 for no compression up to 9 for maximum compression.

For details on the DEFLATE compression algorithm see the document "DEFLATE Compressed Data Format Specification version 1.3" (RFC 1951).

See also **gzinflate()**, **gzcompress()**, **gzuncompress()**, **gzencode()**.

gzencode

(PHP 4 >= 4.0.4)

gzencode -- Create a gzip compressed string

Description

string **gzencode** (string data [, int level [, int encoding_mode]])

This function returns a compressed version of the input *data* compatible with the output of the **gzip** program, or **FALSE** if an error is encountered. The optional parameter *level* can be given as 0 for no compression up to 9 for maximum compression, if not given the default compression level will be the default compression level of the zlib library.

You can also give **FORCE_GZIP** (the default) or **FORCE_DEFLATE** as optional third paramter *encoding_mode*. If you use **FORCE_DEFLATE**, you get a standard zlib deflated string (inclusive zlib headers) after the gzip file header but without the trailing crc32 checksum.

> **Note:** *level* was added in PHP 4.2, before PHP 4.2 **gzencode()** only had the *data* and (optional) *encoding_mode* parameters..

The resulting data contains the appropriate headers and data structure to make a standard .gz file, e.g.:

Example 1. Creating a gzip file

```
<?php
    $data = implode("", file("bigfile.txt"));
    $gzdata = gzencode($data, 9);
    $fp = fopen("bigfile.txt.gz", "w");
    fwrite($fp, $gzdata);
    fclose($fp);
?>
```

For more information on the GZIP file format, see the document: GZIP file format specification version 4.3 (RFC 1952).

See also **gzcompress()**. **gzuncompress()**, **gzdeflate()**, **gzinflate()**.

gzeof

(PHP 3, PHP 4)

gzeof -- Test for end-of-file on a gz-file pointer

Description

int **gzeof** (resource zp)

Returns **TRUE** if the gz-file pointer is at EOF or an error occurs; otherwise returns **FALSE**.

The gz-file pointer must be valid, and must point to a file successfully opened by **gzopen()**.

gzfile

(PHP 3, PHP 4)

gzfile -- Read entire gz-file into an array

Description

array **gzfile** (string filename [, int use_include_path])

Identical to **readgzfile()**, except that **gzfile()** returns the file in an array.

You can use the optional second parameter and set it to "1", if you want to search for the file in the include_path, too.

See also **readgzfile()**, and **gzopen()**.

gzgetc

(PHP 3, PHP 4)

gzgetc -- Get character from gz-file pointer

Description

string **gzgetc** (resource zp)

Returns a string containing a single (uncompressed) character read from the file pointed to by zp. Returns **FALSE** on EOF (as does **gzeof()**).

The gz-file pointer must be valid, and must point to a file successfully opened by **gzopen()**.

See also **gzopen()**, and **gzgets()**.

gzgets

(PHP 3, PHP 4)

gzgets -- Get line from file pointer

Description

string **gzgets** (resource zp, int length)

Returns a (uncompressed) string of up to length - 1 bytes read from the file pointed to by fp. Reading ends when length - 1 bytes have been read, on a newline, or on EOF (whichever comes first).

If an error occurs, returns **FALSE**.

The file pointer must be valid, and must point to a file successfully opened by **gzopen()**.

See also **gzopen()**, **gzgetc()**, and **fgets()**.

gzgetss

(PHP 3, PHP 4)

gzgetss -- Get line from gz-file pointer and strip HTML tags

Description

string **gzgetss** (resource zp, int length [, string allowable_tags])

Identical to **gzgets()**, except that **gzgetss()** attempts to strip any HTML and PHP tags from the text it reads.

You can use the optional third parameter to specify tags which should not be stripped.

> **Note:** *Allowable_tags* was added in PHP 3.0.13, PHP4B3.

See also **gzgets()**, **gzopen()**, and **strip_tags()**.

gzinflate

(PHP 4 >= 4.0.4)

gzinflate -- Inflate a deflated string

Description

string **gzinflate** (string data [, int length])

This function takes *data* compressed by **gzdeflate()** and returns the original uncompressed data or **FALSE** on error. The function will return an error if the uncompressed data is more than 32768 times the length of the compressed input *data* or more than the optional parameter *length*.

See also **gzcompress()**. **gzuncompress()**, **gzdeflate()**, **gzencode()**.

gzopen

(PHP 3, PHP 4)

gzopen -- Open gz-file

Description

resource **gzopen** (string filename, string mode [, int use_include_path])

Opens a gzip (.gz) file for reading or writing. The mode parameter is as in **fopen()** ("rb" or "wb") but can also include a compression level ("wb9") or a strategy: 'f' for filtered data as in "wb6f", 'h' for Huffman only compression as in "wb1h". (See the description of deflateInit2 in zlib.h for more information about the strategy parameter.)

gzopen() can be used to read a file which is not in gzip format; in this case **gzread()** will directly read from the file without decompression.

gzopen() returns a file pointer to the file opened, after that, everything you read from this file descriptor will be transparently decompressed and what you write gets compressed.

If the open fails, the function returns **FALSE**.

You can use the optional third parameter and set it to "1", if you want to search for the file in the include_path, too.

Example 1. gzopen() Example

```
$fp = gzopen ("/tmp/file.gz", "r");
```

See also **gzclose()**.

gzpassthru

(PHP 3, PHP 4)

gzpassthru -- Output all remaining data on a gz-file pointer

Description

int **gzpassthru** (resource zp)

Reads to EOF on the given gz-file pointer and writes the (uncompressed) results to standard output.

If an error occurs, returns **FALSE**.

The file pointer must be valid, and must point to a file successfully opened by **gzopen()**.

The gz-file is closed when **gzpassthru()** is done reading it (leaving *zp* useless).

gzputs

(PHP 3, PHP 4)

gzputs -- Write to a gz-file pointer

Description

int **gzputs** (resource zp, string str [, int length])

gzputs() is an alias to **gzwrite()**, and is identical in every way.

gzread

(PHP 3, PHP 4)

gzread -- Binary-safe gz-file read

Description

string **gzread** (resource zp, int length)

gzread() reads up to *length* bytes from the gz-file pointer referenced by *zp*. Reading stops when *length* (uncompressed) bytes have been read or EOF is reached, whichever comes first.

```
// get contents of a gz-file into a string
$filename = "/usr/local/something.txt.gz";
$zd = gzopen ($filename, "r");
$contents = gzread ($zd, 10000);
gzclose ($zd);
```

See also **gzwrite()**, **gzopen()**, **gzgets()**, **gzgetss()**, **gzfile()**, and **gzpassthru()**.

gzrewind

(PHP 3, PHP 4)

gzrewind -- Rewind the position of a gz-file pointer

Description

int **gzrewind** (resource zp)

Sets the file position indicator for zp to the beginning of the file stream.

If an error occurs, returns 0.

The file pointer must be valid, and must point to a file successfully opened by **gzopen()**.

See also **gzseek()** and **gztell()**.

gzseek

(PHP 3, PHP 4)

gzseek -- Seek on a gz-file pointer

Description

int **gzseek** (resource zp, int offset)

Sets the file position indicator for the file referenced by zp to offset bytes into the file stream. Equivalent to calling (in C) `gzseek( zp, offset, SEEK_SET )`.

If the file is opened for reading, this function is emulated but can be extremely slow. If the file is opened for writing, only forward seeks are supported; gzseek then compresses a sequence of zeroes up to the new starting position.

Upon success, returns 0; otherwise, returns -1. Note that seeking past EOF is not considered an error.

See also **gztell()** and **gzrewind()**.

gztell

(PHP 3, PHP 4)

gztell -- Tell gz-file pointer read/write position

Description

int **gztell** (resource zp)

Returns the position of the file pointer referenced by *zp*; i.e., its offset into the file stream.

If an error occurs, returns **FALSE**.

The file pointer must be valid, and must point to a file successfully opened by **gzopen()**.

See also **gzopen()**, **gzseek()** and **gzrewind()**.

gzuncompress

(PHP 4 >= 4.0.1)

gzuncompress -- Uncompress a deflated string

Description

string **gzuncompress** (string data [, int length])

This function takes *data* compressed by **gzcompress()** and returns the original uncompressed data or **FALSE** on error. The function will return an error if the uncompressed data is more than 32768 times the length of the compressed input *data* or more than the optional parameter *length*.

See also **gzdeflate()**, **gzinflate()**, **gzcompress()**, **gzencode()**.

gzwrite

(PHP 3, PHP 4)

gzwrite -- Binary-safe gz-file write

Description

int **gzwrite** (resource zp, string string [, int length])

gzwrite() writes the contents of *string* to the gz-file stream pointed to by *zp*. If the *length* argument is given, writing will stop after *length* (uncompressed) bytes have been written or the end of *string* is reached, whichever comes first.

gzwrite() returns the number of (uncompressed) bytes written to the gz-file stream pointed to by *zp*.

Note that if the *length* argument is given, then the magic_quotes_runtime configuration option will be ignored and no slashes will be stripped from *string*.

See also **gzread()**, **gzopen()**, and **gzputs()**.

readgzfile

(PHP 3, PHP 4)

readgzfile -- Output a gz-file

Description

int **readgzfile** (string filename [, int use_include_path])

Reads a file, decompresses it and writes it to standard output.

readgzfile() can be used to read a file which is not in gzip format; in this case **readgzfile()** will directly read from the file without decompression.

Returns the number of (uncompressed) bytes read from the file. If an error occurs, **FALSE** is returned and unless the function was called as `@readgzfile`, an error message is printed.

The file *`filename`* will be opened from the filesystem and its contents written to standard output.

You can use the optional second parameter and set it to "1", if you want to search for the file in the include_path, too.

See also **gzpassthru()**, **gzfile()**, and **gzopen()**.

V. Extending PHP 4.0

Hacking the Core of PHP

> Those who know don't talk.
>
> Those who talk don't know.

Sometimes, PHP "as is" simply isn't enough. Although these cases are rare for the average user, professional applications will soon lead PHP to the edge of its capabilities, in terms of either speed or functionality. New functionality cannot always be implemented natively due to language restrictions and inconveniences that arise when having to carry around a huge library of default code appended to every single script, so another method needs to be found for overcoming these eventual lacks in PHP.

As soon as this point is reached, it's time to touch the heart of PHP and take a look at its core, the C code that makes PHP go.

Table of Contents

Hacking the Core of PHP

Chapter 24. Overview

"Extending PHP" is easier said than done. PHP has evolved to a full-fledged tool consisting of a few megabytes of source code, and to hack a system like this quite a few things have to be learned and considered. When structuring this chapter, we finally decided on the "learn by doing" approach. This is not the most scientific and professional approach, but the method that's the most fun and gives the best end results. In the following sections, you'll learn quickly how to get the most basic extensions to work almost instantly. After that, you'll learn about Zend's advanced API functionality. The alternative would have been to try to impart the functionality, design, tips, tricks, etc. as a whole, all at once, thus giving a complete look at the big picture before doing anything practical. Although this is the "better" method, as no dirty hacks have to be made, it can be very frustrating as well as energy- and time-consuming, which is why we've decided on the direct approach.

Note that even though this chapter tries to impart as much knowledge as possible about the inner workings of PHP, it's impossible to really give a complete guide to extending PHP that works 100% of the time in all cases. PHP is such a huge and complex package that its inner workings can only be understood if you make yourself familiar with it by practicing, so we encourage you to work with the source.

What Is Zend? and What Is PHP?

The name *Zend* refers to the language engine, PHP's core. The term *PHP* refers to the complete system as it appears from the outside. This might sound a bit confusing at first, but it's not that complicated (see Figure 24-1). To implement a Web script interpreter, you need three parts:

1. The *interpreter* part analyzes the input code, translates it, and executes it.
2. The *functionality* part implements the functionality of the language (its functions, etc.).
3. The *interface* part talks to the Web server, etc.

Zend takes part 1 completely and a bit of part 2; PHP takes parts 2 and 3. Together they form the complete PHP package. Zend itself really forms only the language core, implementing PHP at its very basics with some predefined functions. PHP contains all the modules that actually create the language's outstanding capabilities.

Figure 24-1. The internal structure of PHP.

The following sections discuss where PHP can be extended and how it's done.

Chapter 25. Extension Possibilities

As shown in Figure 24-1 above, PHP can be extended primarily at three points: external modules, built-in modules, and the Zend engine. The following sections discuss these options.

External Modules

External modules can be loaded at script runtime using the function **dl()**. This function loads a shared object from disk and makes its functionality available to the script to which it's being bound. After the script is terminated, the external module is discarded from memory. This method has both advantages and disadvantages, as described in the following table:

Advantages	Disadvantages
External modules don't require recompiling of PHP.	The shared objects need to be loaded every time a script is being executed (every hit), which is very slow.
The size of PHP remains small by "outsourcing" certain functionality.	External additional files clutter up the disk.
	Every script that wants to use an external module's functionality has to specifically include a call to **dl()**, or the `extension` tag in `php.ini` needs to be modified (which is not always a suitable solution).

To sum up, external modules are great for third-party products, small additions to PHP that are rarely used, or just for testing purposes. To develop additional functionality quickly, external modules provide the best results. For frequent usage, larger implementations, and complex code, the disadvantages outweigh the advantages.

Third parties might consider using the `extension` tag in `php.ini` to create additional external modules to PHP. These external modules are completely detached from the main package, which is a very handy feature in commercial environments. Commercial distributors can simply ship disks or archives containing only their additional modules, without the need to create fixed and solid PHP binaries that don't allow other modules to be bound to them.

Built-in Modules

Built-in modules are compiled directly into PHP and carried around with every PHP process; their functionality is instantly available to every script that's being run. Like external modules, built-in modules have advantages and disadvantages, as described in the following table:

Advantages	Disadvantages
No need to load the module specifically; the functionality is instantly available.	Changes to built-in modules require recompiling of PHP.

No external files clutter up the disk; everything resides in the PHP binary.	The PHP binary grows and consumes more memory.

Built-in modules are best when you have a solid library of functions that remains relatively unchanged, requires better than poor-to-average performance, or is used frequently by many scripts on your site. The need to recompile PHP is quickly compensated by the benefit in speed and ease of use. However, built-in modules are not ideal when rapid development of small additions is required.

The Zend Engine

Of course, extensions can also be implemented directly in the Zend engine. This strategy is good if you need a change in the language behavior or require special functions to be built directly into the language core. In general, however, modifications to the Zend engine should be avoided. Changes here result in incompatibilities with the rest of the world, and hardly anyone will ever adapt to specially patched Zend engines. Modifications can't be detached from the main PHP sources and are overridden with the next update using the "official" source repositories. Therefore, this method is generally considered bad practice and, due to its rarity, is not covered in this book.

Chapter 26. Source Layout

> **Note:** Prior to working through the rest of this chapter, you should retrieve clean, unmodified source trees of your favorite Web server. We're working with Apache (available at http://www.apache.org/) and, of course, with PHP (available at http://www.php.net/ - does it need to be said?).
>
> Make sure that you can compile a working PHP environment by yourself! We won't go into this issue here, however, as you should already have this most basic ability when studying this chapter.

Before we start discussing code issues, you should familiarize yourself with the source tree to be able to quickly navigate through PHP's files. This is a must-have ability to implement and debug extensions.

The following table describes the contents of the major directories.

Directory	Contents
`php4`	Main PHP source files and main header files; here you'll find all of PHP's API definitions, macros, etc. (important). Everything else is below this directory.
`php4/ext`	Repository for dynamic and built-in modules; by default, these are the "official" PHP modules that have been integrated into the main source tree. From PHP 4.0, it's possible to compile these standard extensions as dynamic loadable modules (at least, those that support it).
`php4/main`	This directory contains the main php macros and definitions. (important)
`php4/pear`	Directory for the PHP Extension and Application Repository. This directory contains core PEAR files.
`php4/sapi`	Contains the code for the different server abstraction layers.
`php4/TSRM`	Location of the "Thread Safe Resource Manager" (TSRM) for Zend and PHP.
`php4/Zend`	Location of the Zend Engine files; here you'll find all of Zend's API definitions, macros, etc. (important).

Discussing all the files included in the PHP package is beyond the scope of this chapter. However, you should take a close look at the following files:

- `php4/main/php.h`, located in the main PHP directory. This file contains most of PHP's macro and API definitions.
- `php4/Zend/zend.h`, located in the main Zend directory. This file contains most of Zend's macros and definitions.
- `php4/Zend/zend_API.h`, also located in the Zend directory, which defines Zend's API.

You should also follow some sub-inclusions from these files; for example, the ones relating to the Zend executor, the PHP initialization file support, and such. After reading these files, take the time to navigate around the package a little to see the interdependencies of all files and modules - how they relate to each other and especially how they make use of each other. This also helps you to adapt to the coding style in which PHP is authored. To extend PHP, you should quickly adapt to this style.

Extension Conventions

Zend is built using certain conventions; to avoid breaking its standards, you should follow the rules described in the following sections.

Macros

For almost every important task, Zend ships predefined macros that are extremely handy. The tables and figures in the following sections describe most of the basic functions, structures, and macros. The macro definitions can be found mainly in `zend.h` and `zend_API.h`. We suggest that you take a close look at these files after having studied this chapter. (Although you can go ahead and read them now, not everything will make sense to you yet.)

Memory Management

Resource management is a crucial issue, especially in server software. One of the most valuable resources is memory, and memory management should be handled with extreme care. Memory management has been partially abstracted in Zend, and you should stick to this abstraction for obvious reasons: Due to the abstraction, Zend gets full control over all memory allocations. Zend is able to determine whether a block is in use, automatically freeing unused blocks and blocks with lost references, and thus prevent memory leaks. The functions to be used are described in the following table:

Function	Description
emalloc()	Serves as replacement for **malloc()**.
efree()	Serves as replacement for **free()**.
estrdup()	Serves as replacement for **strdup()**.
estrndup()	Serves as replacement for **strndup()**. Faster than **estrdup()** and binary-safe. This is the recommended function to use if you know the string length prior to duplicating it.
ecalloc()	Serves as replacement for **calloc()**.
erealloc()	Serves as replacement for **realloc()**.

emalloc(), **estrdup()**, **estrndup()**, **ecalloc()**, and **erealloc()** allocate internal memory; **efree()** frees these previously allocated blocks. Memory handled by the **e*()** functions is considered local to the current process and is discarded as soon as the script executed by this process is terminated.

Warning
To allocate resident memory that survives termination of the current script, you can use **malloc()** and **free()**. This should only be done with extreme care, however, and only in conjunction with demands of the Zend API; otherwise, you risk memory leaks.

Zend also features a thread-safe resource manager to provide better native support for multithreaded Web servers. This requires you to allocate local structures for all of your global variables to allow concurrent threads to be run. Because the thread-safe mode of Zend was not finished back when this was written, it is not yet extensively covered here.

Directory and File Functions

The following directory and file functions should be used in Zend modules. They behave exactly like their C counterparts, but provide virtual working directory support on the thread level.

Zend Function	Regular C Function
V_GETCWD()	**getcwd()**
V_FOPEN()	**fopen()**
V_OPEN()	**open()**
V_CHDIR()	**chdir()**
V_GETWD()	**getwd()**
V_CHDIR_FILE()	Takes a file path as an argument and changes the current working directory to that file's directory.
V_STAT()	**stat()**
V_LSTAT()	**lstat()**

String Handling

Strings are handled a bit differently by the Zend engine than other values such as integers, Booleans, etc., which don't require additional memory allocation for storing their values. If you want to return a string from a function, introduce a new string variable to the symbol table, or do something similar, you have to make sure that the memory the string will be occupying has previously been allocated, using the aforementioned **e*()** functions for allocation. (This might not make much sense to you yet; just keep it somewhere in your head for now - we'll get back to it shortly.)

Complex Types

Complex types such as arrays and objects require different treatment. Zend features a single API for these types - they're stored using hash tables.

> **Note:** To reduce complexity in the following source examples, we're only working with simple types such as integers at first. A discussion about creating more advanced types follows later in this chapter.

Chapter 27. PHP's Automatic Build System

PHP 4 features an automatic build system that's very flexible. All modules reside in a subdirectory of the `ext` directory. In addition to its own sources, each module consists of a config.m4 file, for extension configuration. (for example, see http://www.gnu.org/manual/m4/html_mono/m4.html)

All these stub files are generated automatically, along with `.cvsignore`, by a little shell script named `ext_skel` that resides in the `ext` directory. As argument it takes the name of the module that you want to create. The shell script then creates a directory of the same name, along with the appropriate stub files.

Step by step, the process looks like this:

```
:~/cvs/php4/ext:> ./ext_skel --extname=my_module
Creating directory my_module
Creating basic files: config.m4 .cvsignore my_module.c php_my_module.h CREDITS EXPERIMENTAL tests/001.phpt my_module.php [do

To use your new extension, you will have to execute the following steps:

1.  $ cd ..
2.  $ vi ext/my_module/config.m4
3.  $ ./buildconf
4.  $ ./configure --[with|enable]-my_module
5.  $ make
6.  $ ./php -f ext/my_module/my_module.php
7.  $ vi ext/my_module/my_module.c
8.  $ make

Repeat steps 3-6 until you are satisfied with ext/my_module/config.m4 and
step 6 confirms that your module is compiled into PHP. Then, start writing
code and repeat the last two steps as often as necessary.
```

This instruction creates the aforementioned files. To include the new module in the automatic configuration and build process, you have to run `buildconf`, which regenerates the `configure` script by searching through the `ext` directory and including all found `config.m4` files.

The default `config.m4` shown in Example 27-1 is a bit more complex:

Example 27-1. The default `config.m4`.

```
dnl $Id: Extending_Zend_Build.xml,v 1.8 2002/10/10 18:13:11 imajes Exp $
dnl config.m4 for extension my_module

dnl Comments in this file start with the string 'dnl'.
dnl Remove where necessary. This file will not work
dnl without editing.

dnl If your extension references something external, use with:

dnl PHP_ARG_WITH(my_module, for my_module support,
dnl Make sure that the comment is aligned:
dnl [  --with-my_module             Include my_module support])

dnl Otherwise use enable:
```

```
dnl PHP_ARG_ENABLE(my_module, whether to enable my_module support,
dnl Make sure that the comment is aligned:
dnl [  --enable-my_module           Enable my_module support])

if test "$PHP_MY_MODULE" != "no"; then
  dnl Write more examples of tests here...

  dnl # --with-my_module -> check with-path
  dnl SEARCH_PATH="/usr/local /usr"     # you might want to change this
  dnl SEARCH_FOR="/include/my_module.h"  # you most likely want to change this
  dnl if test -r $PHP_MY_MODULE/; then # path given as parameter
  dnl   MY_MODULE_DIR=$PHP_MY_MODULE
  dnl else # search default path list
  dnl   AC_MSG_CHECKING([for my_module files in default path])
  dnl   for i in $SEARCH_PATH ; do
  dnl     if test -r $i/$SEARCH_FOR; then
  dnl       MY_MODULE_DIR=$i
  dnl       AC_MSG_RESULT(found in $i)
  dnl     fi
  dnl   done
  dnl fi
  dnl
  dnl if test -z "$MY_MODULE_DIR"; then
  dnl   AC_MSG_RESULT([not found])
  dnl   AC_MSG_ERROR([Please reinstall the my_module distribution])
  dnl fi

  dnl # --with-my_module -> add include path
  dnl PHP_ADD_INCLUDE($MY_MODULE_DIR/include)

  dnl # --with-my_module -> chech for lib and symbol presence
  dnl LIBNAME=my_module # you may want to change this
  dnl LIBSYMBOL=my_module # you most likely want to change this

  dnl PHP_CHECK_LIBRARY($LIBNAME,$LIBSYMBOL,
  dnl [
  dnl   PHP_ADD_LIBRARY_WITH_PATH($LIBNAME, $MY_MODULE_DIR/lib, MY_MODULE_SHARED_LIBADD)
  dnl   AC_DEFINE(HAVE_MY_MODULELIB,1,[ ])
  dnl ],[
  dnl   AC_MSG_ERROR([wrong my_module lib version or lib not found])
  dnl ],[
  dnl   -L$MY_MODULE_DIR/lib -lm -ldl
  dnl ])
  dnl
  dnl PHP_SUBST(MY_MODULE_SHARED_LIBADD)

  PHP_NEW_EXTENSION(my_module, my_module.c, $ext_shared)
fi
```

If you're unfamiliar with M4 files (now is certainly a good time to get familiar), this might be a bit confusing at first; but it's actually quite easy.

Note: Everything prefixed with `dnl` is treated as a comment and is not parsed.

The `config.m4` file is responsible for parsing the command-line options passed to `configure` at configuration time. This means that it has to check for required external files and do similar configuration and setup tasks.

The default file creates two configuration directives in the `configure` script: `--with-my_module` and `--enable-my_module`. Use the first option when referring external files (such as the `--with-apache` directive that refers to the Apache directory). Use the second option when the user simply has to decide whether to enable your extension. Regardless of which option you use, you should uncomment the other, unnecessary one; that is, if you're using `--enable-my_module`, you should remove support for `--with-my_module`, and vice versa.

By default, the `config.m4` file created by `ext_skel` accepts both directives and automatically enables your extension. Enabling the extension is done by using the `PHP_EXTENSION` macro. To change the default behavior to include your module into the PHP binary when desired by the user (by explicitly specifying `--enable-my_module` or `--with-my_module`), change the test for `$PHP_MY_MODULE` to `== "yes"`:

```
if test "$PHP_MY_MODULE" == "yes"; then dnl
    Action.. PHP_EXTENSION(my_module, $ext_shared)
    fi
```

This would require you to use `--enable-my_module` each time when reconfiguring and recompiling PHP.

Note: Be sure to run `buildconf` every time you change `config.m4`!

We'll go into more details on the M4 macros available to your configuration scripts later in this chapter. For now, we'll simply use the default files.

Chapter 28. Creating Extensions

We'll start with the creation of a very simple extension at first, which basically does nothing more than implement a function that returns the integer it receives as parameter. Example 28-1 shows the source.

Example 28-1. A simple extension.

```
/* include standard header */
#include "php.h"

/* declaration of functions to be exported */
ZEND_FUNCTION(first_module);

/* compiled function list so Zend knows what's in this module */
zend_function_entry firstmod_functions[] =
{
    ZEND_FE(first_module, NULL)
    {NULL, NULL, NULL}
};

/* compiled module information */
zend_module_entry firstmod_module_entry =
{
    STANDARD_MODULE_HEADER,
    "First Module",
    firstmod_functions,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    NO_VERSION_YET,
    STANDARD_MODULE_PROPERTIES
};

/* implement standard "stub" routine to introduce ourselves to Zend */
#if COMPILE_DL_FIRST_MODULE
ZEND_GET_MODULE(firstmod)
#endif

/* implement function that is meant to be made available to PHP */
ZEND_FUNCTION(first_module)
{
    long parameter;

    if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "l", &parameter) == FAILURE) {
        return;
    }

    RETURN_LONG(parameter);
}
```

This code contains a complete PHP module. We'll explain the source code in detail shortly, but first we'd like to discuss the build process. (This will allow the impatient to experiment before we dive into API discussions.)

> **Note:** The example source makes use of some features introduced with the Zend version used in PHP 4.1.0 and above, it won't compile with older PHP 4.0.x versions.

Compiling Modules

There are basically two ways to compile modules:

- Use the provided "make" mechanism in the `ext` directory, which also allows building of dynamic loadable modules.
- Compile the sources manually.

The first method should definitely be favored, since, as of PHP 4.0, this has been standardized into a sophisticated build process. The fact that it is so sophisticated is also its drawback, unfortunately - it's hard to understand at first. We'll provide a more detailed introduction to this later in the chapter, but first let's work with the default files.

The second method is good for those who (for some reason) don't have the full PHP source tree available, don't have access to all files, or just like to juggle with their keyboard. These cases should be extremely rare, but for the sake of completeness we'll also describe this method.

Compiling Using Make. To compile the sample sources using the standard mechanism, copy all their subdirectories to the `ext` directory of your PHP source tree. Then run `buildconf`, which will create an updated `configure` script containing appropriate options for the new extension. By default, all the sample sources are disabled, so you don't have to fear breaking your build process.

After you run `buildconf`, `configure --help` shows the following additional modules:

```
--enable-array_experiments   BOOK: Enables array experiments
  --enable-call_userland       BOOK: Enables userland module
  --enable-cross_conversion    BOOK: Enables cross-conversion module
  --enable-first_module        BOOK: Enables first module
```

```
--enable-infoprint           BOOK: Enables infoprint module
--enable-reference_test      BOOK: Enables reference test module
--enable-resource_test       BOOK: Enables resource test module
--enable-variable_creation   BOOK: Enables variable-creation module
```

The module shown earlier in Example 28-1 can be enabled with `--enable-first_module` or `--enable-first_module=yes`.

Compiling Manually. To compile your modules manually, you need the following commands:

Action	Command
Compiling	cc -fpic -DCOMPILE_DL=1 -I/usr/local/include -I. -I.. -I../Zend -c -o <your_object_file> <your_c_file>
Linking	cc -shared -L/usr/local/lib -rdynamic -o <your_module_file> <your_object_file(s)>

The command to compile the module simply instructs the compiler to generate position-independent code (`-fpic` shouldn't be omitted) and additionally defines the constant `COMPILE_DL` to tell the module code that it's compiled as a dynamically loadable module (the test module above checks for this; we'll discuss it shortly). After these options, it specifies a number of standard include paths that should be used as the minimal set to compile the source files.

Note: All include paths in the example are relative to the directory `ext`. If you're compiling from another directory, change the pathnames accordingly. Required items are the PHP directory, the `Zend` directory, and (if necessary), the directory in which your module resides.

The link command is also a plain vanilla command instructing linkage as a dynamic module.

You can include optimization options in the compilation command, although these have been omitted in this example (but some are included in the makefile template described in an earlier section).

Note: Compiling and linking manually as a static module into the PHP binary involves very long instructions and thus is not discussed here. (It's not very efficient to type all those commands.)

Chapter 29. Using Extensions

Depending on the build process you selected, you should either end up with a new PHP binary to be linked into your Web server (or run as CGI), or with an .so (shared object) file. If you compiled the example file `first_module.c` as a shared object, your result file should be `first_module.so`. To use it, you first have to copy it to a place from which it's accessible to PHP. For a simple test procedure, you can copy it to your `htdocs` directory and try it with the source in Example 29-1. If you compiled it into the PHP binary, omit the call to **dl()**, as the module's functionality is instantly available to your scripts.

Warning
For security reasons, you *should not* put your dynamic modules into publicly accessible directories. Even though it *can* be done and it simplifies testing, you should put them into a separate directory in production environments.

Example 29-1. A test file for first_module.so.

```
<?php

// remove next comment if necessary
// dl("first_module.so");

$param = 2;
$return = first_module($param);

print("We sent '$param' and got '$return'");

?>
```

Calling this PHP file in your Web browser should give you the output shown in Figure 29-1.

Figure 29-1. Output of first_module.php.

If required, the dynamic loadable module is loaded by calling the **dl()** function. This function looks for the specified shared object, loads it, and makes its functions available to PHP. The module exports the function **first_module()**, which accepts a single parameter, converts it to an integer, and returns the result of the conversion.

If you've gotten this far, congratulations! You just built your first extension to PHP.

Chapter 30. Troubleshooting

Actually, not much troubleshooting can be done when compiling static or dynamic modules. The only problem that could arise is that the compiler will complain about missing definitions or something similar. In this case, make sure that all header files are available and that you specified their path correctly in the compilation command. To be sure that everything is located correctly, extract a clean PHP source tree and use the automatic build in the `ext` directory with the fresh files; this will guarantee a safe compilation environment. If this fails, try manual compilation.

PHP might also complain about missing functions in your module. (This shouldn't happen with the sample sources if you didn't modify them.) If the names of external functions you're trying to access from your module are misspelled, they'll remain as "unlinked symbols" in the symbol table. During dynamic loading and linkage by PHP, they won't resolve because of the typing errors - there are no corresponding symbols in the main binary. Look for incorrect declarations in your module file or incorrectly written external references. Note that this problem is specific to dynamic loadable modules; it doesn't occur with static modules. Errors in static modules show up at compile time.

Chapter 31. Source Discussion

Now that you've got a safe build environment and you're able to include the modules into PHP files, it's time to discuss how everything works.

Module Structure

All PHP modules follow a common structure:

- Header file inclusions (to include all required macros, API definitions, etc.)
- C declaration of exported functions (required to declare the Zend function block)
- Declaration of the Zend function block
- Declaration of the Zend module block
- Implementation of **get_module()**
- Implementation of all exported functions

Header File Inclusions

The only header file you really have to include for your modules is `php.h`, located in the PHP directory. This file makes all macros and API definitions required to build new modules available to your code.

Tip: It's good practice to create a separate header file for your module that contains module-specific definitions. This header file should contain all the forward definitions for exported functions and include `php.h`. If you created your module using `ext_skel` you already have such a header file prepared.

Declaring Exported Functions

To declare functions that are to be exported (i.e., made available to PHP as new native functions), Zend provides a set of macros. A sample declaration looks like this:

```
ZEND_FUNCTION ( my_function );
```

`ZEND_FUNCTION` declares a new C function that complies with Zend's internal API. This means that the function is of type `void` and accepts `INTERNAL_FUNCTION_PARAMETERS` (another macro) as parameters. Additionally, it prefixes the function name with `zif`. The immediately expanded version of the above definitions would look like this:

```
void zif_my_function ( INTERNAL_FUNCTION_PARAMETERS );
```

Expanding `INTERNAL_FUNCTION_PARAMETERS` results in the following:

```
void zif_my_function( int ht
                    , zval * return_value
                    , zval * this_ptr
                    , int return_value_used
                    , zend_executor_globals * executor_globals
                    );
```

Since the interpreter and executor core have been separated from the main PHP package, a second API defining macros and function sets has evolved: the Zend API. As the Zend API now handles quite a few of the responsibilities that previously belonged to PHP, a lot of PHP functions have been reduced to macros aliasing to calls into the Zend API. The recommended practice is to use the Zend API wherever possible, as the old API is only preserved for compatibility reasons. For example, the types `zval` and `pval` are identical. `zval` is Zend's definition; `pval` is PHP's definition (actually, `pval` is an alias for `zval` now). As the macro `INTERNAL_FUNCTION_PARAMETERS` is a Zend macro, the above declaration contains `zval`. When writing code, you should always use `zval` to conform to the new Zend API.

The parameter list of this declaration is very important; you should keep these parameters in mind (see Table 31-1 for descriptions).

Table 31-1. Zend's Parameters to Functions Called from PHP

Parameter	Description
`ht`	The number of arguments passed to the Zend function. You should not touch this directly, but instead use ZEND_NUM_ARGS() to obtain the value.
`return_value`	This variable is used to pass any return values of your function back to PHP. Access to this variable is best done using the predefined macros. For a description of these see below.
`this_ptr`	Using this variable, you can gain access to the object in which your function is contained, if it's used within an object. Use the function **getThis()** to obtain this pointer.
`return_value_used`	This flag indicates whether an eventual return value from this function will actually be used by the calling script. `0` indicates that the return value is not used; `1` indicates that the caller expects a return value. Evaluation of this flag can be done to verify correct usage of the function as well as speed optimizations in case returning a value requires expensive operations (for an example, see how `array.c` makes use of this).
`executor_globals`	This variable points to global settings of the Zend engine. You'll find this useful when creating new variables, for example (more about this later). The executor globals can also be introduced to your function by using the macro `TSRMLS_FETCH()`.

Declaration of the Zend Function Block

Now that you have declared the functions to be exported, you also have to introduce them to Zend. Introducing the list of functions is done by using an array of `zend_function_entry`. This array consecutively contains all functions that are to be made available externally, with the function's name as it should appear in PHP and its name as defined in the C source. Internally, `zend_function_entry` is defined as shown in Example 31-1.

Example 31-1. Internal declaration of `zend_function_entry`.

```
typedef struct _zend_function_entry {
    char *fname;
    void (*handler)(INTERNAL_FUNCTION_PARAMETERS);
    unsigned char *func_arg_types;
} zend_function_entry;
```

Entry	Description
`fname`	Denotes the function name as seen in PHP (for example, `fopen`, `mysql_connect`, or, in our example, `first_module`).
`handler`	Pointer to the C function responsible for handling calls to this function. For example, see the standard macro `INTERNAL_FUNCTION_PARAMETERS` discussed earlier.
`func_arg_types`	Allows you to mark certain parameters so that they're forced to be passed by reference. You usually should set this to NULL.

In the example above, the declaration looks like this:

```
zend_function_entry firstmod_functions[] =
```

```
{
    ZEND_FE(first_module, NULL)
    {NULL, NULL, NULL}
};
```

You can see that the last entry in the list always has to be `{NULL, NULL, NULL}`. This marker has to be set for Zend to know when the end of the list of exported functions is reached.

> **Note:** You *cannot* use the predefined macros for the end marker, as these would try to refer to a function named "NULL"!

The macro `ZEND_FE` (short for 'Zend Function Entry') simply expands to a structure entry in `zend_function_entry`. Note that these macros introduce a special naming scheme to your functions - your C functions will be prefixed with `zif_`, meaning that `ZEND_FE(first_module)` will refer to a C function **zif_first_module()**. If you want to mix macro usage with hand-coded entries (not a good practice), keep this in mind.

Tip: Compilation errors that refer to functions named **zif_*()** relate to functions defined with `ZEND_FE`.

Table 31-2 shows a list of all the macros that you can use to define functions.

Table 31-2. Macros for Defining Functions

Macro Name	Description
`ZEND_FE(name, arg_types)`	Defines a function entry of the name `name` in `zend_function_entry`. Requires a corresponding C function. `arg_types` needs to be set to `NULL`. This function uses automatic C function name generation by prefixing the PHP function name with `zif_`. For example, `ZEND_FE("first_module", NULL)` introduces a function **first_module()** to PHP and links it to the C function **zif_first_module()**. Use in conjunction with `ZEND_FUNCTION`.
`ZEND_NAMED_FE(php_name, name, arg_types)`	Defines a function that will be available to PHP by the name `php_name` and links it to the corresponding C function `name`. `arg_types` needs to be set to `NULL`. Use this function if you don't want the automatic name prefixing introduced by `ZEND_FE`. Use in conjunction with `ZEND_NAMED_FUNCTION`.
`ZEND_FALIAS(name, alias, arg_types)`	Defines an alias named `alias` for `name`. `arg_types` needs to be set to `NULL`. Doesn't require a corresponding C function; refers to the alias target instead.
`PHP_FE(name, arg_types)`	Old PHP API equivalent of `ZEND_FE`.
`PHP_NAMED_FE(runtime_name, name, arg_types)`	Old PHP API equivalent of `ZEND_NAMED_FE`.

Note: You can't use `ZEND_FE` in conjunction with `PHP_FUNCTION`, or `PHP_FE` in conjunction with `ZEND_FUNCTION`. However, it's perfectly legal to mix `ZEND_FE` and `ZEND_FUNCTION` with `PHP_FE` and `PHP_FUNCTION` when staying with the same macro set for each function to be declared. But mixing is *not* recommended; instead, you're advised to use the `ZEND_*` macros only.

Declaration of the Zend Module Block

This block is stored in the structure `zend_module_entry` and contains all necessary information to describe the contents of this module to Zend. You can see the internal definition of this module in Example 31-2.

Example 31-2. Internal declaration of `zend_module_entry`.

```
typedef struct _zend_module_entry zend_module_entry;

    struct _zend_module_entry {
    unsigned short size;
    unsigned int zend_api;
    unsigned char zend_debug;
    unsigned char zts;
    char *name;
    zend_function_entry *functions;
    int (*module_startup_func)(INIT_FUNC_ARGS);
    int (*module_shutdown_func)(SHUTDOWN_FUNC_ARGS);
    int (*request_startup_func)(INIT_FUNC_ARGS);
    int (*request_shutdown_func)(SHUTDOWN_FUNC_ARGS);
    void (*info_func)(ZEND_MODULE_INFO_FUNC_ARGS);
    char *version;

[ Rest of the structure is not interesting here ]

};
```

Entry	Description
size, zend_api, zend_debug and zts	Usually filled with the "STANDARD_MODULE_HEADER", which fills these four members with the size of the whole zend_module_entry, the ZEND_MODULE_API_NO, whether it is a debug build or normal build (ZEND_DEBUG) and if ZTS is enabled (USING_ZTS).
name	Contains the module name (for example, "File functions", "Socket functions", "Crypt", etc.). This name will show up in **phpinfo()**, in the section "Additional Modules."
functions	Points to the Zend function block, discussed in the preceding section.
module_startup_func	This function is called once upon module initialization and can be used to do one-time initialization steps (such as initial memory allocation, etc.). To indicate a failure during initialization, return FAILURE; otherwise, SUCCESS. To mark this field as unused, use NULL. To declare a function, use the macro ZEND_MINIT.
module_shutdown_func	This function is called once upon module shutdown and can be used to do one-time deinitialization steps (such as memory deallocation). This is the counterpart to **module_startup_func()**. To indicate a failure during deinitialization, return FAILURE; otherwise, SUCCESS. To mark this field as unused, use NULL. To declare a function, use the macro ZEND_MSHUTDOWN.
request_startup_func	This function is called once upon every page request and can be used to do one-time initialization steps that are required to process a request. To indicate a failure here, return FAILURE; otherwise, SUCCESS. *Note:* As dynamic loadable modules are loaded only on page requests, the request startup function is called right after the module startup function (both initialization events happen at the same time). To mark this field as unused, use NULL. To declare a function, use the macro ZEND_RINIT.
request_shutdown_func	This function is called once after every page request and works as counterpart to **request_startup_func()**. To indicate a failure here, return FAILURE; otherwise, SUCCESS. *Note:* As dynamic loadable modules are loaded only on page requests, the request shutdown function is immediately followed by a call to the module shutdown handler (both deinitialization events happen at the same time). To mark this field as unused, use NULL. To declare a function, use the macro ZEND_RSHUTDOWN.
info_func	When **phpinfo()** is called in a script, Zend cycles through all loaded modules and calls this function. Every module then has the chance to print its own "footprint" into the output page. Generally this is used to dump environmental or statistical information. To mark this field as unused, use NULL. To declare a function, use the macro ZEND_MINFO.
version	The version of the module. You can use NO_VERSION_YET if you don't want to give the module a version number yet, but we really recommend that you add a version string here. Such a version string can look like this (in chronological order): "2.5-dev", "2.5RC1", "2.5" or "2.5pl3".
Remaining structure elements	These are used internally and can be prefilled by using the macro STANDARD_MODULE_PROPERTIES_EX. You should not assign any values to them. Use STANDARD_MODULE_PROPERTIES_EX only if you use global startup and shutdown functions; otherwise, use STANDARD_MODULE_PROPERTIES directly.

In our example, this structure is implemented as follows:

```
zend_module_entry firstmod_module_entry =
{
    STANDARD_MODULE_HEADER,
    "First Module",
    firstmod_functions,
    NULL, NULL, NULL, NULL, NULL,
    NO_VERSION_YET,
    STANDARD_MODULE_PROPERTIES,
};
```

This is basically the easiest and most minimal set of values you could ever use. The module name is set to First Module, then the function list is referenced, after which all startup and shutdown functions are marked as being unused.

For reference purposes, you can find a list of the macros involved in declared startup and shutdown functions in Table 31-3. These are not used in our basic example yet, but will be demonstrated later on. You should make use of these macros to declare your startup and shutdown functions, as these require special arguments to be passed (INIT_FUNC_ARGS and SHUTDOWN_FUNC_ARGS), which are automatically included into the function declaration when using the predefined macros. If you declare your functions manually and the PHP developers decide that a change in the argument list is necessary, you'll have to change your module sources to remain compatible.

Table 31-3. Macros to Declare Startup and Shutdown Functions

Macro	Description
ZEND_MINIT(module)	Declares a function for module startup. The generated name will be zend_minit_<module> (for example, zend_minit_first_module). Use in conjunction with ZEND_MINIT_FUNCTION.
ZEND_MSHUTDOWN(module)	Declares a function for module shutdown. The generated name will be zend_mshutdown_<module> (for example, zend_mshutdown_first_module). Use in conjunction with ZEND_MSHUTDOWN_FUNCTION.
ZEND_RINIT(module)	Declares a function for request startup. The generated name will be zend_rinit_<module> (for example, zend_rinit_first_module). Use in conjunction with ZEND_RINIT_FUNCTION.
ZEND_RSHUTDOWN(module)	Declares a function for request shutdown. The generated name will be zend_rshutdown_<module> (for example, zend_rshutdown_first_module). Use in conjunction with ZEND_RSHUTDOWN_FUNCTION.
ZEND_MINFO(module)	Declares a function for printing module information, used when **phpinfo()** is called. The generated name will be zend_info_<module> (for example, zend_info_first_module). Use in conjunction with ZEND_MINFO_FUNCTION.

Creation of get_module()

This function is special to all dynamic loadable modules. Take a look at the creation via the ZEND_GET_MODULE macro first:

```
#if COMPILE_DL_FIRSTMOD
     ZEND_GET_MODULE(firstmod)
#endif
```

The function implementation is surrounded by a conditional compilation statement. This is needed since the function **get_module()** is only required if your module is built as a dynamic extension. By specifying a definition of COMPILE_DL_FIRSTMOD in the compiler command (see above for a discussion of the compilation instructions required to build a dynamic extension), you can instruct your module whether you intend to build it as a dynamic extension or as a built-in module. If you want a built-in module, the implementation of **get_module()** is simply left out.

get_module() is called by Zend at load time of the module. You can think of it as being invoked by the **dl()** call in your script. Its purpose is to pass the module information block back to Zend in order to inform the engine about the module contents.

If you don't implement a **get_module()** function in your dynamic loadable module, Zend will compliment you with an error message when trying to access it.

Implementation of All Exported Functions

Implementing the exported functions is the final step. The example function in first_module looks like this:

```
ZEND_FUNCTION(first_module)
{
    long parameter;

    if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "l", &parameter) == FAILURE) {
        return;
    }

    RETURN_LONG(parameter);
}
```

The function declaration is done using ZEND_FUNCTION, which corresponds to ZEND_FE in the function entry table (discussed earlier).

After the declaration, code for checking and retrieving the function's arguments, argument conversion, and return value generation follows (more on this later).

Summary

That's it, basically - there's nothing more to implementing PHP modules. Built-in modules are structured similarly to dynamic modules, so, equipped with the information presented in the previous sections, you'll be able to fight the odds when encountering PHP module source files.

Now, in the following sections, read on about how to make use of PHP's internals to build powerful extensions.

Chapter 32. Accepting Arguments

One of the most important issues for language extensions is accepting and dealing with data passed via arguments. Most extensions are built to deal with specific input data (or require parameters to perform their specific actions), and function arguments are the only real way to exchange data between the PHP level and the C level. Of course, there's also the possibility of exchanging data using predefined global values (which is also discussed later), but this should be avoided by all means, as it's extremely bad practice.

PHP doesn't make use of any formal function declarations; this is why call syntax is always completely dynamic and never checked for errors. Checking for correct call syntax is left to the user code. For example, it's possible to call a function using only one argument at one time and four arguments the next time - both invocations are syntactically absolutely correct.

Determining the Number of Arguments

Since PHP doesn't have formal function definitions with support for call syntax checking, and since PHP features variable arguments, sometimes you need to find out with how many arguments your function has been called. You can use the `ZEND_NUM_ARGS` macro in this case. In previous versions of PHP, this macro retrieved the number of arguments with which the function has been called based on the function's hash table entry, `ht`, which is passed in the `INTERNAL_FUNCTION_PARAMETERS` list. As `ht` itself now contains the number of arguments that have been passed to the function, `ZEND_NUM_ARGS` has been stripped down to a dummy macro (see its definition in `zend_API.h`). But it's still good practice to use it, to remain compatible with future changes in the call interface. *Note:* The old PHP equivalent of this macro is `ARG_COUNT`.

The following code checks for the correct number of arguments:

```
if(ZEND_NUM_ARGS() != 2) WRONG_PARAM_COUNT;
```

If the function is not called with two arguments, it exits with an error message. The code snippet above makes use of the tool macro `WRONG_PARAM_COUNT`, which can be used to generate a standard error message (see Figure 32-1).

Figure 32-1. `WRONG_PARAM_COUNT` in action.

This macro prints a default error message and then returns to the caller. Its definition can also be found in `zend_API.h` and looks like this:

```
ZEND_API void wrong_param_count(void);

#define WRONG_PARAM_COUNT { wrong_param_count(); return; }
```

As you can see, it calls an internal function named **wrong_param_count()** that's responsible for printing the warning. For details on generating customized error messages, see the later section "Printing Information."

Retrieving Arguments

> **New parameter parsing API:** This chapter documents the new Zend parameter parsing API introduced by Andrei Zmievski. It was introduced in the development stage between PHP 4.0.6 and 4.1.0 .

Parsing parameters is a very common operation and it may get a bit tedious. It would also be nice to have standardized error checking and error messages. Since PHP 4.1.0, there is a way to do just that by using the new parameter parsing API. It greatly simplifies the process of receiving parameteres, but it has a drawback in that it can't be used for functions that expect variable number of parameters. But since the vast majority of functions do not fall into those categories, this parsing API is recommended as the new standard way.

The prototype for parameter parsing function looks like this:

```
int zend_parse_parameters(int num_args TSRMLS_DC, char *type_spec, ...);
```

The first argument to this function is supposed to be the number of actual parameters passed to your function, so `ZEND_NUM_ARGS()` can be used for that. The second parameter should always be `TSRMLS_CC` macro. The third argument is a string that specifies the number and types of arguments your function is expecting, similar to how printf format string specifies the number and format of the output values it should operate on. And finally the rest of the arguments are pointers to variables which should receive the values from the parameters.

zend_parse_parameters() also performs type conversions whenever possible, so that you always receive the data in the format

you asked for. Any type of scalar can be converted to another one, but conversions between complex types (arrays, objects, and resources) and scalar types are not allowed.

If the parameters could be obtained successfully and there were no errors during type conversion, the function will return SUCCESS, otherwise it will return FAILURE. The function will output informative error messages, if the number of received parameters does not match the requested number, or if type conversion could not be performed.

Here are some sample error messages:

```
Warning - ini_get_all() requires at most 1 parameter, 2 given
     Warning - wddx_deserialize() expects parameter 1 to be string, array given
```

Of course each error message is accompanied by the filename and line number on which it occurs.

Here is the full list of type specifiers:

- `l` - long
- `d` - double
- `s` - string (with possible null bytes) and its length
- `b` - boolean
- `r` - resource, stored in `zval*`
- `a` - array, stored in `zval*`
- `o` - object (of any class), stored in `zval*`
- `O` - object (of class specified by class entry), stored in `zval*`
- `z` - the actual `zval*`

The following characters also have a meaning in the specifier string:

- `|` - indicates that the remaining parameters are optional. The storage variables corresponding to these parameters should be initialized to default values by the extension, since they will not be touched by the parsing function if the parameters are not passed.
- `/` - the parsing function will call **SEPARATE_ZVAL_IF_NOT_REF()** on the parameter it follows, to provide a copy of the parameter, unless it's a reference.
- `!` - the parameter it follows can be of specified type or NULL (only applies to a, o, O, r, and z). If NULL value is passed by the user, the storage pointer will be set to NULL.

The best way to illustrate the usage of this function is through examples:

```
/* Gets a long, a string and its length, and a zval. */
long l;
char *s;
int s_len;
zval *param;
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC,
                          "lsz", &l, &s, &s_len, &param) == FAILURE) {
    return;
}

/* Gets an object of class specified by my_ce, and an optional double. */
zval *obj;
double d = 0.5;
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC,
                          "O|d", &obj, my_ce, &d) == FAILURE) {
    return;
}

/* Gets an object or null, and an array.
   If null is passed for object, obj will be set to NULL. */
zval *obj;
zval *arr;
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "O!a", &obj, &arr) == FAILURE) {
    return;
}

/* Gets a separated array. */
zval *arr;
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "a/", &arr) == FAILURE) {
    return;
}

/* Get only the first three parameters (useful for varargs functions). */
zval *z;
zend_bool b;
zval *r;
if (zend_parse_parameters(3, "zbr!", &z, &b, &r) == FAILURE) {
    return;
```

```
}
```

Note that in the last example we pass 3 for the number of received parameters, instead of **ZEND_NUM_ARGS()**. What this lets us do is receive the least number of parameters if our function expects a variable number of them. Of course, if you want to operate on the rest of the parameters, you will have to use **zend_get_parameters_array_ex()** to obtain them.

The parsing function has an extended version that allows for an additional flags argument that controls its actions.

```
int zend_parse_parameters_ex(int flags, int num_args TSRMLS_DC, char *type_spec, ...);
```

The only flag you can pass currently is `ZEND_PARSE_PARAMS_QUIET`, which instructs the function to not output any error messages during its operation. This is useful for functions that expect several sets of completely different arguments, but you will have to output your own error messages.

For example, here is how you would get either a set of three longs or a string:

```
long l1, l2, l3;
char *s;
if (zend_parse_parameters_ex(ZEND_PARSE_PARAMS_QUIET,
                             ZEND_NUM_ARGS() TSRMLS_CC,
                             "lll", &l1, &l2, &l3) == SUCCESS) {
    /* manipulate longs */
} else if (zend_parse_parameters_ex(ZEND_PARSE_PARAMS_QUIET,
                                    ZEND_NUM_ARGS(), "s", &s, &s_len) == SUCCESS) {
    /* manipulate string */
} else {
    php_error(E_WARNING, "%s() takes either three long values or a string as argument",
              get_active_function_name(TSRMLS_C));
    return;
}
```

With all the abovementioned ways of receiving function parameters you should have a good handle on this process. For even more example, look through the source code for extensions that are shipped with PHP - they illustrate every conceivable situation.

Old way of retrieving arguments (deprecated)

> **Deprecated parameter parsing API:** This API is deprecated and superseded by the new ZEND parameter parsing API.

After having checked the number of arguments, you need to get access to the arguments themselves. This is done with the help of **zend_get_parameters_ex()**:

```
zval **parameter;

if(zend_get_parameters_ex(1, &parameter) != SUCCESS)
  WRONG_PARAM_COUNT;
```

All arguments are stored in a `zval` container, which needs to be pointed to *twice*. The snippet above tries to retrieve one argument and make it available to us via the `parameter` pointer.

zend_get_parameters_ex() accepts at least two arguments. The first argument is the number of arguments to retrieve (which should match the number of arguments with which the function has been called; this is why it's important to check for correct call syntax). The second argument (and all following arguments) are pointers to pointers to pointers to `zval`s. (Confusing, isn't it?) All these pointers are required because Zend works internally with `**zval`; to adjust a local `**zval` in our function,**zend_get_parameters_ex()** requires a pointer to it.

The return value of **zend_get_parameters_ex()** can either be `SUCCESS` or `FAILURE`, indicating (unsurprisingly) success or failure of the argument processing. A failure is most likely related to an incorrect number of arguments being specified, in which case you should exit with `WRONG_PARAM_COUNT`.

To retrieve more than one argument, you can use a similar snippet:

```
zval **param1, **param2, **param3, **param4;

if(zend_get_parameters_ex(4, &param1, &param2, &param3, &param4) != SUCCESS)
    WRONG_PARAM_COUNT;
```

zend_get_parameters_ex() only checks whether you're trying to retrieve too many parameters. If the function is called with five arguments, but you're only retrieving three of them with **zend_get_parameters_ex()**, you won't get an error but will get the first three parameters instead. Subsequent calls of **zend_get_parameters_ex()** won't retrieve the remaining arguments, but will get the same arguments again.

Dealing with a Variable Number of Arguments/Optional

Parameters

If your function is meant to accept a variable number of arguments, the snippets just described are sometimes suboptimal solutions. You have to create a line calling **zend_get_parameters_ex()** for every possible number of arguments, which is often unsatisfying.

For this case, you can use the function **zend_get_parameters_array_ex()**, which accepts the number of parameters to retrieve and an array in which to store them:

```
zval **parameter_array[4];

/* get the number of arguments */
argument_count = ZEND_NUM_ARGS();

/* see if it satisfies our minimal request (2 arguments) */
/* and our maximal acceptance (4 arguments) */
if(argument_count < 2 || argument_count > 5)
    WRONG_PARAM_COUNT;

/* argument count is correct, now retrieve arguments */
if(zend_get_parameters_array_ex(argument_count, parameter_array) != SUCCESS)
    WRONG_PARAM_COUNT;
```

First, the number of arguments is checked to make sure that it's in the accepted range. After that, **zend_get_parameters_array_ex()** is used to fill `parameter_array` with valid pointers to the argument values.

A very clever implementation of this can be found in the code handling PHP's **fsockopen()** located in `ext/standard/fsock.c`, as shown in Example 32-1. Don't worry if you don't know all the functions used in this source yet; we'll get to them shortly.

Example 32-1. PHP's implementation of variable arguments in fsockopen().

```
pval **args[5];
int *sock=emalloc(sizeof(int));
int *sockp;
int arg_count=ARG_COUNT(ht);
int socketd = -1;
unsigned char udp = 0;
struct timeval timeout = { 60, 0 };
unsigned short portno;
unsigned long conv;
char *key = NULL;
FLS_FETCH();

if (arg_count > 5 || arg_count < 2 || zend_get_parameters_array_ex(arg_count,args)==FAILURE) {
    CLOSE_SOCK(1);
    WRONG_PARAM_COUNT;
}

switch(arg_count) {
    case 5:
        convert_to_double_ex(args[4]);
        conv = (unsigned long) (Z_DVAL_P(args[4]) * 1000000.0);
        timeout.tv_sec = conv / 1000000;
        timeout.tv_usec = conv % 1000000;
        /* fall-through */
    case 4:
        if (!PZVAL_IS_REF(*args[3])) {
            php_error(E_WARNING,"error string argument to fsockopen not passed by reference");
        }
        pval_copy_constructor(*args[3]);
        ZVAL_EMPTY_STRING(*args[3]);
        /* fall-through */
    case 3:
        if (!PZVAL_IS_REF(*args[2])) {
            php_error(E_WARNING,"error argument to fsockopen not passed by reference");
            return;
        }
        ZVAL_LONG(*args[2], 0);
        break;
}

convert_to_string_ex(args[0]);
convert_to_long_ex(args[1]);
portno = (unsigned short) Z_LVAL_P(args[1]);

key = emalloc(Z_STRLEN_P(args[0]) + 10);
```

fsockopen() accepts two, three, four, or five parameters. After the obligatory variable declarations, the function checks for the correct range of arguments. Then it uses a fall-through mechanism in a `switch()` statement to deal with all arguments. The `switch()` statement starts with the maximum number of arguments being passed (five). After that, it automatically processes the case of four arguments being passed, then three, by omitting the otherwise obligatory `break` keyword in all stages. After having processed the last case, it exits the `switch()` statement and does the minimal argument processing needed if the function is invoked with only two arguments.

This multiple-stage type of processing, similar to a stairway, allows convenient processing of a variable number of arguments.

Accessing Arguments

To access arguments, it's necessary for each argument to have a clearly defined type. Again, PHP's extremely dynamic nature introduces some quirks. Because PHP never does any kind of type checking, it's possible for a caller to pass any kind of data to your functions, whether you want it or not. If you expect an integer, for example, the caller might pass an array, and vice versa - PHP simply won't notice.

To work around this, you have to use a set of API functions to force a type conversion on every argument that's being passed (see Table 32-1).

Note: All conversion functions expect a `**zval` as parameter.

Table 32-1. Argument Conversion Functions

Function	Description
convert_to_boolean_ex()	Forces conversion to a Boolean type. Boolean values remain untouched. Longs, doubles, and strings containing `0` as well as NULL values will result in Boolean `0` (FALSE). Arrays and objects are converted based on the number of entries or properties, respectively, that they have. Empty arrays and objects are converted to FALSE; otherwise, to TRUE. All other values result in a Boolean `1` (TRUE).
convert_to_long_ex()	Forces conversion to a long, the default integer type. NULL values, Booleans, resources, and of course longs remain untouched. Doubles are truncated. Strings containing an integer are converted to their corresponding numeric representation, otherwise resulting in `0`. Arrays and objects are converted to `0` if empty, `1` otherwise.
convert_to_double_ex()	Forces conversion to a double, the default floating-point type. NULL values, Booleans, resources, longs, and of course doubles remain untouched. Strings containing a number are converted to their corresponding numeric representation, otherwise resulting in `0.0`. Arrays and objects are converted to `0.0` if empty, `1.0` otherwise.
convert_to_string_ex()	Forces conversion to a string. Strings remain untouched. NULL values are converted to an empty string. Booleans containing TRUE are converted to `"1"`, otherwise resulting in an empty string. Longs and doubles are converted to their corresponding string representation. Arrays are converted to the string `"Array"` and objects to the string `"Object"`.
`convert_to_array_ex(value)`	Forces conversion to an array. Arrays remain untouched. Objects are converted to an array by assigning all their properties to the array table. All property names are used as keys, property contents as values. NULL values are converted to an empty array. All other values are converted to an array that contains the specific source value in the element with the key `0`.
`convert_to_object_ex(value)`	Forces conversion to an object. Objects remain untouched. NULL values are converted to an empty object. Arrays are converted to objects by introducing their keys as properties into the objects and their values as corresponding property contents in the object. All other types result in an object with the property `scalar` , having the corresponding source value as content.
`convert_to_null_ex(value)`	Forces the type to become a NULL value, meaning empty.

> **Note:** You can find a demonstration of the behavior in `cross_conversion.php` on the accompanying CD-ROM. Figure 32-2 shows the output.

Figure 32-2. Cross-conversion behavior of PHP.

Using these functions on your arguments will ensure type safety for all data that's passed to you. If the supplied type doesn't match the required type, PHP forces dummy contents on the resulting value (empty strings, arrays, or objects, `0` for numeric

values, FALSE for Booleans) to ensure a defined state.

Following is a quote from the sample module discussed previously, which makes use of the conversion functions:

```
zval **parameter;

if((ZEND_NUM_ARGS() != 1) || (zend_get_parameters_ex(1, &parameter) != SUCCESS))
{
    WRONG_PARAM_COUNT;
}

convert_to_long_ex(parameter);

RETURN_LONG(Z_LVAL_P(parameter));
```

After retrieving the parameter pointer, the parameter value is converted to a long (an integer), which also forms the return value of this function. Understanding access to the contents of the value requires a short discussion of the `zval` type, whose definition is shown in Example 32-2.

Example 32-2. PHP/Zend `zval` type definition.

```
typedef pval zval;

typedef struct _zval_struct zval;

typedef union _zvalue_value {
        long lval;                                      /* long value */
        double dval;                                /* double value */
        struct {
                char *val;
                int len;
        } str;
        HashTable *ht;                              /* hash table value */
        struct {
                zend_class_entry *ce;
                HashTable *properties;
        } obj;
} zvalue_value;

struct _zval_struct {
        /* Variable information */
        zvalue_value value;             /* value */
        unsigned char type;     /* active type */
        unsigned char is_ref;
        short refcount;
};
```

Actually, `pval` (defined in `php.h`) is only an alias of `zval` (defined in `zend.h`), which in turn refers to `_zval_struct`. This is a most interesting structure. `_zval_struct` is the "master" structure, containing the value structure, type, and reference information. The substructure `zvalue_value` is a union that contains the variable's contents. Depending on the variable's type, you'll have to access different members of this union. For a description of both structures, see Table 32-2, Table 32-3 and Table 32-4.

Table 32-2. Zend `zval` Structure

Entry	Description
`value`	Union containing this variable's contents. See Table 32-3 for a description.
`type`	Contains this variable's type. For a list of available types, see Table 32-4.
`is_ref`	0 means that this variable is not a reference; 1 means that this variable is a reference to another variable.
`refcount`	The number of references that exist for this variable. For every new reference to the value stored in this variable, this counter is increased by 1. For every lost reference, this counter is decreased by 1. When the reference counter reaches 0, no references exist to this value anymore, which causes automatic freeing of the value.

Table 32-3. Zend `zvalue_value` Structure

Entry	Description
`lval`	Use this property if the variable is of the type `IS_LONG`, `IS_BOOLEAN`, or `IS_RESOURCE`.
`dval`	Use this property if the variable is of the type `IS_DOUBLE`.
`str`	This structure can be used to access variables of the type `IS_STRING`. The member `len` contains the string length; the member `val` points to the string itself. Zend uses C strings; thus, the string length contains a trailing `0x00`.
`ht`	This entry points to the variable's hash table entry if the variable is an array.
`obj`	Use this property if the variable is of the type `IS_OBJECT`.

Table 32-4. Zend Variable Type Constants

Constant	Description
IS_NULL	Denotes a NULL (empty) value.
IS_LONG	A long (integer) value.
IS_DOUBLE	A double (floating point) value.
IS_STRING	A string.
IS_ARRAY	Denotes an array.
IS_OBJECT	An object.
IS_BOOL	A Boolean value.
IS_RESOURCE	A resource (for a discussion of resources, see the appropriate section below).
IS_CONSTANT	A constant (defined) value.

To access a long you access `zval.value.lval`, to access a double you use `zval.value.dval`, and so on. Because all values are stored in a union, trying to access data with incorrect union members results in meaningless output.

Accessing arrays and objects is a bit more complicated and is discussed later.

Dealing with Arguments Passed by Reference

If your function accepts arguments passed by reference that you intend to modify, you need to take some precautions.

What we didn't say yet is that under the circumstances presented so far, you don't have write access to any `zval` containers designating function parameters that have been passed to you. Of course, you can change any `zval` containers that you created within your function, but you mustn't change any `zval`s that refer to Zend-internal data!

We've only discussed the so-called ***_ex()** API so far. You may have noticed that the API functions we've used are called **zend_get_parameters_ex()** instead of **zend_get_parameters()**, **convert_to_long_ex()** instead of **convert_to_long()**, etc. The ***_ex()** functions form the so-called new "extended" Zend API. They give a minor speed increase over the old API, but as a tradeoff are only meant for providing read-only access.

Because Zend works internally with references, different variables may reference the same value. Write access to a `zval` container requires this container to contain an isolated value, meaning a value that's not referenced by any other containers. If a `zval` container were referenced by other containers and you changed the referenced `zval`, you would automatically change the contents of the other containers referencing this `zval` (because they'd simply point to the changed value and thus change their own value as well).

zend_get_parameters_ex() doesn't care about this situation, but simply returns a pointer to the desired `zval` containers, whether they consist of references or not. Its corresponding function in the traditional API, **zend_get_parameters()**, immediately checks for referenced values. If it finds a reference, it creates a new, isolated `zval` container; copies the referenced data into this newly allocated space; and then returns a pointer to the new, isolated value.

This action is called *zval separation* (or pval separation). Because the ***_ex()** API doesn't perform zval separation, it's considerably faster, while at the same time disabling write access.

To change parameters, however, write access is required. Zend deals with this situation in a special way: Whenever a parameter to a function is passed by reference, it performs automatic zval separation. This means that whenever you're calling a function like this in PHP, Zend will automatically ensure that `$parameter` is being passed as an isolated value, rendering it to a write-safe state:

```
my_function(&$parameter);
```

But this *is not* the case with regular parameters! All other parameters that are not passed by reference are in a read-only state.

This requires you to make sure that you're really working with a reference - otherwise you might produce unwanted results. To check for a parameter being passed by reference, you can use the macro `PZVAL_IS_REF`. This macro accepts a `zval*` to check if it is a reference or not. Examples are given in in Example 32-3.

Example 32-3. Testing for referenced parameter passing.

```
zval *parameter;

if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "z", &parameter) == FAILURE)
    return;

/* check for parameter being passed by reference */
```

```
if (!PZVAL_IS_REF(*parameter)) {
{
    zend_error(E_WARNING, "Parameter wasn't passed by reference");
    RETURN_NULL();
}

/* make changes to the parameter */
ZVAL_LONG(*parameter, 10);
```

Assuring Write Safety for Other Parameters

You might run into a situation in which you need write access to a parameter that's retrieved with **zend_get_parameters_ex()** but not passed by reference. For this case, you can use the macro SEPARATE_ZVAL, which does a zval separation on the provided container. The newly generated zval is detached from internal data and has only a local scope, meaning that it can be changed or destroyed without implying global changes in the script context:

```
zval **parameter;

/* retrieve parameter */
zend_get_parameters_ex(1, &parameter);

/* at this stage, <parameter> still is connected */
/* to Zend's internal data buffers */

/* make <parameter> write-safe */
SEPARATE_ZVAL(parameter);

/* now we can safely modify <parameter> */
/* without implying global changes */
```

SEPARATE_ZVAL uses **emalloc()** to allocate the new zval container, which means that even if you don't deallocate this memory yourself, it will be destroyed automatically upon script termination. However, doing a lot of calls to this macro without freeing the resulting containers will clutter up your RAM.

Note: As you can easily work around the lack of write access in the "traditional" API (with **zend_get_parameters()** and so on), this API seems to be obsolete, and is not discussed further in this chapter.

Chapter 33. Creating Variables

When exchanging data from your own extensions with PHP scripts, one of the most important issues is the creation of variables. This section shows you how to deal with the variable types that PHP supports.

Overview

To create new variables that can be seen "from the outside" by the executing script, you need to allocate a new zval container, fill this container with meaningful values, and then introduce it to Zend's internal symbol table. This basic process is common to all variable creations:

```
zval *new_variable;

/* allocate and initialize new container */
MAKE_STD_ZVAL(new_variable);

/* set type and variable contents here, see the following sections */

/* introduce this variable by the name "new_variable_name" into the symbol table */
ZEND_SET_SYMBOL(EG(active_symbol_table), "new_variable_name", new_variable);

/* the variable is now accessible to the script by using $new_variable_name */
```

The macro MAKE_STD_ZVAL allocates a new zval container using ALLOC_ZVAL and initializes it using INIT_ZVAL. As implemented in Zend at the time of this writing, *initializing* means setting the reference count to 1 and clearing the is_ref flag, but this process could be extended later - this is why it's a good idea to keep using MAKE_STD_ZVAL instead of only using ALLOC_ZVAL. If you want to optimize for speed (and you don't have to explicitly initialize the zval container here), you can use ALLOC_ZVAL, but this isn't recommended because it doesn't ensure data integrity.

ZEND_SET_SYMBOL takes care of introducing the new variable to Zend's symbol table. This macro checks whether the value already exists in the symbol table and converts the new symbol to a reference if so (with automatic deallocation of the old zval

container). This is the preferred method if speed is not a crucial issue and you'd like to keep memory usage low.

Note that `ZEND_SET_SYMBOL` makes use of the Zend executor globals via the macro `EG`. By specifying `EG(active_symbol_table)`, you get access to the currently active symbol table, dealing with the active, local scope. The local scope may differ depending on whether the function was invoked from within a function.

If you need to optimize for speed and don't care about optimal memory usage, you can omit the check for an existing variable with the same value and instead force insertion into the symbol table by using **zend_hash_update()**:

```
zval *new_variable;

/* allocate and initialize new container */
MAKE_STD_ZVAL(new_variable);

/* set type and variable contents here, see the following sections */

/* introduce this variable by the name "new_variable_name" into the symbol table */
zend_hash_update(
    EG(active_symbol_table),
    "new_variable_name",
    strlen("new_variable_name") + 1,
    &new_variable,
    sizeof(zval *),
    NULL
);
```

This is actually the standard method used in most modules.

The variables generated with the snippet above will always be of local scope, so they reside in the context in which the function has been called. To create new variables in the global scope, use the same method but refer to another symbol table:

```
zval *new_variable;

// allocate and initialize new container
MAKE_STD_ZVAL(new_variable);

//
// set type and variable contents here
//

// introduce this variable by the name "new_variable_name" into the global symbol table
ZEND_SET_SYMBOL(&EG(symbol_table), "new_variable_name", new_variable);
```

The macro `ZEND_SET_SYMBOL` is now being called with a reference to the main, global symbol table by referring `EG(symbol_table)`.

Note: The `active_symbol_table` variable is a pointer, but `symbol_table` is not. This is why you have to use `EG(active_symbol_table)` and `&EG(symbol_table)` as parameters to `ZEND_SET_SYMBOL` - it requires a pointer.

Similarly, to get a more efficient version, you can hardcode the symbol table update:

```
zval *new_variable;

// allocate and initialize new container
MAKE_STD_ZVAL(new_variable);

//
// set type and variable contents here
//

// introduce this variable by the name "new_variable_name" into the global symbol table
zend_hash_update(
    &EG(symbol_table),
    "new_variable_name",
    strlen("new_variable_name") + 1,
    &new_variable,
    sizeof(zval *),
    NULL
);
```

Example 33-1 shows a sample source that creates two variables - `local_variable` with a local scope and `global_variable` with a global scope (see Figure 9.7). The full example can be found on the CD-ROM.

Note: You can see that the global variable is actually not accessible from within the function. This is because it's not imported into the local scope using `global $global_variable;` in the PHP source.

Example 33-1. Creating variables with different scopes.

```
ZEND_FUNCTION(variable_creation)
{
    zval *new_var1, *new_var2;

    MAKE_STD_ZVAL(new_var1);
    MAKE_STD_ZVAL(new_var2);

    ZVAL_LONG(new_var1, 10);
    ZVAL_LONG(new_var2, 5);

    ZEND_SET_SYMBOL(EG(active_symbol_table), "local_variable", new_var1);
    ZEND_SET_SYMBOL(&EG(symbol_table), "global_variable", new_var2);
```

```
    RETURN_NULL();
}
```

Longs (Integers)

Now let's get to the assignment of data to variables, starting with longs. Longs are PHP's integers and are very simple to store. Looking at the `zval.value` container structure discussed earlier in this chapter, you can see that the long data type is directly contained in the union, namely in the `lval` field. The corresponding `type` value for longs is `IS_LONG` (see Example 33-2).

Example 33-2. Creation of a long.

```
zval *new_long;

MAKE_STD_ZVAL(new_long);

new_long->type = IS_LONG;
new_long->value.lval = 10;
```

Alternatively, you can use the macro `ZVAL_LONG`:

```
zval *new_long;

MAKE_STD_ZVAL(new_long);
ZVAL_LONG(new_long, 10);
```

Doubles (Floats)

Doubles are PHP's floats and are as easy to assign as longs, because their value is also contained directly in the union. The member in the `zval.value` container is `dval`; the corresponding type is `IS_DOUBLE`.

```
zval *new_double;

MAKE_STD_ZVAL(new_double);

new_double->type = IS_DOUBLE;
new_double->value.dval = 3.45;
```

Alternatively, you can use the macro `ZVAL_DOUBLE`:

```
zval *new_double;

MAKE_STD_ZVAL(new_double);
ZVAL_DOUBLE(new_double, 3.45);
```

Strings

Strings need slightly more effort. As mentioned earlier, all strings that will be associated with Zend's internal data structures need to be allocated using Zend's own memory-management functions. Referencing of static strings or strings allocated with standard routines is not allowed. To assign strings, you have to access the structure `str` in the `zval.value` container. The corresponding type is `IS_STRING`:

```
zval *new_string;
char *string_contents = "This is a new string variable";

MAKE_STD_ZVAL(new_string);

new_string->type = IS_STRING;
new_string->value.str.len = strlen(string_contents);
new_string->value.str.val = estrdup(string_contents);
```

Note the usage of Zend's **estrdup()** here. Of course, you can also use the predefined macro `ZVAL_STRING`:

```
zval *new_string;
char *string_contents = "This is a new string variable";

MAKE_STD_ZVAL(new_string);
ZVAL_STRING(new_string, string_contents, 1);
```

`ZVAL_STRING` accepts a third parameter that indicates whether the supplied string contents should be duplicated (using **estrdup()**). Setting this parameter to `1` causes the string to be duplicated; `0` simply uses the supplied pointer for the variable contents. This is most useful if you want to create a new variable referring to a string that's already allocated in Zend internal

memory.

If you want to truncate the string at a certain position or you already know its length, you can use `ZVAL_STRINGL(zval, string, length, duplicate)`, which accepts an explicit string length to be set for the new string. This macro is faster than `ZVAL_STRING` and also binary-safe.

To create empty strings, set the string length to `0` and use `empty_string` as contents:

```
new_string->type = IS_STRING;
new_string->value.str.len = 0;
new_string->value.str.val = empty_string;
```

Of course, there's a macro for this as well (`ZVAL_EMPTY_STRING`):

```
MAKE_STD_ZVAL(new_string);
ZVAL_EMPTY_STRING(new_string);
```

Booleans

Booleans are created just like longs, but have the type `IS_BOOL`. Allowed values in `lval` are `0` and `1`:

```
zval *new_bool;

MAKE_STD_ZVAL(new_bool);

new_bool->type = IS_BOOL;
new_bool->value.lval = 1;
```

The corresponding macros for this type are `ZVAL_BOOL` (allowing specification of the value) as well as `ZVAL_TRUE` and `ZVAL_FALSE` (which explicitly set the value to `TRUE` and `FALSE`, respectively).

Arrays

Arrays are stored using Zend's internal hash tables, which can be accessed using the **zend_hash_*()** API. For every array that you want to create, you need a new hash table handle, which will be stored in the `ht` member of the `zval.value` container.

There's a whole API solely for the creation of arrays, which is extremely handy. To start a new array, you call **array_init()**.

```
zval *new_array;

MAKE_STD_ZVAL(new_array);

array_init(new_array);
```

array_init() always returns `SUCCESS`.

To add new elements to the array, you can use numerous functions, depending on what you want to do. Table 33-1, Table 33-2 and Table 33-3 describe these functions. All functions return `FAILURE` on failure and `SUCCESS` on success.

Table 33-1. Zend's API for Associative Arrays

Function	Description
add_assoc_long(zval *array, char *key, long n);()	Adds an element of type `long`.
add_assoc_unset(zval *array, char *key);()	Adds an unset element.
add_assoc_bool(zval *array, char *key, int b);()	Adds a Boolean element.
add_assoc_resource(zval *array, char *key, int r);()	Adds a resource to the array.
add_assoc_double(zval *array, char *key, double d);()	Adds a floating-point value.
add_assoc_string(zval *array, char *key, char *str, int duplicate);()	Adds a string to the array. The flag `duplicate` specifies whether the string contents have to be copied to Zend internal memory.
add_assoc_stringl(zval *array, char *key, char *str, uint length, int duplicate); ()	Adds a string with the desired length `length` to the array. Otherwise, behaves like **add_assoc_string()**.
add_assoc_zval(zval *array, char *key, zval *value);()	Adds a zval to the array. Useful for adding other arrays, objects, streams, etc...

Table 33-2. Zend's API for Indexed Arrays, Part 1

Function	Description
add_index_long(zval *array, uint idx, long n);()	Adds an element of type `long`.

add_index_unset(zval *array, uint idx);()	Adds an unset element.
add_index_bool(zval *array, uint idx, int b);()	Adds a Boolean element.
add_index_resource(zval *array, uint idx, int r);()	Adds a resource to the array.
add_index_double(zval *array, uint idx, double d);()	Adds a floating-point value.
add_index_string(zval *array, uint idx, char *str, int duplicate);()	Adds a string to the array. The flag `duplicate` specifies whether the string contents have to be copied to Zend internal memory.
add_index_stringl(zval *array, uint idx, char *str, uint length, int duplicate);()	Adds a string with the desired length `length` to the array. This function is faster and binary-safe. Otherwise, behaves like **add_index_string()()**.
add_index_zval(zval *array, uint idx, zval *value);()	Adds a zval to the array. Useful for adding other arrays, objects, streams, etc...

Table 33-3. Zend's API for Indexed Arrays, Part 2

Function	Description
add_next_index_long(zval *array, long n);()	Adds an element of type `long`.
add_next_index_unset(zval *array);()	Adds an unset element.
add_next_index_bool(zval *array, int b);()	Adds a Boolean element.
add_next_index_resource(zval *array, int r);()	Adds a resource to the array.
add_next_index_double(zval *array, double d);()	Adds a floating-point value.
add_next_index_string(zval *array, char *str, int duplicate);()	Adds a string to the array. The flag `duplicate` specifies whether the string contents have to be copied to Zend internal memory.
add_next_index_stringl(zval *array, char *str, uint length, int duplicate);()	Adds a string with the desired length `length` to the array. This function is faster and binary-safe. Otherwise, behaves like **add_index_string()()**.
add_next_index_zval(zval *array, zval *value);()	Adds a zval to the array. Useful for adding other arrays, objects, streams, etc...

All these functions provide a handy abstraction to Zend's internal hash API. Of course, you can also use the hash functions directly - for example, if you already have a `zval` container allocated that you want to insert into an array. This is done using **zend_hash_update()()** for associative arrays (see Example 33-3) and **zend_hash_index_update()** for indexed arrays (see Example 33-4):

Example 33-3. Adding an element to an associative array.

```
zval *new_array, *new_element;
char *key = "element_key";

MAKE_STD_ZVAL(new_array);
MAKE_STD_ZVAL(new_element);

array_init(new_array);

ZVAL_LONG(new_element, 10);

if(zend_hash_update(new_array->value.ht, key, strlen(key) + 1, (void *)&new_element, sizeof(zval *), NULL) == FAILURE)
{
    // do error handling here
}
```

Example 33-4. Adding an element to an indexed array.

```
zval *new_array, *new_element;
int key = 2;

MAKE_STD_ZVAL(new_array);
MAKE_STD_ZVAL(new_element);

array_init(new_array);

ZVAL_LONG(new_element, 10);

if(zend_hash_index_update(new_array->value.ht, key, (void *)&new_element, sizeof(zval *), NULL) == FAILURE)
{
    // do error handling here
}
```

To emulate the functionality of **add_next_index_*()**, you can use this:

```
zend_hash_next_index_insert(ht, zval **new_element, sizeof(zval *), NULL)
```

Note: To return arrays from a function, use **array_init()** and all following actions on the predefined variable `return_value` (given as argument to your exported function; see the earlier discussion of the call interface). You do not have to use `MAKE_STD_ZVAL` on this.

Tip: To avoid having to write `new_array->value.ht` every time, you can use `HASH_OF(new_array)`, which is also recommended for compatibility and style reasons.

Objects

Since objects can be converted to arrays (and vice versa), you might have already guessed that they have a lot of similarities to arrays in PHP. Objects are maintained with the same hash functions, but there's a different API for creating them.

To initialize an object, you use the function **object_init()**:

```
zval *new_object;

MAKE_STD_ZVAL(new_object);

if(object_init(new_object) != SUCCESS)
{
    // do error handling here
}
```

You can use the functions described in Table 33-4 to add members to your object.

Table 33-4. Zend's API for Object Creation

Function	Description
add_property_long(zval *object, char *key, long l);()	Adds a long to the object.
add_property_unset(zval *object, char *key);()	Adds an unset property to the object.
add_property_bool(zval *object, char *key, int b);()	Adds a Boolean to the object.
add_property_resource(zval *object, char *key, long r);()	Adds a resource to the object.
add_property_double(zval *object, char *key, double d);()	Adds a double to the object.
add_property_string(zval *object, char *key, char *str, int duplicate);()	Adds a string to the object.
add_property_stringl(zval *object, char *key, char *str, uint length, int duplicate);()	Adds a string of the specified length to the object. This function is faster than **add_property_string()** and also binary-safe.
add_property_zval(zval *obect, char *key, zval *container):()	Adds a `zval` container to the object. This is useful if you have to add properties which aren't simple types like integers or strings but arrays or other objects.

Resources

Resources are a special kind of data type in PHP. The term *resources* doesn't really refer to any special kind of data, but to an abstraction method for maintaining any kind of information. Resources are kept in a special resource list within Zend. Each entry in the list has a correspondending type definition that denotes the kind of resource to which it refers. Zend then internally manages all references to this resource. Access to a resource is never possible directly - only via a provided API. As soon as all references to a specific resource are lost, a corresponding shutdown function is called.

For example, resources are used to store database links and file descriptors. The *de facto* standard implementation can be found in the MySQL module, but other modules such as the Oracle module also make use of resources.

> **Note:** In fact, a resource can be a pointer to anything you need to handle in your functions (e.g. pointer to a structure) and the user only has to pass a single resource variable to your function.

To create a new resource you need to register a resource destruction handler for it. Since you can store any kind of data as a resource, Zend needs to know how to free this resource if its not longer needed. This works by registering your own resource destruction handler to Zend which in turn gets called by Zend whenever your resource can be freed (whether manually or automatically). Registering your resource handler within Zend returns you the **resource type handle** for that resource. This handle is needed whenever you want to access a resource of this type later and is most of time stored in a global static variable within your extension. There is no need to worry about thread safety here because you only register your resource handler once during module initialization.

The Zend function to register your resource handler is defined as:

```
ZEND_API int zend_register_list_destructors_ex(rsrc_dtor_func_t ld, rsrc_dtor_func_t pld, char *type_name, int module_number
```

There are two different kinds of resource destruction handlers you can pass to this function: a handler for normal resources and a handler for persistent resources. Persistent resources are for example used for database connection. When registering a resource, either of these handlers must be given. For the other handler just pass `NULL`.

zend_register_list_destructors_ex() accepts the following parameters:

`ld`	Normal resource destruction handler callback
`pld`	Pesistent resource destruction handler callback
`type_name`	A string specifying the name of your resource. It's always a good thing to specify an unique name within PHP for the resource type so when the user for example calls `var_dump($resource);` he also gets the name of the resource.
`module_number`	The `module_number` is automatically available in your `PHP_MINIT_FUNCTION` function and therefore you just pass it over.

The return value is an unique integer ID for your **resource type**.

The resource destruction handler (either normal or persistent resources) has the following prototype:

```
void resource_destruction_handler(zend_rsrc_list_entry *rsrc TSRMLS_DC);
```

The passed `rsrc` is a pointer to the following structure:

```
typedef struct _zend_rsrc_list_entry {

    void *ptr;
    int type;
    int refcount;

} zend_rsrc_list_entry;
```

The member `void *ptr` is the actual pointer to your resource.

Now we know how to start things, we define our own resource we want register within Zend. It is only a simple structure with two integer members:

```
typedef struct {

    int resource_link;
    int resource_type;

} my_resource;
```

Our resource destruction handler is probably going to look something like this:

```
void my_destruction_handler(zend_rsrc_list_entry *rsrc TSRMLS_DC) {

    // You most likely cast the void pointer to your structure type

    my_resource *my_rsrc = (my_resource *) rsrc->ptr;

    // Now do whatever needs to be done with you resource. Closing
    // Files, Sockets, freeing additional memory, etc.
    // Also, don't forget to actually free the memory for your resource too!

    do_whatever_needs_to_be_done_with_the_resource(my_rsrc);
}
```

> **Note:** One important thing to mention: If your resource is a rather complex structure which also contains pointers to memory you allocated during runtime you have to free them **before** freeing the resource itself!

Now that we have defined

1. what our resource is and
2. our resource destruction handler

we can go on and do the rest of the steps:

1. create a global variable within the extension holding the resource ID so it can be accessed from every function which needs it
2. define the resource name
3. write the resource destruction handler
4. and finally register the handler

```
// Somewhere in your extension, define the variable for your registered resources.
    // If you wondered what 'le' stands for: it simply means 'list entry'.
```

```
static int le_myresource;

// It's nice to define your resource name somewhere
#define le_myresource_name  "My type of resource"

[...]

// Now actually define our resource destruction handler
void my_destruction_handler(zend_rsrc_list_entry *rsrc TSRMLS_DC) {

    my_resource *my_rsrc = (my_resource *) rsrc->ptr;
    do_whatever_needs_to_be_done_with_the_resource(my_rsrc);
}

[...]

PHP_MINIT_FUNCTION(my_extension) {

    // Note that 'module_number' is already provided through the
    // PHP_MINIT_FUNCTION() function definition.

    le_myresource = zend_register_resource_destructors_ex(my_destruction_handler, NULL, le_myresource_name, module_numbe

    // You can register additional resources, initialize
    // your global vars, constants, whatever.
}
```

To actually register a new resource you use can either use the **zend_register_resource()** function or the **ZEND_REGISTER_RESOURE()** macro, both defined in zend_list.h . Although the arguments for both map 1:1 it's a good idea to always use macros to be upwards compatible:

```
int ZEND_REGISTER_RESOURCE(zval *rsrc_result, void *rsrc_pointer, int rsrc_type);
```

rsrc_result	This is an already initialized `zval *` container.
rsrc_pointer	Your resource pointer you want to store.
rsrc_type	The type which you received when you registered the resource destruction handler. If you followed the naming scheme this would be `le_myresource`.

The return value is an unique integer identifier for that resource.

What is really going on when you register a new resource is it gets inserted in an internal list in Zend and the result is just stored in the given `zval *` container:

```
rsrc_id = zend_list_insert(rsrc_pointer, rsrc_type);

    if (rsrc_result) {
        rsrc_result->value.lval = rsrc_id;
        rsrc_result->type = IS_RESOURCE;
    }

    return rsrc_id;
```

The returned `rsrc_id` uniquly identifies the newly registered resource. You can use the macro `RETURN_RESOURE` to return it to the user:

```
RETURN_RESOURCE(rsrc_id)
```

> **Note:** It is common practice that if you want to return the resource immidiately to the user you specify the `return_value` as the `zval *` container.

Zend now keeps track of all references to this resource. As soon as all references to the resource are lost, the destructor that you previously registered for this resource is called. The nice thing about this setup is that you don't have to worry about memory leakages introduced by allocations in your module - just register all memory allocations that your calling script will refer to as resources. As soon as the script decides it doesn't need them anymore, Zend will find out and tell you.

Now that the user got his resource, at some point he is passing it back to one of your functions. The `value.lval` inside the `zval *` container contains the key to your resource and thus can be used to fetch the resource with the following macro:

`ZEND_FETCH_RESOURCE`:

```
ZEND_FETCH_RESOURCE(rsrc, rsrc_type, rsrc_id, default_rsrc_id, resource_type_name, resource_type)
```

rsrc	This is your pointer which will point to your previously registered resource.
rsrc_type	This is the typecast argument for your pointer, e.g. `myresource *`.
rsrc_id	This is the address of the `zval *`container the user passed to your function, e.g. `&z_resource` if `zval *z_resource` is given.
default_rsrc_id	This integer specifies the default resource `ID` if no resource could be fetched or -1.
resource_type_name	This is the name of the requested resource. It's a string and is used when the resource can't be found or is invalid to form a meaningful error message.

resource_type	The resource_type you got back when registering the resource destruction handler. In our example this was le_myresource.

This macro has no return value. It is for the developers convenience and takes care of TSRMLS arguments passing and also does check if the resource could be fetched. It throws a warning message and returns the current PHP function with NULL if there was a problem retrieving the resource.

To force removal of a resource from the list, use the function **zend_list_delete()**. You can also force the reference count to increase if you know that you're creating another reference for a previously allocated value (for example, if you're automatically reusing a default database link). For this case, use the function **zend_list_addref()**. To search for previously allocated resource entries, use **zend_list_find()**. The complete API can be found in zend_list.h.

Macros for Automatic Global Variable Creation

In addition to the macros discussed earlier, a few macros allow easy creation of simple global variables. These are nice to know in case you want to introduce global flags, for example. This is somewhat bad practice, but Table Table 33-5 describes macros that do exactly this task. They don't need any zval allocation; you simply have to supply a variable name and value.

Table 33-5. Macros for Global Variable Creation

Macro	Description
SET_VAR_STRING(name, value)	Creates a new string.
SET_VAR_STRINGL(name, value, length)	Creates a new string of the specified length. This macro is faster than SET_VAR_STRING and also binary-safe.
SET_VAR_LONG(name, value)	Creates a new long.
SET_VAR_DOUBLE(name, value)	Creates a new double.

Creating Constants

Zend supports the creation of true constants (as opposed to regular variables). Constants are accessed without the typical dollar sign ($) prefix and are available in all scopes. Examples include TRUE and FALSE, to name just two.

To create your own constants, you can use the macros in Table 33-6. All the macros create a constant with the specified name and value.

You can also specify flags for each constant:

- CONST_CS - This constant's name is to be treated as case sensitive.
- CONST_PERSISTENT - This constant is persistent and won't be "forgotten" when the current process carrying this constant shuts down.

To use the flags, combine them using a inary OR:

```
// register a new constant of type "long"
     REGISTER_LONG_CONSTANT("NEW_MEANINGFUL_CONSTANT", 324, CONST_CS |
     CONST_PERSISTENT);
```

There are two types of macros - REGISTER_*_CONSTANT andREGISTER_MAIN_*_CONSTANT. The first type creates constants bound to the current module. These constants are dumped from the symbol table as soon as the module that registered the constant is unloaded from memory. The second type creates constants that remain in the symbol table independently of the module.

Table 33-6. Macros for Creating Constants

Macro	Description
REGISTER_LONG_CONSTANT(name, value, flags) REGISTER_MAIN_LONG_CONSTANT(name, value, flags)	Registers a new constant of type long.
REGISTER_DOUBLE_CONSTANT(name, value, flags) REGISTER_MAIN_DOUBLE_CONSTANT(name, value, flags)	Registers a new constant of type double.
REGISTER_STRING_CONSTANT(name, value, flags) REGISTER_MAIN_STRING_CONSTANT(name, value, flags)	Registers a new constant of type string. The specified string must reside in Zend's internal memory.

`REGISTER_STRINGL_CONSTANT(name, value, length, flags)` `REGISTER_MAIN_STRINGL_CONSTANT(name, value, length, flags)`	Registers a new constant of type string. The string length is explicitly set to `length`. The specified string must reside in Zend's internal memory.

Chapter 34. Duplicating Variable Contents: The Copy Constructor

Sooner or later, you may need to assign the contents of one `zval` container to another. This is easier said than done, since the `zval` container doesn't contain only type information, but also references to places in Zend's internal data. For example, depending on their size, arrays and objects may be nested with lots of hash table entries. By assigning one `zval` to another, you avoid duplicating the hash table entries, using only a reference to them (at most).

To copy this complex kind of data, use the *copy constructor*. Copy constructors are typically defined in languages that support operator overloading, with the express purpose of copying complex types. If you define an object in such a language, you have the possibility of overloading the "=" operator, which is usually responsible for assigning the contents of the lvalue (result of the evaluation of the left side of the operator) to the rvalue (same for the right side).

Overloading means assigning a different meaning to this operator, and is usually used to assign a function call to an operator. Whenever this operator would be used on such an object in a program, this function would be called with the lvalue and rvalue as parameters. Equipped with that information, it can perform the operation it intends the "=" operator to have (usually an extended form of copying).

This same form of "extended copying" is also necessary for PHP's `zval` containers. Again, in the case of an array, this extended copying would imply re-creation of all hash table entries relating to this array. For strings, proper memory allocation would have to be assured, and so on.

Zend ships with such a function, called **zend_copy_ctor()** (the previous PHP equivalent was **pval_copy_constructor()**).

A most useful demonstration is a function that accepts a complex type as argument, modifies it, and then returns the argument:

```
zval *parameter;

if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "z", &parameter) == FAILURE)
   return;
}

// do modifications to the parameter here

// now we want to return the modified container:
*return_value == *parameter;
zval_copy_ctor(return_value);
```

The first part of the function is plain-vanilla argument retrieval. After the (left out) modifications, however, it gets interesting: The container of `parameter` is assigned to the (predefined) `return_value` container. Now, in order to effectively duplicate its contents, the copy constructor is called. The copy constructor works directly with the supplied argument, and the standard return values are `FAILURE` on failure and `SUCCESS` on success.

If you omit the call to the copy constructor in this example, both `parameter` and `return_value` would point to the same internal data, meaning that `return_value` would be an illegal additional reference to the same data structures. Whenever changes occurred in the data that `parameter` points to, `return_value` might be affected. Thus, in order to create separate copies, the copy constructor must be used.

The copy constructor's counterpart in the Zend API, the destructor **zval_dtor()**, does the opposite of the constructor.

Chapter 35. Returning Values

Returning values from your functions to PHP was described briefly in an earlier section; this section gives the details. Return values are passed via the `return_value` variable, which is passed to your functions as argument. The `return_value` argument consists of a `zval` container (see the earlier discussion of the call interface) that you can freely modify. The container itself is already allocated, so you don't have to run `MAKE_STD_ZVAL` on it. Instead, you can access its members directly.

To make returning values from functions easier and to prevent hassles with accessing the internal structures of the `zval` container, a set of predefined macros is available (as usual). These macros automatically set the correspondent type and value, as described in Table 35-1 and Table 35-2.

> **Note:** The macros in Table 35-1 automatically *return* from your function, those in Table 35-2 only *set* the return value; they don't return from your function.

Table 35-1. Predefined Macros for Returning Values from a Function

Macro	Description
`RETURN_RESOURCE(resource)`	Returns a resource.
`RETURN_BOOL(bool)`	Returns a Boolean.
`RETURN_NULL()`	Returns nothing (a NULL value).
`RETURN_LONG(long)`	Returns a long.
`RETURN_DOUBLE(double)`	Returns a double.
`RETURN_STRING(string, duplicate)`	Returns a string. The `duplicate` flag indicates whether the string should be duplicated using **estrdup()**.
`RETURN_STRINGL(string, length, duplicate)`	Returns a string of the specified length; otherwise, behaves like `RETURN_STRING`. This macro is faster and binary-safe, however.
`RETURN_EMPTY_STRING()`	Returns an empty string.
`RETURN_FALSE`	Returns Boolean false.
`RETURN_TRUE`	Returns Boolean true.

Table 35-2. Predefined Macros for Setting the Return Value of a Function

Macro	Description
`RETVAL_RESOURCE(resource)`	Sets the return value to the specified resource.
`RETVAL_BOOL(bool)`	Sets the return value to the specified Boolean value.
`RETVAL_NULL`	Sets the return value to NULL.
`RETVAL_LONG(long)`	Sets the return value to the specified long.
`RETVAL_DOUBLE(double)`	Sets the return value to the specified double.
`RETVAL_STRING(string, duplicate)`	Sets the return value to the specified string and duplicates it to Zend internal memory if desired (see also `RETURN_STRING`).
`RETVAL_STRINGL(string, length, duplicate)`	Sets the return value to the specified string and forces the length to become `length` (see also `RETVAL_STRING`). This macro is faster and binary-safe, and should be used whenever the string length is known.
`RETVAL_EMPTY_STRING`	Sets the return value to an empty string.
`RETVAL_FALSE`	Sets the return value to Boolean false.
`RETVAL_TRUE`	Sets the return value to Boolean true.

Complex types such as arrays and objects can be returned by using **array_init()** and **object_init()**, as well as the corresponding hash functions on `return_value`. Since these types cannot be constructed of trivial information, there are no predefined macros for them.

Chapter 36. Printing Information

Often it's necessary to print messages to the output stream from your module, just as **print()** would be used within a script. PHP offers functions for most generic tasks, such as printing warning messages, generating output for **phpinfo()**, and so on. The following sections provide more details. Examples of these functions can be found on the CD-ROM.

---

zend_printf()

zend_printf() works like the standard **printf()**, except that it prints to Zend's output stream.

---

zend_error()

zend_error() can be used to generate error messages. This function accepts two arguments; the first is the error type (see `zend_errors.h`), and the second is the error message.

```
zend_error(E_WARNING, "This function has been called with empty arguments");
```

Table 36-1 shows a list of possible values (see Figure 36-1). These values are also referred to in `php.ini`. Depending on which error type you choose, your messages will be logged.

Table 36-1. Zend's Predefined Error Messages.

Error	Description
`E_ERROR`	Signals an error and terminates execution of the script immediately .
`E_WARNING`	Signals a generic warning. Execution continues.
`E_PARSE`	Signals a parser error. Execution continues.
`E_NOTICE`	Signals a notice. Execution continues. Note that by default the display of this type of error messages is turned off in `php.ini`.
`E_CORE_ERROR`	Internal error by the core; shouldn't be used by user-written modules.
`E_COMPILE_ERROR`	Internal error by the compiler; shouldn't be used by user-written modules.
`E_COMPILE_WARNING`	Internal warning by the compiler; shouldn't be used by user-written modules.

Figure 36-1. Display of warning messages in the browser.

Including Output in phpinfo()

After creating a real module, you'll want to show information about the module in **phpinfo()** (in addition to the module name, which appears in the module list by default). PHP allows you to create your own section in the **phpinfo()** output with the `ZEND_MINFO()` function. This function should be placed in the module descriptor block (discussed earlier) and is always called whenever a script calls **phpinfo()**.

PHP automatically prints a section in **phpinfo()** for you if you specify the `ZEND_MINFO` function, including the module name in the heading. Everything else must be formatted and printed by you.

Typically, you can print an HTML table header using **php_info_print_table_start()** and then use the standard functions **php_info_print_table_header()** and **php_info_print_table_row()**. As arguments, both take the number of columns (as integers) and the column contents (as strings). Example 36-1 shows a source example and its output. To print the table footer, use **php_info_print_table_end()**.

Example 36-1. Source code and screenshot for output in phpinfo().

```
php_info_print_table_start();
php_info_print_table_header(2, "First column", "Second column");
php_info_print_table_row(2, "Entry in first row", "Another entry");
php_info_print_table_row(2, "Just to fill", "another row here");
php_info_print_table_end();
```

Execution Information

You can also print execution information, such as the current file being executed. The name of the function currently being executed can be retrieved using the function **get_active_function_name()**. This function returns a pointer to the function name and doesn't accept any arguments. To retrieve the name of the file currently being executed, use **zend_get_executed_filename()**. This function accesses the executor globals, which are passed to it using the `TSRMLS_C` macro. The executor globals are automatically available to every function that's called directly by Zend (they're part of the `INTERNAL_FUNCTION_PARAMETERS` described earlier in this chapter). If you want to access the executor globals in another function that doesn't have them available automatically, call the macro `TSRMLS_FETCH()` once in that function; this will introduce them to your local scope.

Finally, the line number currently being executed can be retrieved using the function **zend_get_executed_lineno()**. This function also requires the executor globals as arguments. For examples of these functions, see Example 36-2.

Example 36-2. Printing execution information.

```
zend_printf("The name of the current function is %s<br>", get_active_function_name(TSRMLS_C));
zend_printf("The file currently executed is %s<br>", zend_get_executed_filename(TSRMLS_C));
zend_printf("The current line being executed is %i<br>", zend_get_executed_lineno(TSRMLS_C));
```

Chapter 37. Startup and Shutdown Functions

Startup and shutdown functions can be used for one-time initialization and deinitialization of your modules. As discussed earlier in this chapter (see the description of the Zend module descriptor block), there are module, and request startup and shutdown events.

The module startup and shutdown functions are called whenever a module is loaded and needs initialization; the request startup and shutdown functions are called every time a request is processed (meaning that a file is being executed).

For dynamic extensions, module and request startup/shutdown events happen at the same time.

Declaration and implementation of these functions can be done with macros; see the earlier section "Declaration of the Zend Module Block" for details.

Chapter 38. Calling User Functions

You can call user functions from your own modules, which is very handy when implementing callbacks; for example, for array walking, searching, or simply for event-based programs.

User functions can be called with the function **call_user_function_ex()**. It requires a hash value for the function table you want to access, a pointer to an object (if you want to call a method), the function name, return value, number of arguments, argument array, and a flag indicating whether you want to perform zval separation.

```
ZEND_API int call_user_function_ex(HashTable *function_table, zval *object,
zval *function_name, zval **retval_ptr_ptr,
int param_count, zval **params[],
int no_separation);
```

Note that you don't have to specify both `function_table` and `object`; either will do. If you want to call a method, you have to supply the object that contains this method, in which case **call_user_function()**automatically sets the function table to this object's function table. Otherwise, you only need to specify `function_table` and can set `object` to `NULL`.

Usually, the default function table is the "root" function table containing all function entries. This function table is part of the compiler globals and can be accessed using the macro `CG`. To introduce the compiler globals to your function, call the macro `TSRMLS_FETCH` once.

The function name is specified in a `zval` container. This might be a bit surprising at first, but is quite a logical step, since most of the time you'll accept function names as parameters from calling functions within your script, which in turn are contained in `zval` containers again. Thus, you only have to pass your arguments through to this function. This `zval` must be of type `IS_STRING`.

The next argument consists of a pointer to the return value. You don't have to allocate memory for this container; the function will do so by itself. However, you have to destroy this container (using **zval_dtor()**) afterward!

Next is the parameter count as integer and an array containing all necessary parameters. The last argument specifies whether the function should perform zval separation - this should always be set to `0`. If set to `1`, the function consumes less memory but fails if any of the parameters need separation.

Example 38-1 shows a small demonstration of calling a user function. The code calls a function that's supplied to it as argument and directly passes this function's return value through as its own return value. Note the use of the constructor and destructor calls at the end - it might not be necessary to do it this way here (since they should be separate values, the assignment might be safe), but this is bulletproof.

Example 38-1. Calling user functions.

```
zval **function_name;
zval *retval;
```

```
if((ZEND_NUM_ARGS() != 1) || (zend_get_parameters_ex(1, &function_name) != SUCCESS))
{
    WRONG_PARAM_COUNT;
}

if((*function_name)->type != IS_STRING)
{
    zend_error(E_ERROR, "Function requires string argument");
}

TSRMSLS_FETCH();

if(call_user_function_ex(CG(function_table), NULL, *function_name, &retval, 0, NULL, 0) != SUCCESS)
{
    zend_error(E_ERROR, "Function call failed");
}

zend_printf("We have %i as type<br>", retval->type);

*return_value = *retval;
zval_copy_ctor(return_value);
zval_ptr_dtor(&retval);


<?php

dl("call_userland.so");

function test_function()
{

    print("We are in the test function!<br>");

    return("hello");

}

$return_value = call_userland("test_function");

print("Return value: \"$return_value\"<br>");
?>
```

Chapter 39. Initialization File Support

PHP 4 features a redesigned initialization file support. It's now possible to specify default initialization entries directly in your code, read and change these values at runtime, and create message handlers for change notifications.

To create an .ini section in your own module, use the macros `PHP_INI_BEGIN()` to mark the beginning of such a section and `PHP_INI_END()` to mark its end. In between you can use `PHP_INI_ENTRY()` to create entries.

```
PHP_INI_BEGIN()
PHP_INI_ENTRY("first_ini_entry",  "has_string_value", PHP_INI_ALL, NULL)
PHP_INI_ENTRY("second_ini_entry", "2",                PHP_INI_SYSTEM, OnChangeSecond)
PHP_INI_ENTRY("third_ini_entry",  "xyz",              PHP_INI_USER, NULL)
PHP_INI_END()
```

The `PHP_INI_ENTRY()` macro accepts four parameters: the entry name, the entry value, its change permissions, and a pointer to a change-notification handler. Both entry name and value must be specified as strings, regardless of whether they really are strings or integers.

The permissions are grouped into three sections:`PHP_INI_SYSTEM` allows a change only directly in the `php.ini` file; `PHP_INI_USER` allows a change to be overridden by a user at runtime using additional configuration files, such as `.htaccess`; and `PHP_INI_ALL` allows changes to be made without restrictions. There's also a fourth level, `PHP_INI_PERDIR`, for which we couldn't verify its behavior yet.

The fourth parameter consists of a pointer to a change-notification handler. Whenever one of these initialization entries is changed, this handler is called. Such a handler can be declared using the `PHP_INI_MH` macro:

```
PHP_INI_MH(OnChangeSecond);            // handler for ini-entry "second_ini_entry"

// specify ini-entries here

PHP_INI_MH(OnChangeSecond)
{

    zend_printf("Message caught, our ini entry has been changed to %s<br>", new_value);

    return(SUCCESS);

}
```

The new value is given to the change handler as string in the variable `new_value`. When looking at the definition of `PHP_INI_MH`, you actually have a few parameters to use:

```
#define PHP_INI_MH(name) int name(php_ini_entry *entry, char *new_value,
                                  uint new_value_length, void *mh_arg1,
                                  void *mh_arg2, void *mh_arg3)
```

All these definitions can be found in `php_ini.h`. Your message handler will have access to a structure that contains the full entry, the new value, its length, and three optional arguments. These optional arguments can be specified with the additional macros `PHP_INI_ENTRY1` (allowing one additional argument), `PHP_INI_ENTRY2` (allowing two additional arguments), and `PHP_INI_ENTRY3` (allowing three additional arguments).

The change-notification handlers should be used to cache initialization entries locally for faster access or to perform certain tasks that are required if a value changes. For example, if a constant connection to a certain host is required by a module and someone changes the hostname, automatically terminate the old connection and attempt a new one.

Access to initialization entries can also be handled with the macros shown in Table 39-1.

Table 39-1. Macros to Access Initialization Entries in PHP

Macro	Description
`INI_INT(name)`	Returns the current value of entry `name` as integer (long).
`INI_FLT(name)`	Returns the current value of entry `name` as float (double).
`INI_STR(name)`	Returns the current value of entry `name` as string. *Note:* This string is not duplicated, but instead points to internal data. Further access requires duplication to local memory.
`INI_BOOL(name)`	Returns the current value of entry `name` as Boolean (defined as `zend_bool`, which currently means `unsigned char`).
`INI_ORIG_INT(name)`	Returns the original value of entry `name` as integer (long).
`INI_ORIG_FLT(name)`	Returns the original value of entry `name` as float (double).
`INI_ORIG_STR(name)`	Returns the original value of entry `name` as string. Note: This string is not duplicated, but instead points to internal data. Further access requires duplication to local memory.
`INI_ORIG_BOOL(name)`	Returns the original value of entry `name` as Boolean (defined as `zend_bool`, which currently means `unsigned char`).

Finally, you have to introduce your initialization entries to PHP. This can be done in the module startup and shutdown functions, using the macros `REGISTER_INI_ENTRIES()` and `UNREGISTER_INI_ENTRIES()`:

```
ZEND_MINIT_FUNCTION(mymodule)
{
    REGISTER_INI_ENTRIES();
}

ZEND_MSHUTDOWN_FUNCTION(mymodule)
{
    UNREGISTER_INI_ENTRIES();
}
```

Chapter 40. Where to Go from Here

You've learned a lot about PHP. You now know how to create dynamic loadable modules and statically linked extensions. You've learned how PHP and Zend deal with internal storage of variables and how you can create and access these variables. You know quite a set of tool functions that do a lot of routine tasks such as printing informational texts, automatically introducing variables to the symbol table, and so on.

Even though this chapter often had a mostly "referential" character, we hope that it gave you insight on how to start writing your own extensions. For the sake of space, we had to leave out a lot; we suggest that you take the time to study the header files and some modules (especially the ones in the `ext/standard` directory and the MySQL module, as these implement commonly known functionality). This will give you an idea of how other people have used the API functions - particularly those that didn't make it into this chapter.

Chapter 41. Reference: Some Configuration Macros

`config.m4`

The file `config.m4` is processed by `buildconf` and must contain all the instructions to be executed during configuration. For example, these can include tests for required external files, such as header files, libraries, and so on. PHP defines a set of macros that can be used in this process, the most useful of which are described in Table 41-1.

Table 41-1. M4 Macros for `config.m4`

Macro	Description
`AC_MSG_CHECKING(message)`	Prints a "checking <message>" text during `configure`.
`AC_MSG_RESULT(value)`	Gives the result to `AC_MSG_CHECKING`; should specify either `yes` or `no` as `value`.
`AC_MSG_ERROR(message)`	Prints `message` as error message during `configure` and aborts the script.
`AC_DEFINE(name,value,description)`	Adds `#define` to `php_config.h` with the value of `value` and a comment that says `description` (this is useful for conditional compilation of your module).
`AC_ADD_INCLUDE(path)`	Adds a compiler include path; for example, used if the module needs to add search paths for header files.
`AC_ADD_LIBRARY_WITH_PATH(libraryname,librarypath)`	Specifies an additional library to link.
`AC_ARG_WITH(modulename,description,unconditionaltest,conditionaltest)`	Quite a powerful macro, adding the module with `description` to the `configure --help` output. PHP checks whether the option `--with-<modulename>` is given to the `configure` script. If so, it runs the script `unconditionaltest` (for example, `--with-myext=yes`), in which case the value of the option is contained in the variable `$withval`. Otherwise, it executes `conditionaltest`.
`PHP_EXTENSION(modulename, [shared])`	This macro is a *must* to call for PHP to configure your extension. You can supply a second argument in addition to your module name, indicating whether you intend compilation as a shared module. This will result in a definition at compile time for your source as `COMPILE_DL_<modulename>`.

Chapter 42. API Macros

A set of macros was introduced into Zend's API that simplify access to `zval` containers (see Table 42-1).

Table 42-1. API Macros for Accessing `zval` Containers

Macro	Refers to
`Z_LVAL(zval)`	`(zval).value.lval`
`Z_DVAL(zval)`	`(zval).value.dval`
`Z_STRVAL(zval)`	`(zval).value.str.val`
`Z_STRLEN(zval)`	`(zval).value.str.len`
`Z_ARRVAL(zval)`	`(zval).value.ht`
`Z_LVAL_P(zval)`	`(*zval).value.lval`
`Z_DVAL_P(zval)`	`(*zval).value.dval`
`Z_STRVAL_P(zval_p)`	`(*zval).value.str.val`
`Z_STRLEN_P(zval_p)`	`(*zval).value.str.len`
`Z_ARRVAL_P(zval_p)`	`(*zval).value.ht`
`Z_LVAL_PP(zval_pp)`	`(**zval).value.lval`
`Z_DVAL_PP(zval_pp)`	`(**zval).value.dval`
`Z_STRVAL_PP(zval_pp)`	`(**zval).value.str.val`
`Z_STRLEN_PP(zval_pp)`	`(**zval).value.str.len`
`Z_ARRVAL_PP(zval_pp)`	`(**zval).value.ht`

Chapter 43. Streams API for PHP Extension Authors

Overview

The PHP Streams API introduces a unified approach to the handling of files and sockets in PHP extension. Using a single API with standard functions for common operations, the streams API allows your extension to access files, sockets, URLs, memory and script-defined objects. Streams is a run-time extensible API that allows dynamically loaded modules (and scripts!) to register new streams.

The aim of the Streams API is to make it comfortable for developers to open files, urls and other streamable data sources with a unified API that is easy to understand. The API is more or less based on the ANSI C stdio family of functions (with identical semantics for most of the main functions), so C programmers will have a feeling of familiarity with streams.

The streams API operates on a couple of different levels: at the base level, the API defines php_stream objects to represent streamable data sources. On a slightly higher level, the API defines php_stream_wrapper objects which "wrap" around the lower level API to provide support for retrieving data and meta-data from URLs.

Streams can be cast (converted) into other types of file-handles, so that they can be used with third-party libraries without a great deal of trouble. This allows those libraries to access data directly from URL sources. If your system has the **fopencookie()** or **funopen()** function, you can even pass any PHP stream to any library that uses ANSI stdio!

> **Note:** The functions in this chapter are for use in the PHP source code and are not PHP functions. Userland stream functions can be found in the Stream Reference.

Streams Basics

Using streams is very much like using ANSI stdio functions. The main difference is in how you obtain the stream handle to begin with. In most cases, you will use **php_stream_open_wrapper()** to obtain the stream handle. This function works very much like fopen, as can be seen from the example below:

Example 43-1. simple stream example that displays the PHP home page

```
php_stream * stream = php_stream_open_wrapper("http://www.php.net", "rb", REPORT_ERRORS, NULL);
if (stream) {
    while(!php_stream_eof(stream)) {
        char buf[1024];

        if (php_stream_gets(stream, buf, sizeof(buf))) {
            printf(buf);
        } else {
            break;
        }
    }
    php_stream_close(stream);
}
```

The table below shows the Streams equivalents of the more common ANSI stdio functions. Unless noted otherwise, the semantics of the functions are identical.

Table 43-1. ANSI stdio equivalent functions in the Streams API

ANSI Stdio Function	PHP Streams Function	Notes
fopen	php_stream_open_wrapper	Streams includes additional parameters
fclose	php_stream_close	
fgets	php_stream_gets	
fread	php_stream_read	The nmemb parameter is assumed to have a value of 1, so the prototype looks more like read(2)
fwrite	php_stream_write	The nmemb parameter is assumed to have a value of 1, so the prototype looks more like write(2)
fseek	php_stream_seek	
ftell	php_stream_tell	
rewind	php_stream_rewind	
feof	php_stream_eof	
fgetc	php_stream_getc	

ANSI Stdio Function	PHP Streams Function	Notes
fputc	php_stream_putc	
fflush	php_stream_flush	
puts	php_stream_puts	Same semantics as puts, NOT fputs
fstat	php_stream_stat	Streams has a richer stat structure

Streams as Resources

All streams are registered as resources when they are created. This ensures that they will be properly cleaned up even if there is some fatal error. All of the filesystem functions in PHP operate on streams resources - that means that your extensions can accept regular PHP file pointers as parameters to, and return streams from their functions. The streams API makes this process as painless as possible:

Example 43-2. How to accept a stream as a parameter

```
PHP_FUNCTION(example_write_hello)
{
    zval *zstream;
    php_stream *stream;

    if (FAILURE == zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "r", &zstream))
        return;

    php_stream_from_zval(stream, &zstream);

    /* you can now use the stream.  However, you do not "own" the
        stream, the script does.  That means you MUST NOT close the
        stream, because it will cause PHP to crash! */

    php_stream_write(stream, "hello\n");

    RETURN_TRUE();
}
```

Example 43-3. How to return a stream from a function

```
PHP_FUNCTION(example_open_php_home_page)
{
    php_stream *stream;

    stream = php_stream_open_wrapper("http://www.php.net", "rb", REPORT_ERRORS, NULL);

    php_stream_to_zval(stream, return_value);

    /* after this point, the stream is "owned" by the script.
        If you close it now, you will crash PHP! */
}
```

Since streams are automatically cleaned up, it's tempting to think that we can get away with being sloppy programmers and not bother to close the streams when we are done with them. Although such an approach might work, it is not a good idea for a number of reasons: streams hold locks on system resources while they are open, so leaving a file open after you have finished with it could prevent other processes from accessing it. If a script deals with a large number of files, the accumulation of the resources used, both in terms of memory and the sheer number of open files, can cause web server requests to fail. Sounds bad, doesn't it? The streams API includes some magic that helps you to keep your code clean - if a stream is not closed by your code when it should be, you will find some helpful debugging information in you web server error log.

> **Note:** Always use a debug build of PHP when developing an extension (`--enable-debug` when running configure), as a lot of effort has been made to warn you about memory and stream leaks.

In some cases, it is useful to keep a stream open for the duration of a request, to act as a log or trace file for example. Writing the code to safely clean up such a stream is not difficult, but it's several lines of code that are not strictly needed. To save yourself the touble of writing the code, you can mark a stream as being OK for auto cleanup. What this means is that the streams API will not emit a warning when it is time to auto-cleanup a stream. To do this, you can use **php_stream_auto_cleanup()**.

Streams Common API Reference

Table of Contents

php_stream_stat_path

(no version information, might be only in CVS)

php_stream_stat_path -- Gets the status for a file or URL

Description

int **php_stream_stat_path** (char * path, php_stream_statbuf * ssb)

php_stream_stat_path() examines the file or URL specified by *path* and returns information such as file size, access and creation times and so on. The return value is 0 on success, -1 on error. For more information about the information returned, see php_stream_statbuf.

php_stream_stat

(no version information, might be only in CVS)

php_stream_stat -- Gets the status for the underlying storage associated with a stream

Description

int **php_stream_stat** (php_stream * stream, php_stream_statbuf * ssb)

php_stream_stat() examines the storage to which *stream* is bound, and returns information such as file size, access and creation times and so on. The return value is 0 on success, -1 on error. For more information about the information returned, see php_stream_statbuf.

php_stream_open_wrapper

(no version information, might be only in CVS)

php_stream_open_wrapper -- Opens a stream on a file or URL

Description

php_stream * **php_stream_open_wrapper** (char * path, char * mode, int options, char ** opened)

php_stream_open_wrapper() opens a stream on the file, URL or other wrapped resource specified by *path*. Depending on the value of *mode*, the stream may be opened for reading, writing, appending or combinations of those. See the table below for the different modes that can be used; in addition to the characters listed below, you may include the character 'b' either as the second or last character in the mode string. The presence of the 'b' character informs the relevant stream implementation to open the stream in a binary safe mode.

The 'b' character is ignored on all POSIX conforming systems which treat binary and text files in the same way. It is a good idea to specify the 'b' character whenever your stream is accessing data where the full 8 bits are important, so that your code will work when compiled on a system where the 'b' flag is important.

Any local files created by the streams API will have their initial permissions set according to the operating system defaults - under UNIX based systems this means that the umask of the process will be used. Under Windows, the file will be owned by the creating process. Any remote files will be created according to the URL wrapper that was used to open the file, and the credentials supplied to the remote server.

r

Open text file for reading. The stream is positioned at the beginning of the file.

r+

Open text file for reading and writing. The stream is positioned at the beginning of the file.

w

Truncate the file to zero length or create text file for writing. The stream is positioned at the beginning of the file.

w+

Open text file for reading and writing. The file is created if it does not exist, otherwise it is truncated. The stream is positioned at the beginning of the file.

a

Open for writing. The file is created if it does not exist. The stream is positioned at the end of the file.

a+

Open text file for reading and writing. The file is created if it does not exist. The stream is positioned at the end of the file.

options affects how the path/URL of the stream is interpreted, safe mode checks and actions taken if there is an error during opening of the stream. See Stream open options for more information about options.

If *opened* is not NULL, it will be set to a string containing the name of the actual file/resource that was opened. This is important when the options include **USE_PATH**, which causes the include_path to be searched for the file. You, the caller, are responsible for calling **efree()** on the filename returned in this parameter.

> **Note:** If you specified **STREAM_MUST_SEEK** in *options*, the path returned in *opened* may not be the name of the actual stream that was returned to you. It will, however, be the name of the original resource from which the seekable stream was manufactured.

php_stream_read

(no version information, might be only in CVS)

php_stream_read -- Read a number of bytes from a stream into a buffer

Description

size_t **php_stream_read** (php_stream * stream, char * buf, size_t count)

php_stream_read() reads up to *count* bytes of data from *stream* and copies them into the buffer *buf*.

php_stream_read() returns the number of bytes that were read successfully. There is no distinction between a failed read or an end-of-file condition - use **php_stream_eof()** to test for an **EOF**.

The internal position of the stream is advanced by the number of bytes that were read, so that subsequent reads will continue reading from that point.

If less than *count* bytes are available to be read, this call will block (or wait) until the required number are available, depending on the blocking status of the stream. By default, a stream is opened in blocking mode. When reading from regular files, the

blocking mode will not usually make any difference: when the stream reaches the **EOF** **php_stream_read()** will return a value less than *count*, and 0 on subsequent reads.

php_stream_write

(no version information, might be only in CVS)

php_stream_write -- Write a number of bytes from a buffer to a stream

Description

size_t **php_stream_write** (php_stream * stream, const char * buf, size_t count)

php_stream_write() writes *count* bytes of data from *buf* into *stream*.

php_stream_write() returns the number of bytes that were read successfully. If there was an error, the number of bytes written will be less than *count*.

The internal position of the stream is advanced by the number of bytes that were written, so that subsequent writes will continue writing from that point.

php_stream_eof

(no version information, might be only in CVS)

php_stream_eof -- Check for an end-of-file condition on a stream

Description

int **php_stream_eof** (php_stream * stream)

php_stream_eof() checks for an end-of-file condition on *stream*.

php_stream_read() returns the 1 to indicate **EOF**, 0 if there is no **EOF** and -1 to indicate an error.

php_stream_getc

(no version information, might be only in CVS)

php_stream_getc -- Read a single byte from a stream

Description

int **php_stream_getc** (php_stream * stream)

php_stream_getc() reads a single character from *stream* and returns it as an unsigned char cast as an int, or **EOF** if the end-of-file is reached, or an error occurred.

php_stream_getc() may block in the same way as **php_stream_read()** blocks.

The internal position of the stream is advanced by 1 if successful.

php_stream_gets

(no version information, might be only in CVS)

php_stream_gets -- Read a line of data from a stream into a buffer

Description

char * **php_stream_gets** (php_stream * stream, char * buf, size_t maxlen)

php_stream_gets() reads up to *count*-1 bytes of data from *stream* and copies them into the buffer *buf*. Reading stops after an **EOF** or a newline. If a newline is read, it is stored in *buf* as part of the returned data. A NUL terminating character is stored as the last character in the buffer.

php_stream_read() returns *buf* when successful or NULL otherwise.

The internal position of the stream is advanced by the number of bytes that were read, so that subsequent reads will continue reading from that point.

This function may block in the same way as **php_stream_read()**.

php_stream_close

(no version information, might be only in CVS)

php_stream_close -- Close a stream

Description

int **php_stream_close** (php_stream * stream)

php_stream_close() safely closes *stream* and releases the resources associated with it. After *stream* has been closed, it's value is undefined and should not be used.

php_stream_close() returns 0 if the stream was closed or **EOF** to indicate an error. Regardless of the success of the call, *stream* is undefined and should not be used after a call to this function.

php_stream_flush

(no version information, might be only in CVS)

php_stream_flush -- Flush stream buffers to storage

Description

int **php_stream_flush** (php_stream * stream)

php_stream_flush() causes any data held in write buffers in *stream* to be committed to the underlying storage.

php_stream_flush() returns 0 if the buffers were flushed, or if the buffers did not need to be flushed, but returns **EOF** to indicate an error.

php_stream_seek

(no version information, might be only in CVS)

php_stream_seek -- Reposition a stream

Description

int **php_stream_seek** (php_stream * stream, off_t offset, int whence)

php_stream_seek() repositions the internal position of *stream*. The new position is determined by adding the *offset* to the position indicated by *whence*. If *whence* is set to **SEEK_SET**, **SEEK_CUR** or **SEEK_END** the offset is relative to the start of the stream, the current position or the end of the stream, respectively.

php_stream_seek() returns 0 on success, but -1 if there was an error.

> **Note:** Not all streams support seeking, although the streams API will emulate a seek if *whence* is set to **SEEK_CUR** and *offset* is positive, by calling **php_stream_read()** to read (and discard) *offset* bytes.
>
> The emulation is only applied when the underlying stream implementation does not support seeking. If the stream is (for example) a file based stream that is wrapping a non-seekable pipe, the streams api will not apply emulation because the file based stream implements a seek operation; the seek will fail and an error result will be returned to

the caller.

php_stream_tell

(no version information, might be only in CVS)

php_stream_tell -- Determine the position of a stream

Description

off_t **php_stream_tell** (php_stream * stream)

php_stream_tell() returns the internal position of *stream*, relative to the start of the stream. If there is an error, -1 is returned.

php_stream_copy_to_stream

(no version information, might be only in CVS)

php_stream_copy_to_stream -- Copy data from one stream to another

Description

size_t **php_stream_copy_to_stream** (php_stream * src, php_stream * dest, size_t maxlen)

php_stream_copy_to_stream() attempts to read up to *maxlen* bytes of data from *src* and write them to *dest*, and returns the number of bytes that were successfully copied.

If you want to copy all remaining data from the *src* stream, pass the constant **PHP_STREAM_COPY_ALL** as the value of *maxlen*.

> **Note:** This function will attempt to copy the data in the most efficient manner, using memory mapped files when possible.

php_stream_copy_to_mem

(no version information, might be only in CVS)

php_stream_copy_to_mem -- Copy data from stream and into an allocated buffer

Description

size_t **php_stream_copy_to_mem** (php_stream * src, char ** buf, size_t maxlen, int persistent)

php_stream_copy_to_mem() allocates a buffer *maxlen*+1 bytes in length using **pemalloc()** (passing *persistent*). It then reads *maxlen* bytes from *src* and stores them in the allocated buffer.

The allocated buffer is returned in *buf*, and the number of bytes successfully read. You, the caller, are responsible for freeing the buffer by passing it and *persistent* to **pefree()**.

If you want to copy all remaining data from the *src* stream, pass the constant **PHP_STREAM_COPY_ALL** as the value of *maxlen*.

> **Note:** This function will attempt to copy the data in the most efficient manner, using memory mapped files when possible.

php_stream_make_seekable

(no version information, might be only in CVS)

php_stream_make_seekable -- Convert a stream into a stream is seekable

Description

int **php_stream_make_seekable** (php_stream * origstream, php_stream ** newstream, int flags)

php_stream_make_seekable() checks if *origstream* is seekable. If it is not, it will copy the data into a new temporary stream. If successful, *newstream* is always set to the stream that is valid to use, even if the original stream was seekable.

flags allows you to specify your preference for the seekeable stream that is returned: use **PHP_STREAM_NO_PREFERENCE** to use the default seekable stream (which uses a dynamically expanding memory buffer, but switches to temporary file backed storage when the stream size becomes large), or use **PHP_STREAM_PREFER_STDIO** to use "regular" temporary file backed storage.

Table 43-1. php_stream_make_seekable() return values

Value	Meaning
PHP_STREAM_UNCHANGED	Original stream was seekable anyway. *newstream* is set to the value of *origstream*.
PHP_STREAM_RELEASED	Original stream was not seekable and has been released. *newstream* is set to the new seekable stream. You should not access *origstream* anymore.
PHP_STREAM_FAILED	An error occurred while attempting conversion. *newstream* is set to NULL; *origstream* is still valid.
PHP_STREAM_CRITICAL	An error occurred while attempting conversion that has left *origstream* in an indeterminate state. *newstream* is set to NULL and it is highly recommended that you close *origstream*.

> **Note:** If you need to seek and write to the stream, it does not make sense to use this function, because the stream it returns is not guaranteed to be bound to the same resource as the original stream.
>
> **Note:** If you only need to seek forwards, there is no need to call this function, as the streams API will emulate forward seeks when the whence parameter is **SEEK_CUR**.
>
> **Note:** If *origstream* is network based, this function will block until the whole contents have been downloaded.
>
> **Note:** NEVER call this function with an *origstream* that is reference by a file pointer in a PHP script! This function may cause the underlying stream to be closed which could cause a crash when the script next accesses the file pointer!
>
> **Note:** In many cases, this function can only succeed when *origstream* is a newly opened stream with no data buffered in the stream layer. For that reason, and because this function is complicated to use correctly, it is recommended that you use **php_stream_open_wrapper()** and pass in **PHP_STREAM_MUST_SEEK** in your options instead of calling this function directly.

php_stream_cast

(no version information, might be only in CVS)

php_stream_cast -- Convert a stream into another form, such as a FILE* or socket

Description

int **php_stream_cast** (php_stream * stream, int castas, void ** ret, int flags)

php_stream_cast() attempts to convert *stream* into a resource indicated by *castas*. If *ret* is NULL, the stream is queried to find out if such a conversion is possible, without actually performing the conversion (however, some internal stream state *might* be changed in this case). If *flags* is set to **REPORT_ERRORS**, an error message will be displayed is there is an error during conversion.

> **Note:** This function returns **SUCCESS** for success or **FAILURE** for failure. Be warned that you must explicitly compare the return value with **SUCCESS** or **FAILURE** because of the underlying values of those constants. A simple boolean expression will not be interpreted as you intended.

Table 43-1. Resource types for *castas*

Value	Meaning
PHP_STREAM_AS_STDIO	Requests an ANSI FILE* that represents the stream
PHP_STREAM_AS_FD	Requests a POSIX file descriptor that represents the stream
PHP_STREAM_AS_SOCKETD	Requests a network socket descriptor that represents the stream

In addition to the basic resource types above, the conversion process can be altered by using the following flags by using the OR operator to combine the resource type with one or more of the following values:

Table 43-2. Resource types for *castas*

Value	Meaning
PHP_STREAM_CAST_TRY_HARD	Tries as hard as possible, at the expense of additional resources, to ensure that the conversion succeeds
PHP_STREAM_CAST_RELEASE	Informs the streams API that some other code (possibly a third party library) will be responsible for closing the underlying handle/resource. This causes the *stream* to be closed in such a way the underlying handle is preserved and returned in *ret*. If this function succeeds, *stream* should be considered closed and should no longer be used.

> **Note:** If your system supports **fopencookie()** (systems using glibc 2 or later), the streams API will always be able to synthesize an ANSI FILE* pointer over any stream. While this is tremendously useful for passing any PHP stream to any third-party libraries, such behaviour is not portable. You are requested to consider the portability implications before distributing you extension. If the fopencookie synthesis is not desireable, you should query the stream to see if it naturally supports FILE* by using **php_stream_is()**

> **Note:** If you ask a socket based stream for a FILE*, the streams API will use **fdopen()** to create it for you. Be warned that doing do may cause data that was buffered in the streams layer to be lost if you intermix streams API calls with ANSI stdio calls.

See also **php_stream_is()** and **php_stream_can_cast()**.

php_stream_can_cast

(no version information, might be only in CVS)

php_stream_can_cast -- Determines if a stream can be converted into another form, such as a FILE* or socket

Description

int **php_stream_can_cast** (php_stream * stream, int castas)

This function is equivalent to calling **php_stream_cast()** with *ret* set to NULL and *flags* set to 0. It returns **SUCCESS** if the stream can be converted into the form requested, or **FAILURE** if the conversion cannot be performed.

> **Note:** Although this function will not perform the conversion, some internal stream state *might* be changed by this call.

> **Note:** You must explicity compare the return value of this function with one of the constants, as described in **php_stream_cast()**.

See also **php_stream_cast()** and **php_stream_is()**.

php_stream_is_persistent

(no version information, might be only in CVS)

php_stream_is_persistent -- Determines if a stream is a persistent stream

Description

int **php_stream_is_persistent** (php_stream * stream)

php_stream_is_persistent() returns 1 if the stream is a persistent stream, 0 otherwise.

php_stream_is

(no version information, might be only in CVS)

php_stream_is -- Determines if a stream is of a particular type

Description

int **php_stream_is** (php_stream * stream, int istype)

php_stream_is() returns 1 if *stream* is of the type specified by *istype*, or 0 otherwise.

Table 43-1. Values for *istype*

Value	Meaning
PHP_STREAM_IS_STDIO	The stream is implemented using the stdio implementation
PHP_STREAM_IS_SOCKET	The stream is implemented using the network stream implementation
PHP_STREAM_IS_USERSPACE	The stream is implemented using the userspace object implementation
PHP_STREAM_IS_MEMORY	The stream is implemented using the grow-on-demand memory stream implementation

> **Note:** The PHP_STREAM_IS_XXX "constants" are actually defined as pointers to the underlying stream operations structure. If your extension (or some other extension) defines additional streams, it should also declare a PHP_STREAM_IS_XXX constant in it's header file that you can use as the basis of this comparison.
>
> **Note:** This function is implemented as a simple (and fast) pointer comparision, and does not change the stream state in any way.

See also **php_stream_cast()** and **php_stream_can_cast()**.

php_stream_passthru

(no version information, might be only in CVS)

php_stream_passthru -- Outputs all remaining data from a stream

Description

size_t **php_stream_passthru** (php_stream * stream)

php_stream_passthru() outputs all remaining data from *stream* to the active output buffer and returns the number of bytes output. If buffering is disabled, the data is written straight to the output, which is the browser making the request in the case of PHP on a web server, or stdout for CLI based PHP. This function will use memory mapped files if possible to help improve performance.

php_register_url_stream_wrapper

(no version information, might be only in CVS)

php_register_url_stream_wrapper -- Registers a wrapper with the Streams API

Description

int **php_register_url_stream_wrapper** (char * protocol, php_stream_wrapper * wrapper, TSRMLS_DC)

php_register_url_stream_wrapper() registers *wrapper* as the handler for the protocol specified by *protocol*.

> **Note:** If you call this function from a loadable module, you *MUST* call **php_unregister_url_stream_wrapper()** in your module shutdown function, otherwise PHP will crash.

php_unregister_url_stream_wrapper

(no version information, might be only in CVS)

php_unregister_url_stream_wrapper -- Un-registers a wrapper from the Streams API

Description

int **php_unregister_url_stream_wrapper** (char * protocol, TSRMLS_DC)

php_unregister_url_stream_wrapper() unregisters the wrapper associated with *protocol*.

php_stream_open_wrapper_ex

(no version information, might be only in CVS)

php_stream_open_wrapper_ex -- Opens a stream on a file or URL, specifying context

Description

php_stream * **php_stream_open_wrapper_ex** (char * path, char * mode, int options, char ** opened, php_stream_context * context)

php_stream_open_wrapper_ex() is exactly like **php_stream_open_wrapper()**, but allows you to specify a php_stream_context object using *context*. To find out more about stream contexts, see XXX

php_stream_open_wrapper_as_file

(no version information, might be only in CVS)

php_stream_open_wrapper_as_file -- Opens a stream on a file or URL, and converts to a FILE*

Description

FILE * **php_stream_open_wrapper_as_file** (char * path, char * mode, int options, char ** opened)

php_stream_open_wrapper_as_file() is exactly like **php_stream_open_wrapper()**, but converts the stream into an ANSI stdio FILE* and returns that instead of the stream. This is a convenient shortcut for extensions that pass FILE* to third-party libraries.

Streams Dir API Reference

Table of Contents

The functions listed in this section work on local files, as well as remote files (provided that the wrapper supports this functionality!).

php_stream_opendir

(no version information, might be only in CVS)

php_stream_opendir -- Open a directory for file enumeration

Description

php_stream * **php_stream_opendir** (char * path, php_stream_context * context)

php_stream_opendir() returns a stream that can be used to list the files that are contained in the directory specified by *path*. This function is functionally equivalent to POSIX **opendir()**. Although this function returns a php_stream object, it is not recommended to try to use the functions from the common API on these streams.

php_stream_readdir

(no version information, might be only in CVS)

php_stream_readdir -- Fetch the next directory entry from an opened dir

Description

php_stream_dirent * **php_stream_readdir** (php_stream * dirstream, php_stream_dirent * ent)

php_stream_readdir() reads the next directory entry from *dirstream* and stores it into *ent*. If the function succeeds, the return value is *ent*. If the function fails, the return value is NULL. See php_stream_dirent for more details about the information returned for each directory entry.

php_stream_rewinddir

(no version information, might be only in CVS)

php_stream_rewinddir -- Rewind a directory stream to the first entry

Description

int **php_stream_rewinddir** (php_stream * dirstream)

php_stream_rewinddir() rewinds a directory stream to the first entry. Returns 0 on success, but -1 on failure.

php_stream_closedir

(no version information, might be only in CVS)

php_stream_closedir -- Close a directory stream and release resources

Description

int **php_stream_closedir** (php_stream * dirstream)

php_stream_closedir() closes a directory stream and releases resources associated with it. Returns 0 on success, but -1 on failure.

Streams File API Reference

Table of Contents

php_stream_fopen_from_file

(no version information, might be only in CVS)

php_stream_fopen_from_file -- Convert an ANSI FILE* into a stream

Description

php_stream * **php_stream_fopen_from_file** (FILE * file, char * mode)

php_stream_fopen_from_file() returns a stream based on the *file*. *mode* must be the same as the mode used to open *file*, otherwise strange errors may occur when trying to write when the mode of the stream is different from the mode on the file.

php_stream_fopen_tmpfile

(no version information, might be only in CVS)

php_stream_fopen_tmpfile -- Open a FILE* with tmpfile() and convert into a stream

Description

php_stream * **php_stream_fopen_tmpfile** (void)

php_stream_fopen_from_file() returns a stream based on a temporary file opened with a mode of "w+b". The temporary file will be deleted automatically when the stream is closed or the process terminates.

php_stream_fopen_temporary_file

(no version information, might be only in CVS)

php_stream_fopen_temporary_file -- Generate a temporary file name and open a stream on it

Description

php_stream * **php_stream_fopen_temporary_file** (const char * dir, const char * pfx, char ** opened)

php_stream_fopen_temporary_file() generates a temporary file name in the directory specified by *dir* and with a prefix of *pfx*. The generated file name is returns in the *opened* parameter, which you are responsible for cleaning up using **efree()**. A stream is opened on that generated filename in "w+b" mode. The file is NOT automatically deleted; you are responsible for unlinking or moving the file when you have finished with it.

Streams Socket API Reference

Table of Contents

php_stream_sock_open_from_socket

(no version information, might be only in CVS)

php_stream_sock_open_from_socket -- Convert a socket descriptor into a stream

Description

php_stream * **php_stream_sock_open_from_socket** (int socket, int persistent)

php_stream_sock_open_from_socket() returns a stream based on the *socket*. *persistent* is a flag that controls whether the stream is opened as a persistent stream. Generally speaking, this parameter will usually be o.

php_stream_sock_open_host

(no version information, might be only in CVS)

php_stream_sock_open_host -- Open a connection to a host and return a stream

Description

php_stream * **php_stream_sock_open_host** (const char * host, unsigned short port, int socktype, struct timeval * timeout, int persistent)

php_stream_sock_open_host() establishes a connect to the specified *host* and *port*. *socktype* specifies the connection semantics that should apply to the connection. Values for *socktype* are system dependent, but will usually include (at a minimum) **SOCK_STREAM** for sequenced, reliable, two-way connection based streams (TCP), or **SOCK_DGRAM** for connectionless, unreliable messages of a fixed maximum length (UDP).

persistent is a flag the controls whether the stream is opened as a persistent stream. Generally speaking, this parameter will

usually be 0.

If not NULL, *timeout* specifies a maximum time to allow for the connection to be made. If the connection attempt takes longer than the timeout value, the connection attempt is aborted and NULL is returned to indicate that the stream could not be opened.

> **Note:** The timeout value does not include the time taken to perform a DNS lookup. The reason for this is because there is no portable way to implement a non-blocking DNS lookup.
>
> The timeout only applies to the connection phase; if you need to set timeouts for subsequent read or write operations, you should use **php_stream_sock_set_timeout()** to configure the timeout duration for your stream once it has been opened.

The streams API places no restrictions on the values you use for *socktype*, but encourages you to consider the portability of values you choose before you release your extension.

php_stream_sock_open_unix

(no version information, might be only in CVS)

php_stream_sock_open_unix -- Open a UNIX domain socket and convert into a stream

Description

php_stream * **php_stream_sock_open_unix** (const char * path, int pathlen, int persistent, struct timeval * timeout)

php_stream_sock_open_unix() attempts to open the UNIX domain socket specified by *path*. *pathlen* specifies the length of *path*. If *timeout* is not NULL, it specifies a timeout period for the connection attempt. *persistent* indicates if the stream should be opened as a persistent stream. Generally speaking, this parameter will usually be 0.

> **Note:** This function will not work under Windows, which does not implement unix domain sockets. A possible exception to this rule is if your PHP binary was built using cygwin. You are encouraged to consider this aspect of the portability of your extension before it's release.
>
> **Note:** This function treats *path* in a binary safe manner, suitable for use on systems with an abstract namespace (such as Linux), where the first character of path is a NUL character.

Streams Structures

Table of Contents

struct php_stream_statbuf

struct php_stream_statbuf -- Holds information about a file or URL

Description

```
php_stream_statbuf
     struct stat sb
```

sb is a regular, system defined, struct stat.

struct php_stream_dirent

struct php_stream_dirent -- Holds information about a single file during dir scanning

Description

```
php_stream_dirent
```

```
char d_name[MAXPATHLEN]
```

d_name holds the name of the file, relative to the directory being scanned.

struct php_stream_ops

struct php_stream_ops -- Holds member functions for a stream implementation

Description

```
typedef struct _php_stream_ops {
             /* all streams MUST implement these operations */
             size_t (*write)(php_stream *stream, const char *buf, size_t count TSRMLS_DC);
             size_t (*read)(php_stream *stream, char *buf, size_t count TSRMLS_DC);
             int (*close)(php_stream *stream, int close_handle TSRMLS_DC);
             int (*flush)(php_stream *stream TSRMLS_DC);

             const char *label; /* name describing this class of stream */

             /* these operations are optional, and may be set to NULL if the stream does not
              * support a particular operation */
            int (*seek)(php_stream *stream, off_t offset, int whence TSRMLS_DC);
            char *(*gets)(php_stream *stream, char *buf, size_t size TSRMLS_DC);
            int (*cast)(php_stream *stream, int castas, void **ret TSRMLS_DC);
            int (*stat)(php_stream *stream, php_stream_statbuf *ssb TSRMLS_DC);
        } php_stream_ops;
```

Streams Constants

Table of Contents

Stream open options

Stream open options -- Affects the operation of stream factory functions

Description

One or more of these values can be combined using the OR operator.

IGNORE_PATH

This is the default option for streams; it requests that the include_path is not to be searched for the requested file.

USE_PATH

Requests that the include_path is to be searched for the requested file.

IGNORE_URL

Requests that registered URL wrappers are to be ignored when opening the stream. Other non-URL wrappers will be taken into consideration when decoding the path. There is no opposite form for this flag; the streams API will use all registered wrappers by default.

IGNORE_URL_WIN

On Windows systems, this is equivalent to IGNORE_URL. On all other systems, this flag has no effect.

ENFORCE_SAFE_MODE

Requests that the underlying stream implementation perform safe_mode checks on the file before opening the file. Omitting this flag will skip safe_mode checks and allow opening of any file that the PHP process has rights to access.

REPORT_ERRORS

If this flag is set, and there was an error during the opening of the file or URL, the streams API will call the php_error function for you. This is useful because the path may contain username/password information that should not be displayed in the browser output (it would be a security risk to do so). When the streams API raises the error, it first strips username/password information from the path, making the error message safe to display in the browser.

STREAM_MUST_SEEK

This flag is useful when your extension really must be able to randomly seek around in a stream. Some streams may not be seekable in their native form, so this flag asks the streams API to check to see if the stream does support seeking. If it does not, it will copy the stream into temporary storage (which may be a temporary file or a memory stream) which does support seeking. Please note that this flag is not useful when you want to seek the stream and write to it, because the stream you are accessing might not be bound to the actual resource you requested.

> **Note:** If the requested resource is network based, this flag will cause the opener to block until the whole contents have been downloaded.

STREAM_WILL_CAST

If your extension is using a third-party library that expects a FILE* or file descriptor, you can use this flag to request the streams API to open the resource but avoid buffering. You can then use **php_stream_cast()** to retrieve the FILE* or file descriptor that the library requires.

The is particularly useful when accessing HTTP URLs where the start of the actual stream data is found after an indeterminate offset into the stream.

Since this option disables buffering at the streams API level, you may experience lower performance when using streams functions on the stream; this is deemed acceptable because you have told streams that you will be using the functions to match the underlying stream implementation. Only use this option when you are sure you need it.

VI. FAQ: Frequently Asked Questions

Table of Contents

Chapter 44. General Information

This section holds the most general questions about PHP: what it is and what it does.

1. What is PHP?

From the preface of the manual:

PHP is an HTML-embedded scripting language. Much of its syntax is borrowed from C, Java and Perl with a couple of unique PHP-specific features thrown in. The goal of the language is to allow web developers to write dynamically generated pages quickly.

A nice introduction to PHP by Stig Sæther Bakken can be found here on the Zend website. Also, much of the PHP Conference Material is freely available.

2. What does PHP stand for?

PHP stands for *PHP: Hypertext Preprocessor*. This confuses many people because the first word of the acronym is the acronym. This type of acronym is called a recursive acronym. The curious can visit Free On-Line Dictionary of Computing for more information on recursive acronyms.

3. What is the relation between the versions?

PHP/FI 2.0 is an early and no longer supported version of PHP. PHP 3 is the successor to PHP/FI 2.0 and is a lot nicer. PHP 4 is the current generation of PHP, which uses the Zend engine under the hood. PHP 5 uses Zend engine 2 which, among other things, offers many additional OOP features. PHP 5 is experimental.

4. Can I run several versions of PHP at the same time?

Yes. See the INSTALL file that is included in the PHP 4 source distribution. Also, read the related appendix.

5. What are the differences between PHP 3 and PHP 4?

There are a couple of articles written on this by the authors of PHP 4. Here's a list of some of the more important new features:

- Extended API module
- Generalized build process under UNIX
- Generic web server interface that also supports multi-threaded web servers
- Improved syntax highlighter
- Native HTTP session support
- Output buffering support
- More powerful configuration system
- Reference counting

Please see the What's new in PHP 4 overview for a detailed explanation of these features and more. If you're migrating from PHP 3 to PHP 4, also read the related appendix.

6. I think I found a bug! Who should I tell?

You should go to the PHP Bug Database and make sure the bug isn't a known bug. If you don't see it in the database, use the reporting form to report the bug. It is important to use the bug database instead of just sending an email to one of the mailing lists because the bug will have a tracking number assigned and it will then be possible for you to go back later and check on the status of the bug. The bug database can be found at http://bugs.php.net/.

Chapter 45. Mailing lists

This section holds questions about how to get in touch with the PHP community. The best way is the mailing lists.

1. Are there any PHP mailing lists?

Of course! There are many mailing lists for several subjects. A whole list of mailing lists can be found on our Support page.

The most general mailing list is `php-general`. To subscribe, send mail to php-general-subscribe@lists.php.net. You don't need to include anything special in the subject or body of the message. To unsubscribe, send mail to php-general-unsubscribe@lists.php.net.

You can also subscribe and unsubscribe using the web interface on our Support page.

2. Are there any other communities?

There are countless of them around the world. We have links for example to some IRC servers and foreign language mailing lists on our Support page.

3. Help! I can't seem to subscribe/unsubscribe to/from one of the mailing lists!

If you have problems subscribing to or unsubscribing from the php-general mailing list, it may be because the mailing list software can't figure out the correct mailing address to use. If your email address was `joeblow@example.com`, you can send your subscription request to `php-general-subscribe-joeblow=example.com@lists.php.net`, or your unsubscription request to

`php-general-unsubscribe-joeblow=example.com@lists.php.net`. Use similar addresses for the other mailing lists.

4. Is there an archive of the mailing lists anywhere?

Yes, you will find a list of archive sites on the Support page. The mailing list articles are also archived as news messages. You can access the news server at news://news.php.net/ with a news client. There is also an experimental web interface for the news server at http://news.php.net/

5. What can I ask the mailing list?

Since PHP is growing more and more popular by the day the traffic has increased on the php-general mailing list and as of now the list gets about 150 to 200 posts a day. Because of this it is in everyones interest that you use the list as a last resort when you have looked everywhere else.

Before you post to the list please have a look in this FAQ and the manual to see if you can find the help there. If there is nothing to be found there try out the mailing list archives (see above). If you're having problem with installing or configuring PHP please read through all included documentation and README's. If you still can't find any information that helps you out you're more than welcome to use the mailing list.

Before asking questions, you may want to read the paper on How To Ask Questions The Smart Way as this is a good idea for everyone.

6. What information should I include when posting to the mailing list?

Posts like "I can't get PHP up and running! Help me! What is wrong?" are of absolutely no use to anyone. If you're having problems getting PHP up and running you must include what operating system you are running on, what version of PHP you're trying to set up, how you got it (pre-compiled, CVS, RPMs and so on), what you have done so far, where you got stuck and the exact error message.

This goes for any other problem as well. You have to include information on what you have done, where you got stuck, what you're trying to do and, if applicable, exact error messages. If you're having problems with your source code you need to include the part of the code that isn't working. Do not include more code than necessary though! It makes the post hard to read and a lot of people might just skip it all together because of this. If you're unsure about how much information to include in the mail it's better that you include to much than to little.

Another important thing to remember is to summarize your problem on the subject line. A subject like "HELP MEEEE!!!" or "What is the problem here?" will be ignored by the majority of the readers.

And lastly, you're encouraged to read the paper on How To Ask Questions The Smart Way as this will be a great help for everyone, especially yourself.

Chapter 46. Obtaining PHP

This section has details about PHP download locations, and OS issues.

1. Where can I obtain PHP?

You can download PHP from any of the members of the PHP network of sites. These can be found at http://www.php.net/. You can also use anonymous CVS to get the absolute latest version of the source. For more information, go to http://cvs.php.net/.

2. Are pre-compiled binary versions available?

We only distribute precompiled binaries for Windows systems, as we are not able to compile PHP for every major Linux/Unix platform with every extension combination. Also note, that many Linux distributions come with PHP built in these days. Windows binaries can be downloaded from our Downloads page, for Linux binaries, please visit your distributions website.

3. Where can I get libraries needed to compile some of the optional PHP extensions?

> **Note:** Those marked with * are not thread-safe libraries, and should not be used with PHP as a server module in the multi-threaded Windows web servers (IIS, Netscape). This does not matter in Unix environments, yet.

- LDAP (Unix).

- LDAP* (Unix).
- LDAP (Unix/Win) : Netscape Directory (LDAP) SDK 1.1.
- free LDAP server.
- Berkeley DB2 (Unix/Win) : http://www.sleepycat.com/.
- SNMP* (Unix): .
- GD* (Unix/Win).
- mSQL* (Win).
- mSQL* (Unix).
- PostgreSQL (Unix).
- IMAP* (Win/Unix).
- Sybase-CT* (Linux, libc5) : Available locally.
- FreeType (libttf):.
- ZLib (Unix/Win32).
- expat XML parser (Unix/Win32).
- PDFLib.
- mcrypt.
- mhash.
- t1lib.
- dmalloc.
- aspell.
- readline.

4. How do I get these libraries to work?

You will need to follow instructions provided with the library. Some of these libraries are detected automatically when you run the 'configure' script of PHP (such as the GD library), and others you will have to enable using '`--with-EXTENSION`' options to '`configure`'. Run '`configure --help`' for a listing of these.

5. I got the latest version of the PHP source code from the CVS repository on my Windows machine, what do I need to compile it?

First, you will need Microsoft Visual C++ v6 (v5 may do it also, but we do it with v6), and you will need some support files. See the manual section about building PHP from source on Windows.

6. Where do I find the Browser Capabilities File?

You can find a `browscap.ini` file at http://www.garykeith.com/browsers/downloads.asp.

Chapter 47. Database issues

This section holds common questions about relation between PHP and databases. Yes, PHP can access virtually any database available today.

1. I heard it's possible to access Microsoft SQL Server from PHP. How?

On Windows machines, you can simply use the included ODBC support and the correct ODBC driver.

On Unix machines, you can use the Sybase-CT driver to access Microsoft SQL Servers because they are (at least mostly) protocol-compatible. Sybase has made a free version of the necessary libraries for Linux systems. For other Unix operating systems, you need to contact Sybase for the correct libraries. Also see the answer to the next question.

2. Can I access Microsoft Access databases?

Yes. You already have all the tools you need if you are running entirely under Windows 9x/Me, or NT/2000, where you can use ODBC and Microsoft's ODBC drivers for Microsoft Access databases.

If you are running PHP on a Unix box and want to talk to MS Access on a Windows box you will need Unix ODBC drivers. OpenLink Software has Unix-based ODBC drivers that can do this. There is a free pilot program where you can download an evaluation copy that doesn't expire and prices start at $675 for the commercial supported version.

Another alternative is to use an SQL server that has Windows ODBC drivers and use that to store the data, which you can then access from Microsoft Access (using ODBC) and PHP (using the built in drivers), or to use an intermediary file format that Access and PHP both understand, such as flat files or dBase databases. On this point Tim Hayes from OpenLink software writes:

```
Using another database as an intermediary is not a good idea, when you can
      use ODBC from PHP straight to your database - i.e. with OpenLink's drivers. If
      you do need to use an intermediary file format, OpenLink have now released
      Virtuoso (a virtual database engine) for NT, Linux and other unix platforms.
      Please visit our website for a free download.
```

One option that has proven successful is to use MySQL and its MyODBC drivers on Windows and synchronizing the databases. Steve Lawrence writes:

- Install MySQL on your platform according to instructions with MySQL. Latest available from www.mysql.com (get it from your mirror!). No special configuration required except when you set up a database, and configure the user account, you should put % in the host field, or the host name of the Windows computer you wish to access MySQL with. Make a note of your server name, username, and password.
- Download the MyODBC for Windows driver from the MySQL site. Latest release is myodbc-2_50_19-win95.zip (NT available too, as well as source code). Install it on your Windows machine. You can test the operation with the utilities included with this program.
- Create a user or system dsn in your ODBC administrator, located in the control panel. Make up a dsn name, enter your hostname, user name, password, port, etc for you MySQL database configured in step 1.
- Install Access with a full install, this makes sure you get the proper add-ins.. at the least you will need ODBC support and the linked table manager.
- Now the fun part! Create a new access database. In the table window right click and select Link Tables, or under the file menu option, select Get External Data and then Link Tables. When the file browser box comes up, select files of type: ODBC. Select System dsn and the name of your dsn created in step 3. Select the table to link, press OK, and presto! You can now open the table and add/delete/edit data on your MySQL server! You can also build queries, import/export tables to MySQL, build forms and reports, etc.

Tips and Tricks:

- You can construct your tables in Access and export them to MySQL, then link them back in. That makes table creation quick.
- When creating tables in Access, you must have a primary key defined in order to have write access to the table in access. Make sure you create a primary key in MySQL before linking in access
- If you change a table in MySQL, you have to re-link it in Access. Go to tools>add-ins>linked table manager, cruise to your ODBC DSN, and select the table to re-link from there. you can also move your dsn source around there, just hit the always prompt for new location checkbox before pressing OK.

3. I upgraded to PHP 4, and now mysql keeps telling me "Warning: MySQL: Unable to save result set in ...". What's up?

Most likely what has happened is, PHP 4 was compiled with the '--with-mysql' option, without specifying the path to MySQL. This means PHP is using its built-in MySQL client library. If your system is running applications, such as PHP 3 as a concurrent Apache module, or auth-mysql, that use other versions of MySQL clients, then there is a conflict between the two differing versions of those clients.

Recompiling PHP 4, and adding the path to MySQL to the flag, '--with-mysql=/your/path/to/mysql' usually solves the problem.

4. After installing shared MySQL support, Apache dumps core as soon as libphp4.so is loaded. Can this be fixed?

If your MySQL libs are linked against pthreads this will happen. Check using ldd. If they are, grab the MySQL tarball and compile from source, or recompile from the source rpm and remove the switch in the spec file that turns on the threaded client code. Either of these suggestions will fix this. Then recompile PHP with the new MySQL libs.

5. Why do I get an error that looks something like this: "Warning: 0 is not a MySQL result index in <file> on line <x>" or "Warning: Supplied argument is not a valid MySQL result resource in <file> on line <x>?

You are trying to use a result identifier that is 0. The 0 indicates that your query failed for some reason. You need to check for errors after submitting a query and before you attempt to use the returned result identifier. The proper way to do this is with code similar to the following:

```
$result = mysql_query("SELECT * FROM tables_priv");
if (!$result) {
    echo mysql_error();
    exit;
}
```

or

```
$result = mysql_query("SELECT * FROM tables_priv")
    or die("Bad query: ".mysql_error());
```

Chapter 48. Installation

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 the INSTALL file located in the distribution. Windows users should also read the install.txt file. There are also some helpful hints for Windows users here.

1. 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-dist` from the distribution to `/etc/php.ini` and edit it to make any local changes you want.

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.

2. 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

3. 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 `http.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 php3_module        modules/libphp3.so     /* for PHP 3 */
LoadModule php4_module        modules/libphp4.so     /* for PHP 4 */
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.

4. 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.

5. 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.

6. 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.

7. 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.

8. Some operating systems: I have installed PHP without errors, but when I try to start apache I get undefined symbol errors:
```
[mybox:user /src/php4] 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.

9. 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`, `php4ts.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.

10. 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.

11. 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.

12. 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 PHP after making changes to `php.ini`

Chapter 49. Build Problems

This section gathers most common errors that occur at build time.

1. I got the latest version of PHP using the anonymous CVS service, but there's no configure script!

You have to have the GNU autoconf package installed so you can generate the configure script from `configure.in`. Just run **./buildconf** in the top-level directory after getting the sources from the CVS server. (Also, unless you run configure with the `--enable-maintainer-mode` option, the configure script will not automatically get rebuilt when the `configure.in` file is updated, so you should make sure to do that manually when you notice configure.in has changed. One symptom of this is finding things like @VARIABLE@ in your Makefile after configure or `config.status` is run.)

2. I'm having problems configuring PHP to work with Apache. It says it can't find `httpd.h`, but it's right where I said it is!

You need to tell the configure/setup script the location of the top-level of your Apache source tree. This means that you want to specify `--with-apache=/path/to/apache` and *not* `--with-apache=/path/to/apache/src`.

3. While configuring PHP (`./configure`), you come across an error similar to the following:

```
checking lex output file root... ./configure: lex: command not found
       configure: error: cannot find output from lex; giving up
```

Be sure to read the installation instructions carefully and note that you need both flex and bison installed to compile PHP. Depending on your setup you will install bison and flex from either source or a package, such as a RPM.

4. When I try to start Apache, I get the the following message:

```
fatal: relocation error: file /path/to/libphp4.so:
       symbol ap_block_alarms: referenced symbol not found
```

This error usually comes up when one compiles the Apache core program as a DSO library for shared usage. Try to reconfigure apache, making sure to use at least the following flags:

```
--enable-shared=max --enable-rule=SHARED_CORE
```

For more information, read the top-level Apache `INSTALL` file or the Apache DSO manual page.

5. When I run configure, it says that it can't find the include files or library for GD, gdbm, or some other package!

You can make the configure script looks for header files and libraries in non-standard locations by specifying additional flags to pass to the C preprocessor and linker, such as:

```
CPPFLAGS=-I/path/to/include LDFLAGS=-L/path/to/library ./configure
```

If you're using a csh-variant for your login shell (why?), it would be:

```
env CPPFLAGS=-I/path/to/include LDFLAGS=-L/path/to/library ./configure
```

6. When it is compiling the file `language-parser.tab.c`, it gives me errors that say `yytname undeclared`.

You need to update your version of Bison. You can find the latest version at ftp://ftp.gnu.org/pub/gnu/bison/.

7. When I run make, it seems to run fine but then fails when it tries to link the final application complaining that it can't find some files.

Some old versions of make that don't correctly put the compiled versions of the files in the functions directory into that same directory. Try running **cp *.o functions** and then re-running **make** to see if that helps. If it does, you should really upgrade to a recent version of GNU make.

8. When linking PHP, it complains about a number of undefined references.

Take a look at the link line and make sure that all of the appropriate libraries are being included at the end. Common ones that you might have missed are '-ldl' and any libraries required for any database support you included.

If you're linking with Apache 1.2.x, did you remember to add the appropriate information to the EXTRA_LIBS line of the Configuration file and re-rerun Apache's Configure script? See the INSTALL file that comes with the distribution for more information.

Some people have also reported that they had to add '-ldl' immediately following `libphp4.a` when linking with Apache.

9. I can't figure out how to build PHP with Apache 1.3.

This is actually quite easy. Follow these steps carefully:

- Grab the latest Apache 1.3 distribution from http://www.apache.org/dist/.
- Ungzip and untar it somewhere, for example `/usr/local/src/apache-1.3`.
- Compile PHP by first running **./configure --with-apache=/<path>/apache-1.3** (substitute <path> for the actual path to your apache-1.3 directory.
- Type **make** followed by **make install** to build PHP and copy the necessary files to the Apache distribution tree.
- Change directories into to your `/<path>/apache-1.3/src` directory and edit the `Configuration` file. Add to the file: `AddModule modules/php4/libphp4.a`.
- Type: **./Configure** followed by `make`.
- You should now have a PHP-enabled httpd binary!

Note: You can also use the new Apache `./configure` script. See the instructions in the `README.configure` file which is part of your Apache distribution. Also have a look at the `INSTALL` file in the PHP distribution.

10. I have followed all the steps to install the Apache module version on UNIX, and my PHP scripts show up in my browser or I am being asked to save the file.

This means that the PHP module is not getting invoked for some reason. Three things to check before asking for further help:

- Make sure that the httpd binary you are running is the actual new httpd binary you just built. To do this, try running: `/path/to/binary/httpd -l`

  If you don't see `mod_php4.c` listed then you are not running the right binary. Find and install the correct binary.

- Make sure you have added the correct Mime Type to one of your `Apache .conf` files. It should be: `AddType application/x-httpd-php3 .php3` (for PHP 3)

  or `AddType application/x-httpd-php .php` (for PHP 4)

  Also make sure that this AddType line is not hidden away inside a <Virtualhost> or <Directory> block which would prevent it from applying to the location of your test script.

- Finally, the default location of the Apache configuration files changed between Apache 1.2 and Apache 1.3. You should check to make sure that the configuration file you are adding the AddType line to is actually being read. You can put an obvious syntax error into your httpd.conf file or some other obvious change that will tell you if the file is being read correctly.

11. It says to use: `--activate-module=src/modules/php4/libphp4.a`, but that file doesn't exist, so I changed it to `--activate-module=src/modules/php4/libmodphp4.a` and it doesn't work!? What's going on?

Note that the `libphp4.a` file is not supposed to exist. The apache process will create it!

12. When I try to build Apache with PHP as a static module using `--activate-module=src/modules/php4/libphp4.a` it tells me that my compiler is not ANSI compliant.

This is a misleading error message from Apache that has been fixed in more recent versions.

13. When I try to build PHP using `--with-apxs` I get strange error messages.

There are three things to check here. First, for some reason when Apache builds the apxs Perl script, it sometimes ends up getting built without the proper compiler and flags variables. Find your apxs script (try the command **which apxs**, it's sometimes found in `/usr/local/apache/bin/apxs` or `/usr/sbin/apxs`. Open it and check for lines similar to these:

```
my $CFG_CFLAGS_SHLIB  = ' ';          # substituted via Makefile.tmpl
my $CFG_LD_SHLIB      = ' ';          # substituted via Makefile.tmpl
my $CFG_LDFLAGS_SHLIB = ' ';          # substituted via Makefile.tmpl
```

If this is what you see, you have found your problem. They may contain just spaces or other incorrect values, such as 'q()'. Change these lines to say:

```
my $CFG_CFLAGS_SHLIB  = '-fpic -DSHARED_MODULE'; # substituted via Makefile.tmpl
my $CFG_LD_SHLIB      = 'gcc';                   # substituted via Makefile.tmpl
my $CFG_LDFLAGS_SHLIB = q(-shared);              # substituted via Makefile.tmpl
```

The second possible problem should only be an issue on Red Hat 6.1 and 6.2. The apxs script Red Hat ships is broken. Look for this line:

```
my $CFG_LIBEXECDIR    = 'modules';         # substituted via APACI install
```

If you see the above line, change it to this:

```
my $CFG_LIBEXECDIR    = '/usr/lib/apache'; # substituted via APACI install
```

Last, if you reconfigure/reinstall Apache, add a **make clean** to the process after **./configure** and before **make**.

14. During make, I get errors in microtime, and a lot of `RUSAGE_` stuff.

During the **make** portion of installation, if you encounter problems that look similar to this:

```
microtime.c: In function `php_if_getrusage':
microtime.c:94: storage size of `usg' isn't known
microtime.c:97: `RUSAGE_SELF' undeclared (first use in this function)
microtime.c:97: (Each undeclared identifier is reported only once
microtime.c:97: for each function it appears in.)
microtime.c:103: `RUSAGE_CHILDREN' undeclared (first use in this function)
make[3]: *** [microtime.lo] Error 1
make[3]: Leaving directory `/home/master/php-4.0.1/ext/standard'
make[2]: *** [all-recursive] Error 1
make[2]: Leaving directory `/home/master/php-4.0.1/ext/standard'
make[1]: *** [all-recursive] Error 1
make[1]: Leaving directory `/home/master/php-4.0.1/ext'
make: *** [all-recursive] Error 1
```

Your system is broken. You need to fix your `/usr/include` files by installing a glibc-devel package that matches your glibc. This has absolutely nothing to do with PHP. To prove this to yourself, try this simple test:

```
$ cat >test.c <<X
#include <sys/resource.h>
X
$ gcc -E test.c >/dev/null
```

If that spews out errors, you know your include files are messed up.

15. I want to upgrade my PHP. Where can I find the ./configure line that was used to build my current PHP installation?

Either you look at config.nice file, in the source tree of your current PHP installation or, if this is not available, you simply run a

```
<?php phpinfo(); ?>
```

script. On top of the output the **./configure** line, that was used to build this PHP installation is shown.

16. When building PHP with the GD library it either gives strange compile errors or segfaults on execution.

Make sure your GD library and PHP are linked against the same depending libraries (e.g. libpng).

Chapter 50. Using PHP

This section gathers most common errors you can face, while writing PHP scripts.

1. I would like to write a generic PHP script that can handle data coming from any form. How do I know which POST method variables are available?

```
function myfunc($argument)
{
    echo $argument + 10;
}
$variable = 10;
echo "myfunc($variable) = " . myfunc($variable);
```



```
<pre>
<?php echo "This should be the first line."; ?>
<?php echo "This should show up after the new line above."; ?>
</pre>
```


1. I would like to write a generic PHP script that can handle data coming from any form. How do I know which POST method variables are available?

Make sure that the track_vars feature is enabled in your `php.ini` file. Since PHP 4.0.3, this feature is always on. When `track_vars` is on, it creates some associative arrays, the most important here is: `$_POST` (this used to be called `$HTTP_POST_VARS` in `PHP` versions prior 4.1.0). So, to write a generic script to handle POST method variables you would need something similar to the following:

```
foreach ($_POST as $var => $value) {
    echo "$var = $value<br>\n";
}
```

2. I need to convert all single-quotes (') to a backslash followed by a single-quote. How can I do this with a regular expression?

First off, take a look at the **addslashes()** function. It will do exactly what you want. You should also have a look at the magic_quotes_gpc directive in your `php.ini` file.

3. When I do the following, the output is printed in the wrong order:

```
function myfunc($argument)
{
    echo $argument + 10;
}
$variable = 10;
echo "myfunc($variable) = " . myfunc($variable);
```

what's going on?

To be able to use the results of your function in an expression (such as concatenating it with other strings in the example above), you need to **return()** the value, not **echo()** it.

4. Hey, what happened to my newlines?

```
<pre>
<?php echo "This should be the first line."; ?>
<?php echo "This should show up after the new line above."; ?>
</pre>
```

In PHP, the ending for a block of code is either "?>" or "?>\n" (where \n means a newline). So in the example above, the echoed sentences will be on one line, because PHP omits the newlines after the block ending. This means that you need to insert an extra newline after each block of PHP code to make it print out one newline.

Why does PHP do this? Because when formatting normal HTML, this usually makes your life easier because you don't want that newline, but you'd have to create extremely long lines or otherwise make the raw page source unreadable to achieve that effect.

5. I get the message 'Warning: Cannot send session cookie - headers already sent...' or 'Cannot add header information - headers already sent...'.

The functions **header()**, **setcookie()** and the session functions need to add headers to the output stream. But headers can only be sent before all other content, check if your script is sending headers after having already sent content.

6. I need to access information in the request header directly. How can I do this?

The **getallheaders()** function will do this if you are running PHP as an Apache module. So, the following bit of code will show you all the request headers:

```
$headers = getallheaders();
foreach ($headers as $name => $content) {
    echo "headers[$name] = $content<br>\n";
}
```

7. When I try to use authentication with IIS I get 'No Input file specified'.

The security model of IIS is at fault here. This is a problem common to all CGI programs running under IIS. A workaround is to create a plain HTML file (not parsed by PHP) as the entry page into an authenticated directory. Then use a META tag to redirect to the PHP page, or have a link to the PHP page. PHP will then recognize the authentication correctly. With the ISAPI module, this is not a problem. This should not effect other NT web servers. For more information, see: http://support.microsoft.com/support/kb/articles/q160/4/22.asp.

8. My PHP script works on IE and Lynx, but on Netscape some of my output is missing. When I do a "View Source" I see the content in IE but not in Netscape.

Netscape is more strict regarding html tags (such as tables) then IE. Running your html output through a html validator, such as validator.w3.org, might be helpful. For example, a missing </table> might cause this.

Also, both IE and Lynx ignore any NULs (`\0`) in the HTML stream, Netscape does not. The best way to check for this is to compile the command line version of PHP (also known as the CGI version) and run your script from the command line. In *nix, pipe it through `od -c` and look for any `\0` characters. If you are on Windows you need to find an editor or some other program that lets you look at binary files. When Netscape sees a NUL in a file it will typically not output anything else on that line whereas both IE and Lynx will.

9. How am I supposed to mix XML and PHP? It complains about my <?xml> tags!

You need to turn off the short tags by setting short_tags to 0 in your `php.ini` file, or by using the appropriate Apache directive. You could even use a <File> section to do this selectively.

10. How can I use PHP with FrontPage or some other HTML editor that insists on moving my code around?

One of the easiest things to do is to enable using ASP tags in your PHP code. This allows you to use the ASP-style <% and %> code delimiters. Some of the popular HTML editors handle those more intelligently (for now). To enable the ASP-style tags, you need to set the asp_tags `php.ini` variable, or use the appropriate Apache directive.

11. Where can I find a complete list of pre-set variables available to me, and why are these not documented in the PHP documentation?

The best way is to stick a `<?php phpinfo(); ?>` part on a page and load it up. This will show you all sorts of information about your PHP setup, including a list of both environment variables and also special variables set by your web server. This list can't really be documented in the PHP documentation because it will change from one server to another.

12. I'm trying to access one of the standard CGI variables (such as $DOCUMENT_ROOT or $HTTP_REFERER) in a user-defined function, and it can't seem to find it. What's wrong?

Environment variables are normal global variables, so you must either declare them as global variables in your function (by using "`global $DOCUMENT_ROOT;`", for example) or by using the global variable array (ie, "`$GLOBALS["DOCUMENT_ROOT"]`").

> **Note:** Since PHP 4.1.0 you can also use the superglobal array `$_SERVER` which is available in every function. For example, you can now use `$_SERVER["DOCUMENT_ROOT"]` instead of `$DOCUMENT_ROOT`.

Chapter 51. PHP and HTML

PHP and HTML interact a lot: PHP can generate HTML, and HTML can pass information to PHP. Before reading these faqs, it's important you learn how to retrieve variables from outside of PHP. The manual page on this topic includes many examples as well. Pay close attention to what `register_globals` means to you too.

1. What encoding/decoding do I need when I pass a value through a form/URL?

There are several stages for which encoding is important. Assuming that you have a **string** `$data`, which contains the string you want to pass on in a non-encoded way, these are the relevant stages:

- HTML interpretation. In order to specify a random string, you *must* include it in double quotes, and **htmlspecialchars()** the whole value.
- URL: A URL consists of several parts. If you want your data to be interpreted as one item, you *must* encode it with **urlencode()**.

Example 51-1. A hidden HTML form element

```
<?php
    echo "<input type=hidden value=\"" . htmlspecialchars($data) . "\">\n";
?>
```

Note: It is wrong to **urlencode()** `$data`, because it's the browsers responsibility to **urlencode()** the data. All popular browsers do that correctly. Note that this will happen regardless of the method (i.e., GET or POST). You'll only notice this in case of GET request though, because POST requests are usually hidden.

Example 51-2. Data to be edited by the user

```
<?php
    echo "<textarea name=mydata>\n";
    echo htmlspecialchars($data)."\n";
    echo "</textarea>";
?>
```

Note: The data is shown in the browser as intended, because the browser will interpret the HTML escaped symbols.

Upon submitting, either via GET or POST, the data will be urlencoded by the browser for transferring, and directly urldecoded by PHP. So in the end, you don't need to do any urlencoding/urldecoding yourself, everything is handled automagically.

Example 51-3. In an URL

```
<?php
    echo "<a href=\"" . htmlspecialchars("/nextpage.php?stage=23&data=" .
        urlencode($data)) . "\">\n";
?>
```

Note: In fact you are faking a HTML GET request, therefore it's necessary to manually **urlencode()** the data.

Note: You need to **htmlspecialchars()** the whole URL, because the URL occurs as value of an HTML-attribute. In this case, the browser will first un-**htmlspecialchars()** the value, and then pass the URL on. PHP will understand the URL correctly, because you **urlencoded()** the data.

You'll notice that the `&` in the URL is replaced by `&`. Although most browsers will recover if you forget this, this isn't always possible. So even if your URL is not dynamic, you *need* to **htmlspecialchars()** the URL.

2. I'm trying to use an <input type="image"> tag, but the `$foo.x` and `$foo.y` variables aren't available. `$_GET['foo.x']` isn't existing either. Where are they?

When submitting a form, it is possible to use an image instead of the standard submit button with a tag like:

```
<input type="image" src="image.gif" name="foo">
```

When the user clicks somewhere on the image, the accompanying form will be transmitted to the server with two additional variables: `foo.x` and `foo.y`.

Because `foo.x` and `foo.y` would make invalid variable names in PHP, they are automagically converted to `foo_x` and `foo_y`. That is, the periods are replaced with underscores. So, you'd access these variables like any other described within the section on retrieving variables from outside of PHP. For example, `$_GET['foo_x']`.

3. How do I create arrays in a HTML <form>?

To get your <form> result sent as an array to your PHP script you name the <input>, <select> or <textarea> elements like this:

```
<input name="MyArray[]">
<input name="MyArray[]">
<input name="MyArray[]">
<input name="MyArray[]">
```

Notice the square brackets after the variable name, that's what makes it an array. You can group the elements into different arrays by assigning the same name to different elements:

```
<input name="MyArray[]">
<input name="MyArray[]">
<input name="MyOtherArray[]">
<input name="MyOtherArray[]">
```

This produces two arrays, MyArray and MyOtherArray, that gets sent to the PHP script. It's also possible to assign specific keys to

your arrays:

```
<input name="AnotherArray[]">
<input name="AnotherArray[]">
<input name="AnotherArray[email]">
<input name="AnotherArray[phone]">
```

The AnotherArray array will now contain the keys 0, 1, email and phone.

> **Note:** Specifying an arrays key is optional in HTML. If you do not specify the keys, the array gets filled in the order the elements appear in the form. Our first example will contain keys 0, 1, 2 and 3.

See also Array Functions and Variables from outside PHP.

4. How do I get all the results from a select multiple HTML tag?

The select multiple tag in an HTML construct allows users to select multiple items from a list. These items are then passed to the action handler for the form. The problem is that they are all passed with the same widget name. ie.

```
<select name="var" multiple>
```

Each selected option will arrive at the action handler as:

```
var=option1
var=option2
var=option3
```

Each option will overwrite the contents of the previous `$var` variable. The solution is to use PHP's "array from form element" feature. The following should be used:

```
<select name="var[]" multiple>
```

This tells PHP to treat `$var` as an array and each assignment of a value to var[] adds an item to the array. The first item becomes `$var[0]`, the next `$var[1]`, etc. The **count()** function can be used to determine how many options were selected, and the **sort()** function can be used to sort the option array if necessary.

Note that if you are using JavaScript the `[]` on the element name might cause you problems when you try to refer to the element by name. Use it's numerical form element ID instead, or enclose the variable name in single quotes and use that as the index to the elements array, for example:

```
variable = documents.forms[0].elements['var[]'];
```

Chapter 52. PHP and COM

PHP can be used to access COM and DCOM objects on Win32 platforms.

1. I have built a DLL to calculate something. Is there any way to run this DLL under PHP ?

If this is a simple DLL there is no way yet to run it from PHP. If the DLL contains a COM server you may be able to access it if it implements the IDispatch interface.

2. What does 'Unsupported variant type: xxxx (oxxxxx)' mean ?

There are dozens of VARIANT types and combinations of them. Most of them are already supported but a few still have to be implemented. Arrays are not completely supported. Only single dimensional indexed only arrays can be passed between PHP and COM. If you find other types that aren't supported, please report them as a bug (if not already reported) and provide as much information as available.

3. Is it possible manipulate visual objects in PHP ?

Generally it is, but as PHP is mostly used as a web scripting language it runs in the web servers context, thus visual objects will never appear on the servers desktop. If you use PHP for application scripting e.g. in conjunction with PHP-GTK there is no limitation in accessing and manipulating visual objects through COM.

4. Can I store a COM object in a session ?

No, you can't. COM instances are treated as resources and therefore they are only available in a single script's context.

5. How can I trap COM errors ?

Currently it's not possible to trap COM errors beside the ways provided by PHP itself (@, track_errors, ..), but we are thinking of a way to implement this.

6. Can I generate DLL files from PHP scripts like i can in Perl ?

No, unfortunately there is no such tool available for PHP.

7. What does 'Unable to obtain IDispatch interface for CLSID {xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}' mean ?

This error can have multiple reasons:

- the CLSID is wrong
- the requested DLL is missing
- the requested component doesn't implement the IDispatch interface

8. How can I run COM object from remote server ?

Exactly like you run local objects. You only have to pass the IP of the remote machine as second parameter to the COM constructor.

Make sure that you have set `com.allow_dcom=true` in your `php.ini`.

9. I get 'DCOM is disabled in C:\path...\scriptname.php on line 6', what can I do ?

Edit your `php.ini` and set `com.allow_dcom=true`.

10. Is it possible to load/manipulate an ActiveX object in a page with PHP ?

This has nothing to do with PHP. ActiveX objects are loaded on client side if they are requested by the HTML document. There is no relation to the PHP script and therefore there is no direct server side interaction possible.

11. Is it possible to get a running instance of a component ?

This is possible with the help of monikers. If you want to get multiple references to the same word instance you can create that instance like shown:

```
$word = new COM("C:\docs\word.doc");
```

This will create a new instance if there is no running instance available or it will return a handle to the running instance, if available.

12. Is there a way to handle an event sent from COM object ?

Starting in PHP 4.3.0, you can define an event sink and bind it as shown in the example below. You can use **com_print_typeinfo()** to have PHP generate a skeleton for the event sink class.

Example 52-1. COM event sink example

```
<?php
class IEEventSinker {
  var $terminated = false;

  function ProgressChange($progress, $progressmax) {
    echo "Download progress: $progress / $progressmax\n";
  }

  function DocumentComplete(&$dom, $url) {
    echo "Document $url complete\n";
  }

  function OnQuit() {
    echo "Quit!\n";
    $this->terminated = true;
  }
}
$ie = new COM("InternetExplorer.Application");
```

```
$sink =& new IEEventSinker();
com_event_sink($ie, $sink, "DWebBrowserEvents2");
$ie->Visible = true;
$ie->Navigate("http://www.php.net");
while(!$sink->terminated) {
  com_message_pump(4000);
}
$ie = null;
?>
```

13. I'm having problems when trying to invoke a method of a COM object which exposes more than one interface. What can I do ?

The answer is as simple as unsatisfying. I don't know exactly but i think you can do nothing. If someone has specific information about this, please let me know :)

14. So PHP works with COM, how about COM+ ?

COM+ extends COM by a framework for managing components through MTS and MSMQ but there is nothing special that PHP has to support to use such components.

15. If PHP can manipulate COM objects, can we imagine to use MTS to manage components resources, in conjunction with PHP ?

PHP itself doesn't handle transactions yet. Thus if an error occurs no rollback is initiated. If you use components that support transactions you will have to implement the transaction management yourself.

Chapter 53. PHP and other languages

PHP is the best language for web programing, but what about other languages?

1. PHP vs. ASP?
2. Is there an ASP to PHP converter?
3. PHP vs. Cold Fusion?
4. PHP vs. Perl?

1. PHP vs. ASP?

ASP is not really a language in itself, it's an acronym for Active Server Pages, the actual language used to program ASP with is Visual Basic Script or JScript. The biggest drawback of ASP is that it's a proprietary system that is natively used only on Microsoft Internet Information Server (IIS). This limits it's availability to Win32 based servers. There are a couple of projects in the works that allows ASP to run in other environments and webservers: InstantASP from Halcyon (commercial), Chili!Soft ASP from Chili!Soft (commercial) and OpenASP from ActiveScripting.org (free). ASP is said to be a slower and more cumbersome language than PHP, less stable as well. Some of the pros of ASP is that since it primarily uses VBScript it's relatively easy to pick up the language if you're already know how to program in Visual Basic. ASP support is also enabled by default in the IIS server making it easy to get up and running. The components built in ASP are really limited, so if you need to use "advanced" features like interacting with FTP servers, you need to buy additional components.

2. Is there an ASP to PHP converter?

Yes, asp2php is the one most often referred to.

3. PHP vs. Cold Fusion?

PHP is commonly said to be faster and more efficient for complex programming tasks and trying out new ideas. PHP is generally referred to as more stable and less resource intensive as well. Cold Fusion has better error handling, database abstraction and date parsing although database abstraction is addressed in PHP 4. Another thing that is listed as one of Cold Fusion's strengths is its excellent search engine, but it has been mentioned that a search engine is not something that should be included in a web scripting language. PHP runs on almost every platform there is; Cold Fusion is only available on Win32, Solaris, Linux and HP/UX. Cold Fusion has a good IDE and is generally easier to get started with, whereas PHP initially requires more programming knowledge. Cold Fusion is designed with non-programmers in mind, while PHP is focused on programmers.

A great summary by Michael J Sheldon on this topic has been posted to the PHP mailing list. A copy can be found here.

4. PHP vs. Perl?

The biggest advantage of PHP over Perl is that PHP was designed for scripting for the web where Perl was designed to do a lot more and can because of this get very complicated. The flexibility / complexity of Perl makes it easier to write code that another author / coder has a hard time reading. PHP has a less confusing and stricter format without losing flexibility. PHP is easier to integrate into existing HTML than Perl. PHP has pretty much all the 'good' functionality of Perl: constructs, syntax and so on, without making it as complicated as Perl can be. Perl is a very tried and true language, it's been around since the late eighties,

but PHP is maturing very quickly.

Chapter 54. Migrating from PHP 2 to PHP 3

PHP has already a long history behind him: Legendary PHP 1.0, PHP/FI, PHP 3.0 and PHP 4.0.

1. Migrating from PHP 2 to PHP 3?

PHP/FI 2.0 is no longer supported. Please see appropriate manual section for information about migration from PHP/FI 2.0.

If you are still working with PHP 2, we *strongly* recommend you to upgrade straight to PHP 4.

Chapter 55. Migrating from PHP 3 to PHP 4

PHP has already a long history behind him : Legendary PHP 1.0, PHP/FI, PHP 3.0 and PHP 4.0.

1. Migrating from PHP3 to PHP4

PHP 4 was designed to be as compatible with earlier versions of PHP as possible and very little functionality was broken in the process. If you're really unsure about compatibility you should install PHP 4 in a test environment and run your scripts there.

Also see the appropriate migration appendix of this manual.

2. Incompatible functions?

Since PHP 4 is basically a rewrite of the entire PHP engine there was very few functions that were altered and only then some of the more exotic ones.

Chapter 56. Miscellaneous Questions

There can be some questions we can't put into other categories. Here you can find them.

1. Where did the pop-ups go on the website? Can I have the code for that?

The yellow pop-up windows on the old site were pretty cool, but were very difficult to maintain (since some companies seem to enjoy changing the way their browsers work with every new release).

All the code for previous versions of the website is still available through CVS. Specifically, the last version of shared.inc (that had all the Javascript and DHTML to do the popups) is available here.

2. How can I handle the bz2 compressed manuals on Windows?

If you don't have an archiver-tool to handle bz2 files download the commandline tool from Redhat (please find further information below).

If you would not like to use a command line tool, you can try free tools like Stuffit Expander, UltimateZip, 7-Zip, or Quick Zip. If you have tools like WinRAR or Power Archiver, you can easily decompress the bz2 files with it. If you use Windows Commander, a bz2 plugin for that program is available freely from the Windows Commander site.

The bzip2 commandline tool from Redhat:

Win2k Sp2 users grab the latest version 1.0.2, all other Windows user should grab version 1.00. After downloading rename the executable to bzip2.exe. For convenience put it into a directory in your path, e.g. C:\Windows where C represents your windows installation drive.

Note: lang stands for your language and x for the desired format, e.g.: pdf. To uncompress the php_manual_lang.x.bz2 follow

these simple instructions:

- open a command prompt window
- cd to the folder where you stored the downloaded php_manual_lang.x.bz2
- invoke bzip2 -d php_manual_lang.x.bz2, extracting php_manual_lang.x in the same folder

In case you downloaded the php_manual_lang.tar.bz2 with many html-files in it, the procedure is the same. The only difference is that you got a file php_manual_lang.tar. The tar format is known to be treated with most common archivers on Windows like e.g. WinZip.

VII. Appendixes

Table of Contents

Appendix A. History of PHP and related projects

PHP has come a long way in the last few years. Growing to be one of the most prominent languages powering the Web was not an easy task. Those of you interested in briefly seeing how PHP grew out to what it is today, read on.

History of PHP

PHP/FI

PHP succeeds an older product, named PHP/FI. PHP/FI was created by Rasmus Lerdorf in 1995, initially as a simple set of Perl scripts for tracking accesses to his online resume. He named this set of scripts 'Personal Home Page Tools'. As more functionality was required, Rasmus wrote a much larger C implementation, which was able to communicate with databases, and enabled users to develop simple dynamic Web applications. Rasmus chose to release the source code for PHP/FI for everybody to see, so that anybody can use it, as well as fix bugs in it and improve the code.

PHP/FI, which stood for Personal Home Page / Forms Interpreter, included some of the basic functionality of PHP as we know it today. It had Perl-like variables, automatic interpretation of form variables and HTML embedded syntax. The syntax itself was similar to that of Perl, albeit much more limited, simple, and somewhat inconsistent.

By 1997, PHP/FI 2.0, the second write-up of the C implementation, had a cult of several thousand users around the world (estimated), with approximately 50,000 domains reporting as having it installed, accounting for about 1% of the domains on the Internet. While there were several people contributing bits of code to this project, it was still at large a one-man project.

PHP/FI 2.0 was officially released only in November 1997, after spending most of its life in beta releases. It was shortly afterwards succeeded by the first alphas of PHP 3.0.

PHP 3

PHP 3.0 was the first version that closely resembles PHP as we know it today. It was created by Andi Gutmans and Zeev Suraski in 1997 as a complete rewrite, after they found PHP/FI 2.0 severely underpowered for developing their own eCommerce application. In an effort to cooperate and start building upon PHP/FI's existing user-base, Andi, Rasmus and Zeev decided to cooperate and announce PHP 3.0 as the official successor of PHP/FI 2.0, and development of PHP/FI 2.0 was mostly halted.

One of the biggest strengths of PHP 3.0 was its strong extensibility features. In addition to providing end users with a solid

infrastructure for lots of different databases, protocols and APIs, PHP 3.0's extensibility features attracted dozens of developers to join in and submit new extension modules. Arguably, this was the key to PHP 3.0's tremendous success. Other key features introduced in PHP 3.0 were the object oriented syntax support and the much more powerful and consistent language syntax.

The whole new language was released under a new name, that removed the implication of limited personal use that the PHP/FI 2.0 name held. It was named plain 'PHP', with the meaning being a recursive acronym - PHP: Hypertext Preprocessor.

By the end of 1998, PHP grew to an install base of tens of thousands of users (estimated) and hundreds of thousands of Web sites reporting it installed. At its peak, PHP 3.0 was installed on approximately 10% of the Web servers on the Internet.

PHP 3.0 was officially released in June 1998, after having spent about 9 months in public testing.

PHP 4

By the winter of 1998, shortly after PHP 3.0 was officially released, Andi Gutmans and Zeev Suraski had begun working on a rewrite of PHP's core. The design goals were to improve performance of complex applications, and improve the modularity of PHP's code base. Such applications were made possible by PHP 3.0's new features and support for a wide variety of third party databases and APIs, but PHP 3.0 was not designed to handle such complex applications efficiently.

The new engine, dubbed 'Zend Engine' (comprised of their first names, Zeev and Andi), met these design goals successfully, and was first introduced in mid 1999. PHP 4.0, based on this engine, and coupled with a wide range of additional new features, was officially released in May 2000, almost two years after its predecessor, PHP 3.0. In addition to the highly improved performance of this version, PHP 4.0 included other key features such as support for many more Web servers, HTTP sessions, output buffering, more secure ways of handling user input and several new language constructs.

PHP 4 is currently the latest released version of PHP. Work has already begun on modifying and improving the Zend Engine to integrate the features which were designed for PHP 5.0.

Today, PHP is being used by hundreds of thousands of developers (estimated), and several million sites report as having it installed, which accounts for over 20% of the domains on the Internet.

PHP's development team includes dozens of developers, as well as dozens others working on PHP-related projects such as PEAR and the documentation project.

PHP 5

The future of PHP is mainly driven by it's core, the Zend Engine. PHP 5 will include the new Zend Engine 2.0. To get more information on this engine, see it's webpage.

History of PHP related projects

PEAR

PEAR, the PHP Extension and Application Repository (originally, PHP Extension and Add-on Repository) is PHP's version of foundation classes, and may grow in the future to be one of the key ways to distribute both PHP and C-based PHP extensions among developers.

PEAR was born in discussions held in the PHP Developers' Meeting (PDM) held in January 2000 in Tel Aviv. It was created by Stig S. Bakken, and is dedicated to his first-born daughter, Malin Bakken.

Since early 2000, PEAR has grown to be a big, significant project with a large number of developers working on implementing common, reusable functionality for the benefit of the entire PHP community. PEAR today includes a wide variety of infrastructure foundation classes for database access, content caching, mathematical calculations, eCommerce and much more.

PHP Quality Assurance Initiative

The PHP Quality Assurance Initiative was set up in the summer of 2000 in response to criticism that PHP releases were not being tested well enough for production environments. The team now consists of a core group of developers with a good understanding of the PHP code base. These developers spend a lot of their time localizing and fixing bugs within PHP. In addition there are many other team members who test and provide feedback on these fixes using a wide variety of platforms.

PHP-GTK

PHP-GTK is the PHP solution for writing client side GUI applications. Andrei Zmievski remembers the planing and creation process of PHP-GTK:

> GUI programming has always been of my interests, and I found that Gtk+ is a very nice toolkit, except that programming with it in C is somewhat tedious. After witnessing PyGtk and GTK-Perl implementations, I decided to see if PHP could be made to interface with Gtk+, even minimally. Starting in August of 2000, I began to have a bit more free time so that is when I started experimenting. My main guideline was the PyGtk implementation as it was fairly feature complete and had a nice object-oriented interface. James Henstridge, the author of PyGtk, provided very helpful advice during those initial stages.
>
> Hand-writing the interfaces to all the Gtk+ functions was out of the question, so I seized upon the idea of code-generator, similar to how PyGtk did it. The code generator is a PHP program that reads a set of .defs file containing the Gtk+ classes, constants, and methods information and generates C code that interfaces PHP with them. What cannot be generated automatically can be written by hand in .overrides file.
>
> Working on the code generator and the infrastructure took some time, because I could spend little time on PHP-GTK during the fall of 2000. After I showed PHP-GTK to Frank Kromann, he got interested and started helping me out with code generator work and Win32 implementation. When we wrote the first Hello World program and fired it up, it was extremely exciting. It took a couple more months to get the project to a presentable condition and the initial version was released on March 1, 2001. The story promptly hit SlashDot.
>
> Sensing that PHP-GTK might be extensive, I set up separate mailing lists and CVS repositories for it, as well as the gtk.php.net website with the help of Colin Viebrock. The documentation would also need to be done and James Moore came in to help with that.
>
> Since its release PHP-GTK has been gaining popularity. We have our own documentation team, the manual keeps improving, people start writing extensions for PHP-GTK, and more and more exciting applications with it.

Books about PHP

As PHP grew, it began to be recognized as a world-wide popular development platform. One of the most interesting ways of seeing this trend was by observing the books about PHP that came out throughout the years.

To the best of our knowledge, the first book dedicated to PHP was 'PHP - tvorba interaktivních internetových aplikací' - a Czech book published in April 1999, authored by Jirka Kosek. Next month followed a German book authored by Egon Schmid, Christian Cartus and Richard Blume. The first book in English about PHP was published shortly afterwards, and was 'Core PHP Programming' by Leon Atkinson. Both of these books covered PHP 3.0.

While these books were the first of their kind - they were followed by a large number of books from a host of authors and publishers. There are over 40 books in English, 50 books in German, and over 20 books in French! In addition, you can find books about PHP in many other languages, including Spanish, Korean, Japanese and Hebrew.

Clearly, this large number of books, written by different authors, published by many publishers, and their availability in so many languages - are a strong testimony for PHP's world-wide success.

Publications about PHP

To the best of our knowledge, the first article about PHP in a hard-copy magazine was published in the Czech mutation of Computerworld in the spring of 1998, and covered PHP 3.0. As with books, this was the first in a series of many articles published about PHP in various prominent magazines.

Articles about PHP appeared in Dr. Dobbs, Linux Enterprise, Linux Magazine and many more. Articles about migrating ASP-based applications to PHP under Windows even appear on Microsoft's very own MSDN!

Appendix B. Migrating from PHP 3 to PHP 4

What has changed in PHP 4

PHP 4 and the integrated Zend engine have greatly improved PHP's performance and capabilities, but great care has been taken to break as little existing code as possible. So migrating your code from PHP 3 to 4 should be much easier than migrating from

PHP/FI 2 to PHP 3. A lot of existing PHP 3 code should be ready to run without changes, but you should still know about the few differences and take care to test your code before switching versions in production environments. The following should give you some hints about what to look for.

Running PHP 3 and PHP 4 concurrently

Recent operating systems provide the ability to perform versioning and scoping. This features make it possible to let PHP 3 and PHP 4 run as concurrent modules in one Apache server.

This feature is known to work on the following platforms:

- Linux with recent binutils (binutils 2.9.1.0.25 tested)
- Solaris 2.5 or better
- FreeBSD (3.2, 4.0 tested)

To enable it, configure PHP3 and PHP4 to use APXS (--with-apxs) and the necessary link extensions (--enable-versioning). Otherwise, all standard installations instructions apply. For example:

```
$ ./configure \
  --with-apxs=/apache/bin/apxs \
  --enable-versioning \
  --with-mysql \
  --enable-track-vars
```

Migrating Configuration Files

The global configuration file, `php3.ini`, has changed its name to `php.ini`.

For the Apache configuration file, there are slightly more changes. The MIME types recognized by the PHP module have changed.

```
application/x-httpd-php3        -->    application/x-httpd-php
application/x-httpd-php3-source -->    application/x-httpd-php-source
```

You can make your configuration files work with both versions of PHP (depending on which one is currently compiled into the server), using the following syntax:

```
AddType  application/x-httpd-php3        .php3
AddType  application/x-httpd-php3-source .php3s

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

In addition, the PHP directive names for Apache have changed.

Starting with PHP 4.0, there are only four Apache directives that relate to PHP:

```
php_value [PHP directive name] [value]
php_flag [PHP directive name] [On|Off]
php_admin_value [PHP directive name] [value]
php_admin_flag [PHP directive name] [On|Off]
```

There are two differences between the Admin values and the non admin values:

- Admin values (or flags) can only appear in the server-wide apache configuration files (e.g., httpd.conf).
- Standard values (or flags) cannot control certain PHP directives, for example - safe mode (if you could override safe mode settings in .htaccess files, it would defeat safe-mode's purpose). In contrast, Admin values can modify the value of any PHP directive.

To make the transition process easier, PHP 4 is bundled with scripts that automatically convert your Apache configuration and .htaccess files to work with both PHP 3 and PHP 4. These scripts do NOT convert the mime type lines! You have to convert these yourself.

To convert your Apache configuration files, run the apconf-conv.sh script (available in the scripts/apache/ directory). For example:

```
~/php4/scripts/apache:#  ./apconf-conv.sh /usr/local/apache/conf/httpd.conf
```

Your original configuration file will be saved in httpd.conf.orig.

To convert your .htaccess files, run the aphtaccess-conv.sh script (available in the scripts/apache/ directory as well):

```
~/php4/scripts/apache:#  find / -name .htaccess -exec ./aphtaccess-conv.sh {} \;
```

Likewise, your old .htaccess files will be saved with an .orig prefix.

The conversion scripts require awk to be installed.

Parser behavior

Parsing and execution are now two completely separated steps, no execution of a files code will happen until the complete file and everything it requires has completely and successfully been parsed.

One of the new requirements introduced with this split is that required and included files now have to be syntactically complete. You can no longer spread the different controlling parts of a control structure across file boundaries. That is you cannot start a `for` or `while` loop, an `if` statement or a `switch` block in one file and have the end of loop, `else`, `endif`, `case` or `break` statements in a different file.

It still perfectly legal to include additional code within loops or other control structures, only the controlling keywords and corresponding curly braces `{...}` have to be within the same compile unit (file or **eval()**ed string).

This should not harm to much as spreading code like this should be considered as very bad style anyway.

Another thing no longer possible, though rarely seen in PHP 3 code is returning values from a required file. Returning a value from an included file is still possible.

Error reporting

Configuration changes

With PHP 3 the error reporting level was set as a simple numeric value formed by summing up the numbers related to different error levels. Usual values where 15 for reporting all errors and warnings or 7 for reporting everything but simple notice messages reporting bad style and things like that.

PHP 4 has a larger set of error and warning levels and comes with a configuration parser that now allows for symbolic constants to be used for setting the intended behavior.

Error reporting level should now be configured by explicitly taking away the warning levels you do not want to generate error messages by x-oring them from the symbolic constant `E_ALL`. Sounds complicated? Well, lets say you want the error reporting system to report all but the simple style warnings that are categorized by the symbolic constant `E_NOTICE`. Then you'll put the following into your `php.ini`: `error_reporting = E_ALL & ~ ( E_NOTICE )`. If you want to suppress warnings too you add up the appropriate constant within the braces using the binary or operator '|': `error_reporting= E_ALL & ~ ( E_NOTICE | E_WARNING )`.

Warning
When upgrading code or servers from PHP 3 to PHP 4 you should check these settings and calls to **error_reporting()** or you might disable reporting the new error types, especially E_COMPILE_ERROR. This may lead to empty documents without any feedback of what happened or where to look for the problem.

Warning
Using the old values 7 and 15 for setting up error reporting is a very bad idea as this suppresses some of the newly added error classes including parse errors. This may lead to very strange behavior as scripts might no longer work without error messages showing up anywhere. This has lead to a lot of unreproducible bug reports in the past where people reported script engine problems they were not capable to track down while the **TRUE** case was usually some missing '}' in a required file that the parser was not able to report due to a misconfigured error reporting system. So checking your error reporting setup should be the first thing to do whenever your scripts silently die. The Zend engine can be considered mature enough nowadays to not cause this kind of strange behavior.

Additional warning messages

A lot of existing PHP 3 code uses language constructs that should be considered as very bad style as this code, while doing the intended thing now, could easily be broken by changes in other places. PHP 4 will output a lot of notice messages in such situations where PHP 3 didn't. The easy fix is to just turn off E_NOTICE messages, but it is usually a good idea to fix the code instead.

The most common case that will now produce notice messages is the use of unquoted string constants as array indices. Both PHP 3 and 4 will fall back to interpret these as strings if no keyword or constant is known by that name, but whenever a constant by that name had been defined anywhere else in the code it might break your script. This can even become a security risk if some intruder manages to redefine string constants in a way that makes your script give him access rights he wasn't intended to have. So PHP 4 will now warn you whenever you use unquoted string constants as for example in `$_SERVER[REQUEST_METHOD]`. Changing it to `$_SERVER['REQUEST_METHOD']` will make the parser happy and greatly improve the style and security of your code.

Another thing PHP 4 will now tell you about is the use of uninitialized variables or array elements.

Initializers

Static variable and class member initializers only accept scalar values while in PHP 3 they accepted any valid expression. This is, once again, due to the split between parsing and execution as no code has yet been executed when the parser sees the initializer.

For classes you should use constructors to initialize member variables instead. For static variables anything but a simple static value rarely makes sense anyway.

`empty("0")`

The perhaps most controversial change in behavior has happened to the behavior of the **empty()**. A String containing only the character '0' (zero) is now considered empty while it wasn't in PHP 3.

This new behavior makes sense in web applications, with all input fields returning strings even if numeric input is requested, and with PHP's capabilities of automatic type conversion. But on the other hand it might break your code in a rather subtle way, leading to misbehavior that is hard to track down if you do not know about what to look for.

Missing functions

While PHP 4 comes with a lot of new features, functions and extensions, you may still find some functions from version 3 missing. A small number of core functions has vanished because they do not work with the new scheme of splitting parsing and execution as introduced into 4 with the Zend engine. Other functions and even complete extensions have become obsolete as newer functions and extensions serve the same task better and/or in a more general way. Some functions just simply haven't been ported yet and finally some functions or extensions may be missing due to license conflicts.

Functions missing due to conceptual changes

As PHP 4 now separates parsing from execution it is no longer possible to change the behavior of the parser (now embedded in the Zend engine) at runtime as parsing already happened by then. So the function **short_tags()** no longer exists. You can still change the parsers behavior by setting appropriate values in the `php.ini` file.

Another feature of PHP 3 that is not a part of PHP 4 is the bundled debugging interface. There are third-party add-ons for the Zend engine which add similar functionality.

Deprecate functions and extensions

The Adabas and Solid database extensions are no more. Long live the unified ODBC extension instead.

Changed status for unset()

unset(), although still available, is implemented as a language construct rather than a function.

This does not have any consequences on the behavior of unset(), but testing for "unset" using function_exists() will return FALSE as it would with other language constructs that look like functions such as echo().

Another more practical change is that it is no longer possible to call unset() indirectly, that is `$func="unset"; $func($somevar)` won't work anymore.

PHP 3 extension

Extensions written for PHP 3 will not work with PHP 4, neither as binaries nor at the source level. It is not difficult to port extensions to PHP 4 if you have access to the original source. A detailed description of the actual porting process is not part of this text.

Variable substitution in strings

PHP 4 adds a new mechanism to variable substitution in strings. You can now finally access object member variables and elements from multidimensional arrays within strings.

To do so you have to enclose your variables with curly braces with the dollar sign immediately following the opening brace: `{$...}`

To embed the value of an object member variable into a string you simply write `"text {$obj->member} text"` while in PHP 3 you had to use something like `"text ".$obj->member." text"`.

This should lead to more readable code, while it may break existing scripts written for PHP 3. But you can easily check for this kind of problem by checking for the character combination `{$` in your code and by replacing it with `\{$` with your favorite search-and-replace tool.

Cookies

PHP 3 had the bad habit of setting cookies in the reverse order of the setcookie() calls in your code. PHP 4 breaks with this habit and creates the cookie header lines in exactly the same order as you set the cookies in the code.

This might break some existing code, but the old behaviour was so strange to understand that it deserved a change to prevent further problems in the future.

Handling of global variables

While handling of global variables had the focus on to be easy in PHP 3 and early versions of PHP 4, the focus has changed to be more secure. While in PHP 3 the following example worked fine, in PHP 4 it has to be unset($GLOBALS["id"]);. This is only one issue of global variable handling. You should always have used $GLOBALS, with newer versions of PHP 4 you are forced to do so in most cases. Read more on this subject in the `global` references section.

Example B-1. Migration of global variables

```
<?php
$id = 1;
function test()
{
    global $id;
    unset($id);
}
test();
echo($id); // This will print out 1 in PHP 4
```

```
?>
```

Appendix C. Migrating from PHP/FI 2 to PHP 3

About the incompatibilities in 3.0

PHP 3.0 is rewritten from the ground up. It has a proper parser that is much more robust and consistent than 2.0's. 3.0 is also significantly faster, and uses less memory. However, some of these improvements have not been possible without compatibility changes, both in syntax and functionality.

In addition, PHP's developers have tried to clean up both PHP's syntax and semantics in version 3.0, and this has also caused some incompatibilities. In the long run, we believe that these changes are for the better.

This chapter will try to guide you through the incompatibilities you might run into when going from PHP/FI 2.0 to PHP 3.0 and help you resolve them. New features are not mentioned here unless necessary.

A conversion program that can automatically convert your old PHP/FI 2.0 scripts exists. It can be found in the `convertor` subdirectory of the PHP 3.0 distribution. This program only catches the syntax changes though, so you should read this chapter carefully anyway.

Start/end tags

The first thing you probably will notice is that PHP's start and end tags have changed. The old `<? >` form has been replaced by three new possible forms:

Example C-1. Migration: old start/end tags

```
<? echo "This is PHP/FI 2.0 code.\n"; ?>
```

As of version 2.0, PHP/FI also supports this variation:

Example C-2. Migration: first new start/end tags

```
<? echo "This is PHP 3.0 code!\n"; ?>
```

Notice that the end tag now consists of a question mark and a greater-than character instead of just greater-than. However, if you plan on using XML on your server, you will get problems with the first new variant, because PHP may try to execute the XML markup in XML documents as PHP code. Because of this, the following variation was introduced:

Example C-3. Migration: second new start/end tags

```
<?php echo "This is PHP 3.0 code!\n"; ?>
```

Some people have had problems with editors that don't understand the processing instruction tags at all. Microsoft FrontPage is one such editor, and as a workaround for these, the following variation was introduced as well:

Example C-4. Migration: third new start/end tags

```
<script language="php">
  echo "This is PHP 3.0 code!\n";
</script>
```

---

if..endif syntax

The `alternative' way to write if/elseif/else statements, using if(); elseif(); else; endif; cannot be efficiently implemented without adding a large amount of complexity to the 3.0 parser. Because of this, the syntax has been changed:

Example C-5. Migration: old if..endif syntax

```
if ($foo);
    echo "yep\n";
elseif ($bar);
```

```
    echo "almost\n";
else;
    echo "nope\n";
endif;
```

Example C-6. Migration: new if..endif syntax

```
if ($foo):
    echo "yep\n";
elseif ($bar):
    echo "almost\n";
else:
    echo "nope\n";
endif;
```

Notice that the semicolons have been replaced by colons in all statements but the one terminating the expression (endif).

---

while syntax

Just like with if..endif, the syntax of while..endwhile has changed as well:

Example C-7. Migration: old while..endwhile syntax

```
while ($more_to_come);
    ...
endwhile;
```

Example C-8. Migration: new while..endwhile syntax

```
while ($more_to_come):
    ...
endwhile;
```

Warning
If you use the old while..endwhile syntax in PHP 3.0, you will get a never-ending loop.

Expression types

PHP/FI 2.0 used the left side of expressions to determine what type the result should be. PHP 3.0 takes both sides into account when determining result types, and this may cause 2.0 scripts to behave unexpectedly in 3.0.

Consider this example:

```
$a[0]=5;
$a[1]=7;

$key = key($a);
while ("" != $key) {
    echo "$keyn";
    next($a);
}
```

In PHP/FI 2.0, this would display both of $a's indices. In PHP 3.0, it wouldn't display anything. The reason is that in PHP 2.0, because the left argument's type was string, a string comparison was made, and indeed `""` does not equal `"0"`, and the loop went through. In PHP 3.0, when a string is compared with an integer, an integer comparison is made (the string is converted to an integer). This results in comparing `atoi("")` which is `0`, and `variablelist` which is also `0`, and since `0==0`, the loop doesn't go through even once.

The fix for this is simple. Replace the while statement with:

```
while ((string)$key != "") {
```

Error messages have changed

PHP 3.0's error messages are usually more accurate than 2.0's were, but you no longer get to see the code fragment causing the error. You will be supplied with a file name and a line number for the error, though.

Short-circuited boolean evaluation

In PHP 3.0 boolean evaluation is short-circuited. This means that in an expression like `(1 || test_me())`, the function **test_me()** would not be executed since nothing can change the result of the expression after the `1`.

This is a minor compatibility issue, but may cause unexpected side-effects.

Function TRUE/FALSE return values

Most internal functions have been rewritten so they return **TRUE** when successful and **FALSE** when failing, as opposed to 0 and -1 in PHP/FI 2.0, respectively. The new behaviour allows for more logical code, like `$fp = fopen("/your/file") or fail("darn!");`. Because PHP/FI 2.0 had no clear rules for what functions should return when they failed, most such scripts will probably have to be checked manually after using the 2.0 to 3.0 convertor.

Example C-9. Migration from 2.0: return values, old code

```
$fp = fopen($file, "r");
if ($fp == -1);
    echo("Could not open $file for reading<br>\n");
endif;
```

Example C-10. Migration from 2.0: return values, new code

```
$fp = @fopen($file, "r") or print("Could not open $file for reading<br>\n");
```

Other incompatibilities

- The PHP 3.0 Apache module no longer supports Apache versions prior to 1.2. Apache 1.2 or later is required.
- **echo()** no longer supports a format string. Use the **printf()** function instead.
- In PHP/FI 2.0, an implementation side-effect caused `$foo[0]` to have the same effect as `$foo`. This is not true for PHP 3.0.
- Reading arrays with `$array[]` is no longer supported

  That is, you cannot traverse an array by having a loop that does `$data = $array[]`. Use **current()** and **next()** instead.

  Also, `$array1[] = $array2` does not append the values of `$array2` to `$array1`, but appends `$array2` as the last entry of `$array1`. See also multidimensional array support.
- `"+"` is no longer overloaded as a concatenation operator for strings, instead it converts it's arguments to numbers and performs numeric addition. Use `"."` instead.

Example C-11. Migration from 2.0: concatenation for strings

```
echo "1" + "1";
```

In PHP 2.0 this would echo 11, in PHP 3.0 it would echo 2. Instead use:

```
echo "1"."1";

$a = 1;
$b = 1;
echo $a + $b;
```

This would echo 2 in both PHP 2.0 and 3.0.

```
$a = 1;
$b = 1;
echo $a.$b;
```

This will echo 11 in PHP 3.0.

Appendix D. Debugging PHP

About the debugger

PHP 3 includes support for a network-based debugger.

PHP 4 does not have an internal debugging facility. You can use one of the external debuggers though. The Zend IDE includes a debugger, and there are also some free debugger extensions like DBG at http://dd.cron.ru/dbg/, the Advanced PHP Debugger (APD) or Xdebug which even has a compatible debugger interface as PHP 3's debugging functionality as is described in this section.

Using the Debugger

The internal debugger in PHP 3 is useful for tracking down evasive bugs. The debugger works by connecting to a TCP port for every time PHP 3 starts up. All error messages from that request will be sent to this TCP connection. This information is intended for "debugging server" that can run inside an IDE or programmable editor (such as Emacs).

How to set up the debugger:

1. Set up a TCP port for the debugger in the configuration file (debugger.port) and enable it (debugger.enabled).
2. Set up a TCP listener on that port somewhere (for example **socket -l -s 1400** on UNIX).
3. In your code, run "debugger_on(*host*)", where *host* is the IP number or name of the host running the TCP listener.

Now, all warnings, notices etc. will show up on that listener socket, *even if you turned them off with **error_reporting()***.

Debugger Protocol

The PHP 3 debugger protocol is line-based. Each line has a *type*, and several lines compose a *message*. Each message starts with a line of the type `start` and terminates with a line of the type `end`. PHP 3 may send lines for different messages simultaneously.

A line has this format:

```
date time
host(pid)
type:
message-data
```

date

Date in ISO 8601 format (*yyyy-mm-dd*)

time

Time including microseconds: *hh:mm:uuuuuu*

host

DNS name or IP address of the host where the script error was generated.

pid

PID (process id) on *host* of the process with the PHP 3 script that generated this error.

type

Type of line. Tells the receiving program about what it should treat the following data as:

Table D-1. Debugger Line Types

Name	Meaning
`start`	Tells the receiving program that a debugger message starts here. The contents of *data* will be the type of error message, listed below.

Name	Meaning
`message`	The PHP 3 error message.
`location`	File name and line number where the error occurred. The first `location` line will always contain the top-level location. *`data`* will contain *`file`*:*`line`*. There will always be a `location` line after `message` and after every `function`.
`frames`	Number of frames in the following stack dump. If there are four frames, expect information about four levels of called functions. If no "frames" line is given, the depth should be assumed to be 0 (the error occurred at top-level).
`function`	Name of function where the error occurred. Will be repeated once for every level in the function call stack.
`end`	Tells the receiving program that a debugger message ends here.

`data`

Line data.

Table D-2. Debugger Error Types

Debugger	PHP 3 Internal
warning	E_WARNING
error	E_ERROR
parse	E_PARSE
notice	E_NOTICE
core-error	E_CORE_ERROR
core-warning	E_CORE_WARNING
unknown	(any other)

Example D-1. Example Debugger Message

```
1998-04-05 23:27:400966 lucifer.guardian.no(20481) start: notice
1998-04-05 23:27:400966 lucifer.guardian.no(20481) message: Uninitialized variable
1998-04-05 23:27:400966 lucifer.guardian.no(20481) location: (NULL):7
1998-04-05 23:27:400966 lucifer.guardian.no(20481) frames: 1
1998-04-05 23:27:400966 lucifer.guardian.no(20481) function: display
1998-04-05 23:27:400966 lucifer.guardian.no(20481) location: /home/ssb/public_html/test.php3:10
1998-04-05 23:27:400966 lucifer.guardian.no(20481) end: notice
```

Appendix E. Extending PHP

Adding functions to PHP

Function Prototype

All functions look like this:

```
void php3_foo(INTERNAL_FUNCTION_PARAMETERS) {

}
```

Even if your function doesn't take any arguments, this is how it is called.

Function Arguments

Arguments are always of type pval. This type contains a union which has the actual type of the argument. So, if your function takes two arguments, you would do something like the following at the top of your function:

Example E-1. Fetching function arguments

```
pval *arg1, *arg2;
if (ARG_COUNT(ht) != 2 || getParameters(ht,2,&arg1,&arg2)==FAILURE) {
   WRONG_PARAM_COUNT;
}
```

NOTE: Arguments can be passed either by value or by reference. In both cases you will need to pass &(pval *) to getParameters. If you want to check if the n'th parameter was sent to you by reference or not, you can use the function, ParameterPassedByReference(ht,n). It will return either 1 or 0.

When you change any of the passed parameters, whether they are sent by reference or by value, you can either start over with the parameter by calling pval_destructor on it, or if it's an ARRAY you want to add to, you can use functions similar to the ones in internal_functions.h which manipulate return_value as an ARRAY.

Also if you change a parameter to IS_STRING make sure you first assign the new estrdup()'ed string and the string length, and only later change the type to IS_STRING. If you change the string of a parameter which already IS_STRING or IS_ARRAY you should run pval_destructor on it first.

Variable Function Arguments

A function can take a variable number of arguments. If your function can take either 2 or 3 arguments, use the following:

Example E-2. Variable function arguments

```
pval *arg1, *arg2, *arg3;
int arg_count = ARG_COUNT(ht);

if (arg_count < 2 || arg_count > 3 ||
    getParameters(ht,arg_count,&arg1,&arg2,&arg3)==FAILURE) {
    WRONG_PARAM_COUNT;
}
```

Using the Function Arguments

The type of each argument is stored in the pval type field. This type can be any of the following:

Table E-1. PHP Internal Types

IS_STRING	String
IS_DOUBLE	Double-precision floating point
IS_LONG	Long integer
IS_ARRAY	Array
IS_EMPTY	None
IS_USER_FUNCTION	??
IS_INTERNAL_FUNCTION	?? (if some of these cannot be passed to a function - delete)
IS_CLASS	??
IS_OBJECT	??

If you get an argument of one type and would like to use it as another, or if you just want to force the argument to be of a certain type, you can use one of the following conversion functions:

```
convert_to_long(arg1);
convert_to_double(arg1);
convert_to_string(arg1);
convert_to_boolean_long(arg1); /* If the string is "" or "0" it becomes 0, 1 otherwise */
convert_string_to_number(arg1);  /* Converts string to either LONG or DOUBLE depending on string */
```

These function all do in-place conversion. They do not return anything.

The actual argument is stored in a union; the members are:

- IS_STRING: arg1->value.str.val
- IS_LONG: arg1->value.lval
- IS_DOUBLE: arg1->value.dval

Memory Management in Functions

Any memory needed by a function should be allocated with either emalloc() or estrdup(). These are memory handling abstraction functions that look and smell like the normal malloc() and strdup() functions. Memory should be freed with efree().

There are two kinds of memory in this program: memory which is returned to the parser in a variable, and memory which you need for temporary storage in your internal function. When you assign a string to a variable which is returned to the parser you need to make sure you first allocate the memory with either emalloc() or estrdup(). This memory should NEVER be freed by you, unless you later in the same function overwrite your original assignment (this kind of programming practice is not good though).

For any temporary/permanent memory you need in your functions/library you should use the three emalloc(), estrdup(), and efree() functions. They behave EXACTLY like their counterpart functions. Anything you emalloc() or estrdup() you have to efree() at some point or another, unless it's supposed to stick around until the end of the program; otherwise, there will be a memory leak. The meaning of "the functions behave exactly like their counterparts" is: if you efree() something which was not emalloc()'ed nor estrdup()'ed you might get a segmentation fault. So please take care and free all of your wasted memory.

If you compile with "-DDEBUG", PHP will print out a list of all memory that was allocated using emalloc() and estrdup() but never freed with efree() when it is done running the specified script.

Setting Variables in the Symbol Table

A number of macros are available which make it easier to set a variable in the symbol table:

- SET_VAR_STRING(name,value)
- SET_VAR_DOUBLE(name,value)
- SET_VAR_LONG(name,value)

Warning
Be careful with SET_VAR_STRING. The value part must be malloc'ed manually because the memory management code will try to free this pointer later. Do not pass statically allocated memory into a SET_VAR_STRING.

Symbol tables in PHP are implemented as hash tables. At any given time, &symbol_table is a pointer to the 'main' symbol table, and active_symbol_table points to the currently active symbol table (these may be identical like in startup, or different, if you're inside a function).

The following examples use 'active_symbol_table'. You should replace it with &symbol_table if you specifically want to work with the 'main' symbol table. Also, the same functions may be applied to arrays, as explained below.

Example E-3. Checking whether $foo exists in a symbol table

```
if (hash_exists(active_symbol_table,"foo",sizeof("foo"))) { exists... }
else { doesn't exist }
```

Example E-4. Finding a variable's size in a symbol table

```
hash_find(active_symbol_table,"foo",sizeof("foo"),&pvalue);
check(pvalue.type);
```

Arrays in PHP are implemented using the same hashtables as symbol tables. This means the two above functions can also be used to check variables inside arrays.

If you want to define a new array in a symbol table, you should do the following.

First, you may want to check whether it exists and abort appropriately, using hash_exists() or hash_find().

Next, initialize the array:

Example E-5. Initializing a new array

```
pval arr;

if (array_init(&arr) == FAILURE) { failed... };
hash_update(active_symbol_table,"foo",sizeof("foo"),&arr,sizeof(pval),NULL);
```

This code declares a new array, named $foo, in the active symbol table. This array is empty.

Here's how to add new entries to it:

Example E-6. Adding entries to a new array

```
pval entry;

entry.type = IS_LONG;
entry.value.lval = 5;

/* defines $foo["bar"] = 5 */
hash_update(arr.value.ht,"bar",sizeof("bar"),&entry,sizeof(pval),NULL);

/* defines $foo[7] = 5 */
hash_index_update(arr.value.ht,7,&entry,sizeof(pval),NULL);

/* defines the next free place in $foo[],
 * $foo[8], to be 5 (works like php2)
 */
hash_next_index_insert(arr.value.ht,&entry,sizeof(pval),NULL);
```

If you'd like to modify a value that you inserted to a hash, you must first retrieve it from the hash. To prevent that overhead, you can supply a pval ** to the hash add function, and it'll be updated with the pval * address of the inserted element inside the hash. If that value is **NULL** (like in all of the above examples) - that parameter is ignored.

hash_next_index_insert() uses more or less the same logic as "$foo[] = bar;" in PHP 2.0.

If you are building an array to return from a function, you can initialize the array just like above by doing:

```
if (array_init(return_value) == FAILURE) { failed...; }
```

...and then adding values with the helper functions:

```
add_next_index_long(return_value,long_value);
add_next_index_double(return_value,double_value);
add_next_index_string(return_value,estrdup(string_value));
```

Of course, if the adding isn't done right after the array initialization, you'd probably have to look for the array first:

```
pval *arr;

if (hash_find(active_symbol_table,"foo",sizeof("foo"),(void **)&arr)==FAILURE) { can't find... }
else { use arr->value.ht... }
```

Note that hash_find receives a pointer to a pval pointer, and not a pval pointer.

Just about any hash function returns SUCCESS or FAILURE (except for hash_exists(), which returns a boolean truth value).

Returning simple values

A number of macros are available to make returning values from a function easier.

The RETURN_* macros all set the return value and return from the function:

- RETURN
- RETURN_FALSE
- RETURN_TRUE
- RETURN_LONG(l)
- RETURN_STRING(s,dup) If dup is **TRUE**, duplicates the string
- RETURN_STRINGL(s,l,dup) Return string (s) specifying length (l).
- RETURN_DOUBLE(d)

The RETVAL_* macros set the return value, but do not return.

- RETVAL_FALSE
- RETVAL_TRUE
- RETVAL_LONG(l)
- RETVAL_STRING(s,dup) If dup is **TRUE**, duplicates the string
- RETVAL_STRINGL(s,l,dup) Return string (s) specifying length (l).

- RETVAL_DOUBLE(d)

The string macros above will all estrdup() the passed 's' argument, so you can safely free the argument after calling the macro, or alternatively use statically allocated memory.

If your function returns boolean success/error responses, always use RETURN_TRUE and RETURN_FALSE respectively.

Returning complex values

Your function can also return a complex data type such as an object or an array.

Returning an object:

1. Call object_init(return_value).
2. Fill it up with values. The functions available for this purpose are listed below.
3. Possibly, register functions for this object. In order to obtain values from the object, the function would have to fetch "this" from the active_symbol_table. Its type should be IS_OBJECT, and it's basically a regular hash table (i.e., you can use regular hash functions on .value.ht). The actual registration of the function can be done using:
```
add_method( return_value, function_name, function_ptr );
```

The functions used to populate an object are:

- add_property_long(return_value, property_name, l) - Add a property named 'property_name', of type long, equal to 'l'
- add_property_double(return_value, property_name, d) - Same, only adds a double
- add_property_string(return_value, property_name, str) - Same, only adds a string
- add_property_stringl(return_value, property_name, str, l) - Same, only adds a string of length 'l'

Returning an array:

1. Call array_init(return_value).
2. Fill it up with values. The functions available for this purpose are listed below.

The functions used to populate an array are:

- add_assoc_long(return_value,key,l) - add associative entry with key 'key' and long value 'l'
- add_assoc_double(return_value,key,d)
- add_assoc_string(return_value,key,str,duplicate)
- add_assoc_stringl(return_value,key,str,length,duplicate) specify the string length
- add_index_long(return_value,index,l) - add entry in index 'index' with long value 'l'
- add_index_double(return_value,index,d)
- add_index_string(return_value,index,str)
- add_index_stringl(return_value,index,str,length) - specify the string length
- add_next_index_long(return_value,l) - add an array entry in the next free offset with long value 'l'
- add_next_index_double(return_value,d)
- add_next_index_string(return_value,str)
- add_next_index_stringl(return_value,str,length) - specify the string length

---

Using the resource list

PHP has a standard way of dealing with various types of resources. This replaces all of the local linked lists in PHP 2.0.

Available functions:

- php3_list_insert(ptr, type) - returns the 'id' of the newly inserted resource
- php3_list_delete(id) - delete the resource with the specified id
- php3_list_find(id,*type) - returns the pointer of the resource with the specified id, updates 'type' to the resource's type

Typically, these functions are used for SQL drivers but they can be used for anything else; for instance, maintaining file descriptors.

Typical list code would look like this:

Example E-7. Adding a new resource

```
RESOURCE *resource;

/* ...allocate memory for resource and acquire resource... */
/* add a new resource to the list */
return_value->value.lval = php3_list_insert((void *) resource, LE_RESOURCE_TYPE);
return_value->type = IS_LONG;
```

Example E-8. Using an existing resource

```
pval *resource_id;
RESOURCE *resource;
int type;

convert_to_long(resource_id);
resource = php3_list_find(resource_id->value.lval, &type);
if (type != LE_RESOURCE_TYPE) {
        php3_error(E_WARNING,"resource index %d has the wrong type",resource_id->value.lval);
        RETURN_FALSE;
}
/* ...use resource... */
```

Example E-9. Deleting an existing resource

```
pval *resource_id;
RESOURCE *resource;
int type;

convert_to_long(resource_id);
php3_list_delete(resource_id->value.lval);
```

The resource types should be registered in php3_list.h, in enum list_entry_type. In addition, one should add shutdown code for any new resource type defined, in list.c's list_entry_destructor() (even if you don't have anything to do on shutdown, you must add an empty case).

Using the persistent resource table

PHP has a standard way of storing persistent resources (i.e., resources that are kept in between hits). The first module to use this feature was the MySQL module, and mSQL followed it, so one can get the general impression of how a persistent resource should be used by reading mysql.c. The functions you should look at are:

php3_mysql_do_connect
php3_mysql_connect()
php3_mysql_pconnect()

The general idea of persistence modules is this:

1. Code all of your module to work with the regular resource list mentioned in section (9).
2. Code extra connect functions that check if the resource already exists in the persistent resource list. If it does, register it as in the regular resource list as a pointer to the persistent resource list (because of 1., the rest of the code should work immediately). If it doesn't, then create it, add it to the persistent resource list AND add a pointer to it from the regular resource list, so all of the code would work since it's in the regular resource list, but on the next connect, the resource would be found in the persistent resource list and be used without having to recreate it. You should register these resources with a different type (e.g. LE_MYSQL_LINK for non-persistent link and LE_MYSQL_PLINK for a persistent link).

If you read mysql.c, you'll notice that except for the more complex connect function, nothing in the rest of the module has to be changed.

The very same interface exists for the regular resource list and the persistent resource list, only 'list' is replaced with 'plist':

- php3_plist_insert(ptr, type) - returns the 'id' of the newly inserted resource

- php3_plist_delete(id) - delete the resource with the specified id
- php3_plist_find(id,*type) - returns the pointer of the resource with the specified id, updates 'type' to the resource's type

However, it's more than likely that these functions would prove to be useless for you when trying to implement a persistent module. Typically, one would want to use the fact that the persistent resource list is really a hash table. For instance, in the MySQL/mSQL modules, when there's a pconnect() call (persistent connect), the function builds a string out of the host/user/passwd that were passed to the function, and hashes the SQL link with this string as a key. The next time someone calls a pconnect() with the same host/user/passwd, the same key would be generated, and the function would find the SQL link in the persistent list.

Until further documented, you should look at mysql.c or msql.c to see how one should use the plist's hash table abilities.

One important thing to note: resources going into the persistent resource list must *NOT* be allocated with PHP's memory manager, i.e., they should NOT be created with emalloc(), estrdup(), etc. Rather, one should use the regular malloc(), strdup(), etc. The reason for this is simple - at the end of the request (end of the hit), every memory chunk that was allocated using PHP's memory manager is deleted. Since the persistent list isn't supposed to be erased at the end of a request, one mustn't use PHP's memory manager for allocating resources that go to it.

When you register a resource that's going to be in the persistent list, you should add destructors to it both in the non-persistent list and in the persistent list. The destructor in the non-persistent list destructor shouldn't do anything. The one in the persistent list destructor should properly free any resources obtained by that type (e.g. memory, SQL links, etc). Just like with the non-persistent resources, you *MUST* add destructors for every resource, even it requires no destruction and the destructor would be empty. Remember, since emalloc() and friends aren't to be used in conjunction with the persistent list, you mustn't use efree() here either.

Adding runtime configuration directives

Many of the features of PHP can be configured at runtime. These configuration directives can appear in either the designated php3.ini file, or in the case of the Apache module version in the Apache .conf files. The advantage of having them in the Apache .conf files is that they can be configured on a per-directory basis. This means that one directory may have a certain safemodeexecdir for example, while another directory may have another. This configuration granularity is especially handy when a server supports multiple virtual hosts.

The steps required to add a new directive:

1. Add directive to php3_ini_structure struct in mod_php3.h.
2. In main.c, edit the php3_module_startup function and add the appropriate cfg_get_string() or cfg_get_long() call.
3. Add the directive, restrictions and a comment to the php3_commands structure in mod_php3.c. Note the restrictions part. RSRC_CONF are directives that can only be present in the actual Apache .conf files. Any OR_OPTIONS directives can be present anywhere, include normal .htaccess files.
4. In either php3take1handler() or php3flaghandler() add the appropriate entry for your directive.
5. In the configuration section of the _php3_info() function in functions/info.c you need to add your new directive.
6. And last, you of course have to use your new directive somewhere. It will be addressable as php3_ini.directive.

Calling User Functions

To call user functions from an internal function, you should use the **call_user_function()** function.

call_user_function() returns SUCCESS on success, and FAILURE in case the function cannot be found. You should check that return value! If it returns SUCCESS, you are responsible for destroying the retval pval yourself (or return it as the return value of your function). If it returns FAILURE, the value of retval is undefined, and you mustn't touch it.

All internal functions that call user functions *must* be reentrant. Among other things, this means they must not use globals or static variables.

call_user_function() takes six arguments:

HashTable *function_table

This is the hash table in which the function is to be looked up.

---

pval *object

This is a pointer to an object on which the function is invoked. This should be `NULL` if a global function is called. If it's not `NULL` (i.e. it points to an object), the function_table argument is ignored, and instead taken from the object's hash. The object *may* be modified by the function that is invoked on it (that function will have access to it via $this). If for some reason you don't want that to happen, send a copy of the object instead.

---

pval *function_name

The name of the function to call. Must be a pval of type IS_STRING with function_name.str.val and function_name.str.len set to the appropriate values. The function_name is modified by call_user_function() - it's converted to lowercase. If you need to preserve the case, send a copy of the function name instead.

---

pval *retval

A pointer to a pval structure, into which the return value of the invoked function is saved. The structure must be previously allocated - **call_user_function()** does NOT allocate it by itself.

---

int param_count

The number of parameters being passed to the function.

---

pval *params[]

An array of pointers to values that will be passed as arguments to the function, the first argument being in offset 0, the second in offset 1, etc. The array is an array of pointers to pval's; The pointers are sent as-is to the function, which means if the function modifies its arguments, the original values are changed (passing by reference). If you don't want that behavior, pass a copy instead.

Reporting Errors

To report errors from an internal function, you should call the **php3_error()** function. This takes at least two parameters -- the first is the level of the error, the second is the format string for the error message (as in a standard **printf()** call), and any following arguments are the parameters for the format string. The error levels are:

E_NOTICE

Notices are not printed by default, and indicate that the script encountered something that could indicate an error, but could also happen in the normal course of running a script. For example, trying to access the value of a variable which has not been set, or calling **stat()** on a file that doesn't exist.

E_WARNING

Warnings are printed by default, but do not interrupt script execution. These indicate a problem that should have been trapped by the script before the call was made. For example, calling **ereg()** with an invalid regular expression.

E_ERROR

Errors are also printed by default, and execution of the script is halted after the function returns. These indicate errors that can not be recovered from, such as a memory allocation problem.

E_PARSE

Parse errors should only be generated by the parser. The code is listed here only for the sake of completeness.

E_CORE_ERROR

This is like an E_ERROR, except it is generated by the core of PHP. Functions should not generate this type of error.

E_CORE_WARNING

This is like an E_WARNING, except it is generated by the core of PHP. Functions should not generate this type of error.

E_COMPILE_ERROR

This is like an E_ERROR, except it is generated by the Zend Scripting Engine. Functions should not generate this type of error.

E_COMPILE_WARNING

This is like an E_WARNING, except it is generated by the Zend Scripting Engine. Functions should not generate this type of error.

E_USER_ERROR

This is like an E_ERROR, except it is generated in PHP code by using the PHP function **trigger_error()**. Functions should not generate this type of error.

E_USER_WARNING

This is like an E_WARNING, except it is generated by using the PHP function **trigger_error()**. Functions should not generate this type of error.

E_USER_NOTICE

This is like an E_NOTICE, except it is generated by using the PHP function **trigger_error()**. Functions should not generate this type of error.

E_ALL

All of the above. Using this error_reporting level will show all error types.

Appendix F. List of Function Aliases

Here is the aliases list. All aliases are listed here. It is usually a bad idea to use aliases, as they may be bound to obsolescence or renaming, which will lead to unportable script. This list is provided to help those who want to upgrade their old scripts to newer syntax.

However, some functions simply have two names, and there is no real preference. (For example, **is_int()** and **is_integer()** are equally good)

This list is consistent with PHP 4.0.6. For an alias list that updates daily, have a look here.

Table F-1. Aliases

Alias	Master function	Extension used
_	**gettext()**	Gettext
add	**swfmovie_add()**	Ming (flash)
add	**swfsprite_add()**	Ming (flash)
add_root	**domxml_add_root()**	DOM XML
addaction	**swfbutton_addAction()**	Ming (flash)
addcolor	**swfdisplayitem_addColor()**	Ming (flash)
addentry	**swfgradient_addEntry()**	Ming (flash)
addfill	**swfshape_addfill()**	Ming (flash)
addshape	**swfbutton_addShape()**	Ming (flash)
addstring	**swftext_addString()**	Ming (flash)
addstring	**swftextfield_addString()**	Ming (flash)
align	**swftextfield_align()**	Ming (flash)
attributes	**domxml_attributes()**	DOM XML
children	**domxml_children()**	DOM XML
chop	**rtrim()**	Base syntax
close	**closedir()**	Base syntax
com_get	**com_propget()**	COM
com_propset	**com_propput()**	COM
com_set	**com_propput()**	COM
cv_add	**ccvs_add()**	CCVS
cv_auth	**ccvs_auth()**	CCVS
cv_command	**ccvs_command()**	CCVS
cv_count	**ccvs_count()**	CCVS
cv_delete	**ccvs_delete()**	CCVS
cv_done	**ccvs_done()**	CCVS
cv_init	**ccvs_init()**	CCVS
cv_lookup	**ccvs_lookup()**	CCVS
cv_new	**ccvs_new()**	CCVS
cv_report	**ccvs_report()**	CCVS
cv_return	**ccvs_return()**	CCVS
cv_reverse	**ccvs_reverse()**	CCVS
cv_sale	**ccvs_sale()**	CCVS
cv_status	**ccvs_status()**	CCVS
cv_textvalue	**ccvs_textvalue()**	CCVS
cv_void	**ccvs_void()**	CCVS
die	**exit()**	Miscellaneous functions
dir	**getdir()**	Base syntax
diskfreespace	**disk_free_space()**	Filesystem
domxml_getattr	**domxml_get_attribute()**	DOM XML
domxml_setattr	**domxml_set_attribute()**	DOM XML
doubleval	**floatval()**	Base syntax
drawarc	**swfshape_drawarc()**	Ming (flash)
drawcircle	**swfshape_drawcircle()**	Ming (flash)
drawcubic	**swfshape_drawcubic()**	Ming (flash)
drawcubicto	**swfshape_drawcubicto()**	Ming (flash)
drawcurve	**swfshape_drawcurve()**	Ming (flash)
drawcurveto	**swfshape_drawcurveto()**	Ming (flash)
drawglyph	**swfshape_drawglyph()**	Ming (flash)

Alias	Master function	Extension used
drawline	**swfshape_drawline()**	Ming (flash)
drawlineto	**swfshape_drawlineto()**	Ming (flash)
dtd	**domxml_intdtd()**	DOM XML
dumpmem	**domxml_dumpmem()**	DOM XML
fbsql	**fbsql_db_query()**	FrontBase
fputs	**fwrite()**	Base syntax
get_attribute	**domxml_get_attribute()**	DOM XML
getascent	**swffont_getAscent()**	Ming (flash)
getascent	**swftext_getAscent()**	Ming (flash)
getattr	**domxml_get_attribute()**	DOM XML
getdescent	**swffont_getDescent()**	Ming (flash)
getdescent	**swftext_getDescent()**	Ming (flash)
getheight	**swfbitmap_getHeight()**	Ming (flash)
getleading	**swffont_getLeading()**	Ming (flash)
getleading	**swftext_getLeading()**	Ming (flash)
getshape1	**swfmorph_getShape1()**	Ming (flash)
getshape2	**swfmorph_getShape2()**	Ming (flash)
getwidth	**swfbitmap_getWidth()**	Ming (flash)
getwidth	**swffont_getWidth()**	Ming (flash)
getwidth	**swftext_getWidth()**	Ming (flash)
gzputs	**gzwrite()**	Zlib
i18n_convert	**mb_convert_encoding()**	Multi-bytes Strings
i18n_discover_encoding	**mb_detect_encoding()**	Multi-bytes Strings
i18n_http_input	**mb_http_input()**	Multi-bytes Strings
i18n_http_output	**mb_http_output()**	Multi-bytes Strings
i18n_internal_encoding	**mb_internal_encoding()**	Multi-bytes Strings
i18n_ja_jp_hantozen	**mb_convert_kana()**	Multi-bytes Strings
i18n_mime_header_decode	**mb_decode_mimeheader()**	Multi-bytes Strings
i18n_mime_header_encode	**mb_encode_mimeheader()**	Multi-bytes Strings
imap_create	**imap_createmailbox()**	IMAP
imap_fetchtext	**imap_body()**	IMAP
imap_getmailboxes	**imap_list_full()**	IMAP
imap_getsubscribed	**imap_lsub_full()**	IMAP
imap_header	**imap_headerinfo()**	IMAP
imap_listmailbox	**imap_list()**	IMAP
imap_listsubscribed	**imap_lsub()**	IMAP
imap_rename	**imap_renamemailbox()**	IMAP
imap_scan	**imap_listscan()**	IMAP
imap_scanmailbox	**imap_listscan()**	IMAP
ini_alter	**ini_set()**	Base syntax
is_double	**is_float()**	Base syntax
is_integer	**is_int()**	Base syntax
is_long	**is_int()**	Base syntax
is_real	**is_float()**	Base syntax
is_writeable	**is_writable()**	Base syntax
join	**implode()**	Base syntax
labelframe	**swfmovie_labelFrame()**	Ming (flash)
labelframe	**swfsprite_labelFrame()**	Ming (flash)
last_child	**domxml_last_child()**	DOM XML
lastchild	**domxml_last_child()**	DOM XML
ldap_close	**ldap_unbind()**	LDAP

Alias	Master function	Extension used
magic_quotes_runtime	**set_magic_quotes_runtime()**	Base syntax
mbstrcut	**mb_strcut()**	Multi-bytes Strings
mbstrlen	**mb_strlen()**	Multi-bytes Strings
mbstrpos	**mb_strpos()**	Multi-bytes Strings
mbstrrpos	**mb_strrpos()**	Multi-bytes Strings
mbsubstr	**mb_substr()**	Multi-bytes Strings
ming_setcubicthreshold	**ming_setCubicThreshold()**	Ming (flash)
ming_setscale	**ming_setScale()**	Ming (flash)
move	**swfdisplayitem_move()**	Ming (flash)
movepen	**swfshape_movepen()**	Ming (flash)
movepento	**swfshape_movepento()**	Ming (flash)
moveto	**swfdisplayitem_moveTo()**	Ming (flash)
moveto	**swffill_moveTo()**	Ming (flash)
moveto	**swftext_moveTo()**	Ming (flash)
msql	**msql_db_query()**	mSQL
msql_createdb	**msql_create_db()**	mSQL
msql_dbname	**msql_result()**	mSQL
msql_dropdb	**msql_drop_db()**	mSQL
msql_fieldflags	**msql_field_flags()**	mSQL
msql_fieldlen	**msql_field_len()**	mSQL
msql_fieldname	**msql_field_name()**	mSQL
msql_fieldtable	**msql_field_table()**	mSQL
msql_fieldtype	**msql_field_type()**	mSQL
msql_freeresult	**msql_free_result()**	mSQL
msql_listdbs	**msql_list_dbs()**	mSQL
msql_listfields	**msql_list_fields()**	mSQL
msql_listtables	**msql_list_tables()**	mSQL
msql_numfields	**msql_num_fields()**	mSQL
msql_numrows	**msql_num_rows()**	mSQL
msql_regcase	**sql_regcase()**	mSQL
msql_selectdb	**msql_select_db()**	mSQL
msql_tablename	**msql_result()**	mSQL
mssql_affected_rows	**sybase_affected_rows()**	Sybase
mssql_affected_rows	**sybase_affected_rows()**	Sybase
mssql_close	**sybase_close()**	Sybase
mssql_close	**sybase_close()**	Sybase
mssql_connect	**sybase_connect()**	Sybase
mssql_connect	**sybase_connect()**	Sybase
mssql_data_seek	**sybase_data_seek()**	Sybase
mssql_data_seek	**sybase_data_seek()**	Sybase
mssql_fetch_array	**sybase_fetch_array()**	Sybase
mssql_fetch_array	**sybase_fetch_array()**	Sybase
mssql_fetch_field	**sybase_fetch_field()**	Sybase
mssql_fetch_field	**sybase_fetch_field()**	Sybase
mssql_fetch_object	**sybase_fetch_object()**	Sybase
mssql_fetch_object	**sybase_fetch_object()**	Sybase
mssql_fetch_row	**sybase_fetch_row()**	Sybase
mssql_fetch_row	**sybase_fetch_row()**	Sybase
mssql_field_seek	**sybase_field_seek()**	Sybase
mssql_field_seek	**sybase_field_seek()**	Sybase
mssql_free_result	**sybase_free_result()**	Sybase

Alias	Master function	Extension used
mssql_free_result	**sybase_free_result()**	Sybase
mssql_get_last_message	**sybase_get_last_message()**	Sybase
mssql_get_last_message	**sybase_get_last_message()**	Sybase
mssql_min_client_severity	**sybase_min_client_severity()**	Sybase
mssql_min_error_severity	**sybase_min_error_severity()**	Sybase
mssql_min_message_severity	**sybase_min_message_severity()**	Sybase
mssql_min_server_severity	**sybase_min_server_severity()**	Sybase
mssql_num_fields	**sybase_num_fields()**	Sybase
mssql_num_fields	**sybase_num_fields()**	Sybase
mssql_num_rows	**sybase_num_rows()**	Sybase
mssql_num_rows	**sybase_num_rows()**	Sybase
mssql_pconnect	**sybase_pconnect()**	Sybase
mssql_pconnect	**sybase_pconnect()**	Sybase
mssql_query	**sybase_query()**	Sybase
mssql_query	**sybase_query()**	Sybase
mssql_result	**sybase_result()**	Sybase
mssql_result	**sybase_result()**	Sybase
mssql_select_db	**sybase_select_db()**	Sybase
mssql_select_db	**sybase_select_db()**	Sybase
multcolor	**swfdisplayitem_multColor()**	Ming (flash)
mysql	**mysql_db_query()**	MySQL
mysql_createdb	**mysql_create_db()**	MySQL
mysql_db_name	**mysql_result()**	MySQL
mysql_dbname	**mysql_result()**	MySQL
mysql_dropdb	**mysql_drop_db()**	MySQL
mysql_fieldflags	**mysql_field_flags()**	MySQL
mysql_fieldlen	**mysql_field_len()**	MySQL
mysql_fieldname	**mysql_field_name()**	MySQL
mysql_fieldtable	**mysql_field_table()**	MySQL
mysql_fieldtype	**mysql_field_type()**	MySQL
mysql_freeresult	**mysql_free_result()**	MySQL
mysql_listdbs	**mysql_list_dbs()**	MySQL
mysql_listfields	**mysql_list_fields()**	MySQL
mysql_listtables	**mysql_list_tables()**	MySQL
mysql_numfields	**mysql_num_fields()**	MySQL
mysql_numrows	**mysql_num_rows()**	MySQL
mysql_selectdb	**mysql_select_db()**	MySQL
mysql_tablename	**mysql_result()**	MySQL
name	**domxml_attrname()**	DOM XML
new_child	**domxml_new_child()**	DOM XML
new_xmldoc	**domxml_new_xmldoc()**	DOM XML
nextframe	**swfmovie_nextFrame()**	Ming (flash)
nextframe	**swfsprite_nextFrame()**	Ming (flash)
node	**domxml_node()**	DOM XML
oci8append	**ocicollappend()**	OCI8
oci8assign	**ocicollassign()**	OCI8
oci8assignelem	**ocicollassignelem()**	OCI8
oci8close	**ocicloselob()**	OCI8
oci8free	**ocifreecoll()**	OCI8
oci8free	**ocifreedesc()**	OCI8
oci8getelem	**ocicollgetelem()**	OCI8

Alias	Master function	Extension used
oci8load	**ociloadlob()**	OCI8
oci8max	**ocicollmax()**	OCI8
oci8ocifreecursor	**ocifreestatement()**	OCI8
oci8save	**ocisavelob()**	OCI8
oci8savefile	**ocisavelobfile()**	OCI8
oci8size	**ocicollsize()**	OCI8
oci8trim	**ocicolltrim()**	OCI8
oci8writetemporary	**ociwritetemporarylob()**	OCI8
oci8writetofile	**ociwritelobtofile()**	OCI8
odbc_do	**odbc_exec()**	OCI8
odbc_field_precision	**odbc_field_len()**	OCI8
output	**swfmovie_output()**	Ming (flash)
parent	**domxml_parent()**	DOM XML
pdf_add_outline	**pdf_add_bookmark()**	PDF
pg_clientencoding	**pg_client_encoding()**	PostgreSQL
pg_setclientencoding	**pg_set_client_encoding()**	PostgreSQL
pos	**current()**	Base syntax
recode	**recode_string()**	Recode
remove	**swfmovie_remove()**	Ming (flash)
remove	**swfsprite_remove()**	Ming (flash)
rewind	**rewinddir()**	Base syntax
root	**domxml_root()**	DOM XML
rotate	**swfdisplayitem_rotate()**	Ming (flash)
rotateto	**swfdisplayitem_rotateTo()**	Ming (flash)
rotateto	**swffill_rotateTo()**	Ming (flash)
save	**swfmovie_save()**	Ming (flash)
savetofile	**swfmovie_saveToFile()**	Ming (flash)
scale	**swfdisplayitem_scale()**	Ming (flash)
scaleto	**swfdisplayitem_scaleTo()**	Ming (flash)
scaleto	**swffill_scaleTo()**	Ming (flash)
set_attribute	**domxml_set_attribute()**	DOM XML
set_content	**domxml_set_content()**	DOM XML
setaction	**swfbutton_setAction()**	Ming (flash)
setattr	**domxml_set_attribute()**	DOM XML
setbackground	**swfmovie_setBackground()**	Ming (flash)
setbounds	**swftextfield_setBounds()**	Ming (flash)
setcolor	**swftext_setColor()**	Ming (flash)
setcolor	**swftextfield_setColor()**	Ming (flash)
setdepth	**swfdisplayitem_setDepth()**	Ming (flash)
setdimension	**swfmovie_setDimension()**	Ming (flash)
setdown	**swfbutton_setDown()**	Ming (flash)
setfont	**swftext_setFont()**	Ming (flash)
setfont	**swftextfield_setFont()**	Ming (flash)
setframes	**swfmovie_setFrames()**	Ming (flash)
setframes	**swfsprite_setFrames()**	Ming (flash)
setheight	**swftext_setHeight()**	Ming (flash)
setheight	**swftextfield_setHeight()**	Ming (flash)
sethit	**swfbutton_setHit()**	Ming (flash)
setindentation	**swftextfield_setIndentation()**	Ming (flash)
setleftfill	**swfshape_setleftfill()**	Ming (flash)
setleftmargin	**swftextfield_setLeftMargin()**	Ming (flash)

Alias	Master function	Extension used
setline	**swfshape_setline()**	Ming (flash)
setlinespacing	**swftextfield_setLineSpacing()**	Ming (flash)
setmargins	**swftextfield_setMargins()**	Ming (flash)
setmatrix	**swfdisplayitem_setMatrix()**	Ming (flash)
setname	**swfdisplayitem_setName()**	Ming (flash)
setname	**swftextfield_setName()**	Ming (flash)
setover	**swfbutton_setOver()**	Ming (flash)
setrate	**swfmovie_setRate()**	Ming (flash)
setratio	**swfdisplayitem_setRatio()**	Ming (flash)
setrightfill	**swfshape_setrightfill()**	Ming (flash)
setrightmargin	**swftextfield_setRightMargin()**	Ming (flash)
setspacing	**swftext_setSpacing()**	Ming (flash)
setup	**swfbutton_setUp()**	Ming (flash)
show_source	**highlight_file ()**	Base syntax
sizeof	**count()**	Base syntax
skewx	**swfdisplayitem_skewX()**	Ming (flash)
skewxto	**swfdisplayitem_skewXTo()**	Ming (flash)
skewxto	**swffill_skewXTo()**	Ming (flash)
skewy	**swfdisplayitem_skewY()**	Ming (flash)
skewyto	**swfdisplayitem_skewYTo()**	Ming (flash)
skewyto	**swffill_skewYTo()**	Ming (flash)
snmpwalkoid	**snmprealwalk()**	SNMP
strchr	**strstr()**	Base syntax
streammp3	**swfmovie_streamMp3()**	Ming (flash)
swfaction	**swfaction_init()**	Ming (flash)
swfbitmap	**swfbitmap_init()**	Ming (flash)
swfbutton	**swfbutton_init()**	Ming (flash)
swffill	**swffill_init()**	Ming (flash)
swffont	**swffont_init()**	Ming (flash)
swfgradient	**swfgradient_init()**	Ming (flash)
swfmorph	**swfmorph_init()**	Ming (flash)
swfmovie	**swfmovie_init()**	Ming (flash)
swfshape	**swfshape_init()**	Ming (flash)
swfsprite	**swfsprite_init()**	Ming (flash)
swftext	**swftext_init()**	Ming (flash)
swftextfield	**swftextfield_init()**	Ming (flash)
unlink	**domxml_unlink_node()**	DOM XML
xptr_new_context	**xpath_new_context()**	DOM XML

Appendix G. List of Reserved Words

The following is a listing of predefined identifiers in PHP. None of the identifiers listed here should be used as identifiers in a your scripts. These lists include keywords and predefined variable, constant, and class names. These lists are neither exhaustive or complete.

List of Keywords

These words have special meaning in PHP. Some of them represent things which look like functions, some look like constants, and so on--but they're not, really: they are language constructs. You cannot use any of the following words as constants, class

names, or function names. Using them as variable names is generally OK, but could lead to confusion.

Table G-1. PHP Keywords

and	or	xor	__FILE__	
__LINE__	**array()**	as	break	case
cfunction	class	const	continue	declare
default	**die()**	do	**echo()**	else
elseif	**empty()**	enddeclare	endfor	endforeach
endif	endswitch	endwhile	eval	**exit()**
extends	for	foreach	function	global
if	**include()**	**include_once()**	**isset()**	**list()**
new	old_function	**print()**	**require()**	**require_once()**
return()	static	switch	**unset()**	use
var	while	**__FUNCTION__**	**__CLASS__**	

Predefined Variables

Server variables: $_SERVER

> **Note:** Introduced in 4.1.0. In earlier versions, use $HTTP_SERVER_VARS.

$_SERVER is an array containing information such as headers, paths, and script locations. The entries in this array are created by the webserver. There is no guarantee that every webserver will provide any of these; servers may omit some, or provide others not listed here. That said, a large number of these variables are accounted for in the CGI 1.1 specification, so you should be able to expect those.

This is a 'superglobal', or automatic global, variable. This simply means that it is available in all scopes throughout a script. You don't need to do a **global $_SERVER;** to access it within functions or methods, as you do with $HTTP_SERVER_VARS.

$HTTP_SERVER_VARS contains the same initial information, but is not an autoglobal. (Note that $HTTP_SERVER_VARS and $_SERVER are different variables and that PHP handles them as such)

If the register_globals directive is set, then these variables will also be made available in the global scope of the script; i.e., separate from the $_SERVER and $HTTP_SERVER_VARS arrays. For related information, see the security chapter titled Using Register Globals. These individual globals are not autoglobals.

You may or may not find any of the following elements in $_SERVER. Note that few, if any, of these will be available (or indeed have any meaning) if running PHP on the command line.

'PHP_SELF'

> The filename of the currently executing script, relative to the document root. For instance, `$_SERVER['PHP_SELF']` in a script at the address `http://example.com/test.php/foo.bar` would be `/test.php/foo.bar`.
>
> If PHP is running as a command-line processor, this variable is not available.

'argv'

> Array of arguments passed to the script. When the script is run on the command line, this gives C-style access to the command line parameters. When called via the GET method, this will contain the query string.

'argc'

> Contains the number of command line parameters passed to the script (if run on the command line).

'GATEWAY_INTERFACE'

> What revision of the CGI specification the server is using; i.e. '`CGI/1.1`'.

'SERVER_NAME'

> The name of the server host under which the current script is executing. If the script is running on a virtual host, this will be the value defined for that virtual host.

'SERVER_SOFTWARE'

Server identification string, given in the headers when responding to requests.

'SERVER_PROTOCOL'

Name and revision of the information protocol via which the page was requested; i.e. 'HTTP/1.0';

'REQUEST_METHOD'

Which request method was used to access the page; i.e. 'GET', 'HEAD', 'POST', 'PUT'.

'QUERY_STRING'

The query string, if any, via which the page was accessed.

'DOCUMENT_ROOT'

The document root directory under which the current script is executing, as defined in the server's configuration file.

'HTTP_ACCEPT'

Contents of the Accept: header from the current request, if there is one.

'HTTP_ACCEPT_CHARSET'

Contents of the Accept-Charset: header from the current request, if there is one. Example: 'iso-8859-1,*,utf-8'.

'HTTP_ACCEPT_ENCODING'

Contents of the Accept-Encoding: header from the current request, if there is one. Example: 'gzip'.

'HTTP_ACCEPT_LANGUAGE'

Contents of the Accept-Language: header from the current request, if there is one. Example: 'en'.

'HTTP_CONNECTION'

Contents of the Connection: header from the current request, if there is one. Example: 'Keep-Alive'.

'HTTP_HOST'

Contents of the Host: header from the current request, if there is one.

'HTTP_REFERER'

The address of the page (if any) which referred the user agent to the current page. This is set by the user agent. Not all user agents will set this, and some provide the ability to modify HTTP_REFERER as a feature. In short, it cannot really be trusted.

'HTTP_USER_AGENT'

Contents of the User_Agent: header from the current request, if there is one. This is a string denoting the user agent being which is accessing the page. A typical example is: Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586). Among other things, you can use this value with **get_browser()** to tailor your page's output to the capabilities of the user agent.

'REMOTE_ADDR'

The IP address from which the user is viewing the current page.

'REMOTE_HOST'

The Host name from which the user is viewing the current page. The reverse dns lookup is based off the REMOTE_ADDR of the user.

> **Note:** Your web server must be configured to create this variable. For example in Apache you'll need HostnameLookups On inside httpd.conf for it to exist. See also **gethostbyaddr()**.

'REMOTE_PORT'

The port being used on the user's machine to communicate with the web server.

'SCRIPT_FILENAME'

The absolute pathname of the currently executing script.

'SERVER_ADMIN'

The value given to the SERVER_ADMIN (for Apache) directive in the web server configuration file. If the script is running on a virtual host, this will be the value defined for that virtual host.

'SERVER_PORT'

The port on the server machine being used by the web server for communication. For default setups, this will be '80'; using SSL, for instance, will change this to whatever your defined secure HTTP port is.

'SERVER_SIGNATURE'

String containing the server version and virtual host name which are added to server-generated pages, if enabled.

'PATH_TRANSLATED'

Filesystem- (not document root-) based path to the current script, after the server has done any virtual-to-real mapping.

'SCRIPT_NAME'

Contains the current script's path. This is useful for pages which need to point to themselves.

'REQUEST_URI'

The URI which was given in order to access this page; for instance, '/index.html'.

'PHP_AUTH_USER'

When running under Apache as module doing HTTP authentication this variable is set to the username provided by the user.

'PHP_AUTH_PW'

When running under Apache as module doing HTTP authentication this variable is set to the password provided by the user.

'PHP_AUTH_TYPE'

When running under Apache as module doing HTTP authenticated this variable is set to the authentication type.

Environment variables: $_ENV

Note: Introduced in 4.1.0. In earlier versions, use $HTTP_ENV_VARS.

These variables are imported into PHP's global namespace from the environment under which the PHP parser is running. Many are provided by the shell under which PHP is running and different systems are likely running different kinds of shells, a definitive list is impossible. Please see your shell's documentation for a list of defined environment variables.

Other environment variables include the CGI variables, placed there regardless of whether PHP is running as a server module or CGI processor.

This is a 'superglobal', or automatic global, variable. This simply means that it is available in all scopes throughout a script. You don't need to do a **global $_ENV;** to access it within functions or methods, as you do with $HTTP_ENV_VARS.

$HTTP_ENV_VARS contains the same initial information, but is not an autoglobal. (Note that HTTP_ENV_VARS and $_ENV are different variables and that PHP handles them as such)

If the register_globals directive is set, then these variables will also be made available in the global scope of the script; i.e., separate from the $_ENV and $HTTP_ENV_VARS arrays. For related information, see the security chapter titled Using Register Globals. These individual globals are not autoglobals.

HTTP Cookies: $_COOKIE

Note: Introduced in 4.1.0. In earlier versions, use $HTTP_COOKIE_VARS.

An associative array of variables passed to the current script via HTTP cookies. Automatically global in any scope.

This is a 'superglobal', or automatic global, variable. This simply means that it is available in all scopes throughout a script. You don't need to do a **global $_COOKIE;** to access it within functions or methods, as you do with $HTTP_COOKIE_VARS.

$HTTP_COOKIE_VARS contains the same initial information, but is not an autoglobal. (Note that HTTP_COOKIE_VARS and $_COOKIE are different variables and that PHP handles them as such)

If the register_globals directive is set, then these variables will also be made available in the global scope of the script; i.e., separate from the $_COOKIE and $HTTP_COOKIE_VARS arrays. For related information, see the security chapter titled Using Register Globals. These individual globals are not autoglobals.

HTTP GET variables: $_GET

Note: Introduced in 4.1.0. In earlier versions, use $HTTP_GET_VARS.

An associative array of variables passed to the current script via the HTTP GET method. Automatically global in any scope.

This is a 'superglobal', or automatic global, variable. This simply means that it is available in all scopes throughout a script. You don't need to do a **global $_GET;** to access it within functions or methods, as you do with $HTTP_GET_VARS.

$HTTP_GET_VARS contains the same initial information, but is not an autoglobal. (Note that HTTP_GET_VARS and $_GET are different variables and that PHP handles them as such)

If the register_globals directive is set, then these variables will also be made available in the global scope of the script; i.e., separate from the $_GET and $HTTP_GET_VARS arrays. For related information, see the security chapter titled Using Register Globals. These individual globals are not autoglobals.

HTTP POST variables: $_POST

Note: Introduced in 4.1.0. In earlier versions, use $HTTP_POST_VARS.

An associative array of variables passed to the current script via the HTTP POST method. Automatically global in any scope.

This is a 'superglobal', or automatic global, variable. This simply means that it is available in all scopes throughout a script. You don't need to do a **global $_POST;** to access it within functions or methods, as you do with $HTTP_POST_VARS.

$HTTP_POST_VARS contains the same initial information, but is not an autoglobal. (Note that HTTP_POST_VARS and $_POST are different variables and that PHP handles them as such)

If the register_globals directive is set, then these variables will also be made available in the global scope of the script; i.e., separate from the $_POST and $HTTP_POST_VARS arrays. For related information, see the security chapter titled Using Register Globals. These individual globals are not autoglobals.

HTTP File upload variables: $_FILES

Note: Introduced in 4.1.0. In earlier versions, use $HTTP_POST_FILES.

An associative array of items uploaded to the current script via the HTTP POST method. Automatically global in any scope.

This is a 'superglobal', or automatic global, variable. This simply means that it is available in all scopes throughout a script. You don't need to do a **global $_FILES;** to access it within functions or methods, as you do with $HTTP_POST_FILES.

$HTTP_POST_FILES contains the same information, but is not an autoglobal.

If the register_globals directive is set, then these variables will also be made available in the global scope of the script; i.e., separate from the $_FILES and $HTTP_POST_FILES arrays. For related information, see the security chapter titled Using Register Globals. These individual globals are not autoglobals.

Request variables: $_REQUEST

Note: Introduced in 4.1.0. There is no equivalent array in earlier versions.

An associative array consisting of the contents of $_GET, $_POST, and $_COOKIE.

Note: Prior to PHP 4.3.0, $_FILES information was also included into $_REQUEST.

This is a 'superglobal', or automatic global, variable. This simply means that it is available in all scopes throughout a script. You don't need to do a **global $_REQUEST;** to access it within functions or methods.

If the register_globals directive is set, then these variables will also be made available in the global scope of the script; i.e., separate from the $_REQUEST array. For related information, see the security chapter titled Using Register Globals. These

individual globals are not autoglobals.

Session variables: `$_SESSION`

> **Note:** Introduced in 4.1.0. In earlier versions, use `$HTTP_SESSION_VARS`.

An associative array containing session variables available to the current script. See the Session functions documentation for more information on how this is used.

This is a 'superglobal', or automatic global, variable. This simply means that it is available in all scopes throughout a script. You don't need to do a **global $_SESSION;** to access it within functions or methods, as you do with `$HTTP_SESSION_VARS`.

`$HTTP_SESSION_VARS` contains the same information, but is not an autoglobal.

If the register_globals directive is set, then these variables will also be made available in the global scope of the script; i.e., separate from the `$_SESSION` and `$HTTP_SESSION_VARS` arrays. For related information, see the security chapter titled Using Register Globals. These individual globals are not autoglobals.

Global variables: `$GLOBALS`

> **Note:** `$GLOBALS` has been available since PHP 3.0.0.

An associative array containing references to all variables which are currently defined in the global scope of the script. The variable names are the keys of the array.

This is a 'superglobal', or automatic global, variable. This simply means that it is available in all scopes throughout a script. You don't need to do a **global $GLOBALS;** to access it within functions or methods.

The previous error message: `$php_errormsg`

`$php_errormsg` is a variable containing the text of the last error message generated by PHP. This variable will only be available within the scope in which the error occurred, and only if the track_errors configuration option is turned on (it defaults to off).

Predefined Classes

Standard Defined Classes

These classes are defined in the standard set of functions included in the PHP build.

Directory

> The class from which **dir()** is instantiated.

stdClass

Ming Defined Classes

These classes are defined in the Ming extension, and will only be available when that extension has either been compiled into PHP or dynamically loaded at runtime.

swfshape

swffill

swfgradient

swfbitmap

swftext

swftextfield

swffont

swfdisplayitem

swfmovie

swfbutton

swfaction

swfmorph

swfsprite

Oracle 8 Defined Classes

These classes are defined in the Oracle 8 extension, and will only be available when that extension has either been compiled into PHP or dynamically loaded at runtime.

OCI-Lob

OCI-Collection

---

qtdom Defined Classes

These classes are defined in the qtdom extension, and will only be available when that extension has either been compiled into PHP or dynamically loaded at runtime.

QDomDocument

QDomNode

Predefined Constants

Table of Contents

Core Predefined Constants

Core Predefined Constants -- Constants defined in the PHP core, Zend, and SAPI modules

Description

These constants are defined by the PHP core. This includes PHP, the Zend engine, and SAPI modules.

PHP_VERSION (string)

PHP_OS (string)

DEFAULT_INCLUDE_PATH (string)

PEAR_INSTALL_DIR (string)

PEAR_EXTENSION_DIR (string)

PHP_EXTENSION_DIR (string)

PHP_BINDIR (string)

PHP_LIBDIR (string)

PHP_DATADIR (string)

PHP_SYSCONFDIR (string)

PHP_LOCALSTATEDIR (string)

PHP_CONFIG_FILE_PATH (string)

PHP_OUTPUT_HANDLER_START (integer)

PHP_OUTPUT_HANDLER_CONT (integer)

PHP_OUTPUT_HANDLER_END (integer)

E_ERROR (integer)

E_WARNING (integer)

E_PARSE (integer)

E_NOTICE (integer)

E_CORE_ERROR (integer)

E_CORE_WARNING (integer)

E_COMPILE_ERROR (integer)

E_COMPILE_WARNING (integer)

E_USER_ERROR (integer)

E_USER_WARNING (integer)

E_USER_NOTICE (integer)

E_ALL (integer)

Standard Predefined Constants

Standard Predefined Constants -- Constants defined in PHP by default

Description

These constants are defined in PHP by default.

EXTR_OVERWRITE (integer)

EXTR_SKIP (integer)

EXTR_PREFIX_SAME (integer)

EXTR_PREFIX_ALL (integer)

EXTR_PREFIX_INVALID (integer)

EXTR_PREFIX_IF_EXISTS (integer)

EXTR_IF_EXISTS (integer)

SORT_ASC (integer)

SORT_DESC (integer)

SORT_REGULAR (integer)

SORT_NUMERIC (integer)

SORT_STRING (integer)

CASE_LOWER (integer)

CASE_UPPER (integer)

COUNT_NORMAL (integer)

COUNT_RECURSIVE (integer)

ASSERT_ACTIVE (integer)

ASSERT_CALLBACK (integer)

ASSERT_BAIL (integer)

ASSERT_WARNING (integer)

ASSERT_QUIET_EVAL (integer)

CONNECTION_ABORTED (integer)

CONNECTION_NORMAL (integer)

CONNECTION_TIMEOUT (integer)

INI_USER (integer)

INI_PERDIR (integer)

INI_SYSTEM (integer)

INI_ALL (integer)

M_E (float)

M_LOG2E (float)

M_LOG10E (float)

M_LN2 (float)

M_LN10 (float)

M_PI (float)

M_PI_2 (float)

M_PI_4 (float)

M_1_PI (float)

M_2_PI (float)

M_2_SQRTPI (float)

M_SQRT2 (float)

M_SQRT1_2 (float)

CRYPT_SALT_LENGTH (integer)

CRYPT_STD_DES (integer)

CRYPT_EXT_DES (integer)

CRYPT_MD5 (integer)

CRYPT_BLOWFISH (integer)

DIRECTORY_SEPARATOR (integer)

SEEK_SET (integer)

SEEK_CUR (integer)

SEEK_END (integer)

LOCK_SH (integer)

LOCK_EX (integer)

LOCK_UN (integer)

LOCK_NB (integer)

HTML_SPECIALCHARS (integer)

HTML_ENTITIES (integer)

ENT_COMPAT (integer)

ENT_QUOTES (integer)

ENT_NOQUOTES (integer)

INFO_GENERAL (integer)

INFO_CREDITS (integer)

INFO_CONFIGURATION (integer)

INFO_MODULES (integer)

INFO_ENVIRONMENT (integer)

INFO_VARIABLES (integer)

INFO_LICENSE (integer)

INFO_ALL (integer)

CREDITS_GROUP (integer)

CREDITS_GENERAL (integer)

CREDITS_SAPI (integer)

CREDITS_MODULES (integer)

CREDITS_DOCS (integer)

CREDITS_FULLPAGE (integer)

CREDITS_QA (integer)

CREDITS_ALL (integer)

STR_PAD_LEFT (integer)

STR_PAD_RIGHT (integer)

STR_PAD_BOTH (integer)

PATHINFO_DIRNAME (integer)

PATHINFO_BASENAME (integer)

PATHINFO_EXTENSION (integer)

CHAR_MAX (integer)

LC_CTYPE (integer)

LC_NUMERIC (integer)

LC_TIME (integer)

LC_COLLATE (integer)

LC_MONETARY (integer)

LC_ALL (integer)

LC_MESSAGES (integer)

ABDAY_1 (integer)

ABDAY_2 (integer)

ABDAY_3 (integer)

ABDAY_4 (integer)

ABDAY_5 (integer)

ABDAY_6 (integer)

ABDAY_7 (integer)

DAY_1 (integer)

DAY_2 (integer)

DAY_3 (integer)

DAY_4 (integer)

DAY_5 (integer)

DAY_6 (integer)

DAY_7 (integer)

ABMON_1 (integer)

ABMON_2 (integer)

ABMON_3 (integer)

ABMON_4 (integer)

ABMON_5 (integer)

ABMON_6 (integer)

ABMON_7 (integer)

ABMON_8 (integer)

ABMON_9 (integer)

ABMON_10 (integer)

ABMON_11 (integer)

ABMON_12 (integer)

MON_1 (integer)

MON_2 (integer)

MON_3 (integer)

MON_4 (integer)

MON_5 (integer)

MON_6 (integer)

MON_7 (integer)

MON_8 (integer)

MON_9 (integer)

MON_10 (integer)

MON_11 (integer)

MON_12 (integer)

AM_STR (integer)

PM_STR (integer)

D_T_FMT (integer)

D_FMT (integer)

T_FMT (integer)

T_FMT_AMPM (integer)

ERA (integer)

ERA_YEAR (integer)

ERA_D_T_FMT (integer)

ERA_D_FMT (integer)

ERA_T_FMT (integer)

ALT_DIGITS (integer)

INT_CURR_SYMBOL (integer)

CURRENCY_SYMBOL (integer)

CRNCYSTR (integer)

MON_DECIMAL_POINT (integer)

MON_THOUSANDS_SEP (integer)

MON_GROUPING (integer)

POSITIVE_SIGN (integer)

NEGATIVE_SIGN (integer)

INT_FRAC_DIGITS (integer)

FRAC_DIGITS (integer)

P_CS_PRECEDES (integer)

P_SEP_BY_SPACE (integer)

N_CS_PRECEDES (integer)

N_SEP_BY_SPACE (integer)

P_SIGN_POSN (integer)

N_SIGN_POSN (integer)

DECIMAL_POINT (integer)

RADIXCHAR (integer)

THOUSANDS_SEP (integer)

THOUSEP (integer)

GROUPING (integer)

YESEXPR (integer)

NOEXPR (integer)

YESSTR (integer)

NOSTR (integer)

CODESET (integer)

LOG_EMERG (integer)

LOG_ALERT (integer)

LOG_CRIT (integer)

LOG_ERR (integer)

LOG_WARNING (integer)

LOG_NOTICE (integer)

LOG_INFO (integer)

LOG_DEBUG (integer)

LOG_KERN (integer)

LOG_USER (integer)

LOG_MAIL (integer)

LOG_DAEMON (integer)

LOG_AUTH (integer)

LOG_SYSLOG (integer)

LOG_LPR (integer)

LOG_NEWS (integer)

LOG_UUCP (integer)

LOG_CRON (integer)

LOG_AUTHPRIV (integer)

LOG_LOCAL0 (integer)

LOG_LOCAL1 (integer)

LOG_LOCAL2 (integer)

LOG_LOCAL3 (integer)

LOG_LOCAL4 (integer)

LOG_LOCAL5 (integer)

LOG_LOCAL6 (integer)

LOG_LOCAL7 (integer)

LOG_PID (integer)

LOG_CONS (integer)

LOG_ODELAY (integer)

LOG_NDELAY (integer)

LOG_NOWAIT (integer)

LOG_PERROR (integer)

Appendix H. List of Resource Types

The following is a list of functions which create, use or destroy PHP resources. The function **is_resource()** can be used to determine if a variable is a resource and **get_resource_type()** will return the type of resource it is.

Table H-1. Resource Types

Resource Type Name	Created By	Used By	Destroyed By	Definition
aspell	**aspell_new()**	**aspell_check()**, **aspell_check_raw()**, **aspell_suggest()**	None	Aspell dictionary

Resource Type Name	Created By	Used By	Destroyed By	Definition
bzip2	bzopen()	bzerrno(), bzerror(), bzerrstr(), bzflush(), bzread(), bzwrite()	bzclose()	Bzip2 file
COM	com_load()	com_invoke(), com_propget(), com_get(), com_propput(), com_set(), com_propput()	None	COM object reference
VARIANT				
cpdf	cpdf_open()	cpdf_page_init(), cpdf_finalize_page(), cpdf_finalize(), cpdf_output_buffer(), cpdf_save_to_file(), cpdf_set_current_page(), cpdf_begin_text(), cpdf_end_text(), cpdf_show(), cpdf_show_xy(), cpdf_text(), cpdf_set_font(), cpdf_set_leading(), cpdf_set_text_rendering(), cpdf_set_horiz_scaling(), cpdf_set_text_rise(), cpdf_set_text_matrix(), cpdf_set_text_pos(), cpdf_set_text_pos(), cpdf_set_word_spacing(), cpdf_continue_text(), cpdf_stringwidth(), cpdf_save(), cpdf_translate(), cpdf_restore(), cpdf_scale(), cpdf_rotate(), cpdf_setflat(), cpdf_setlinejoin(), cpdf_setlinecap(), cpdf_setmiterlimit(), cpdf_setlinewidth(), cpdf_setdash(), cpdf_moveto(), cpdf_rmoveto(), cpdf_curveto(), cpdf_lineto(), cpdf_rlineto(), cpdf_circle(), cpdf_arc(), cpdf_rect(), cpdf_closepath(), cpdf_stroke(), cpdf_closepath_fill_stroke(), cpdf_fill_stroke(), cpdf_clip(), cpdf_fill(), cpdf_setgray_fill(), cpdf_setgray_stroke(), cpdf_setgray(), cpdf_setrgbcolor_fill(), cpdf_setrgbcolor_stroke(), cpdf_setrgbcolor(), cpdf_add_outline(), cpdf_set_page_animation(), cpdf_import_jpeg(), cpdf_place_inline_image(), cpdf_add_annotation()	cpdf_close()	PDF document with CPDF lib
cpdf outline				
curl	curl_init()	curl_init(), curl_exec()	curl_close()	Curl session
dbm	dbmopen()	dbmexists(), dbmfetch(), dbminsert(), dbmreplace(), dbmdelete(), dbmfirstkey(), dbmnextkey()	dbmclose()	Link to DBM database
dba	dba_open()	dba_delete(), dba_exists(), dba_fetch(), dba_firstkey(), dba_insert(), dba_nextkey(), dba_optimize(), dba_replace(), dba_sync()	dba_close()	Link to DBA database
dba persistent	dba_popen()	dba_delete(), dba_exists(), dba_fetch(), dba_firstkey(), dba_insert(), dba_nextkey(), dba_optimize(), dba_replace(), dba_sync()	None	Persistent link to DBA database
dbase	dbase_open()	dbase_pack(), dbase_add_record(), dbase_replace_record(), dbase_delete_record(), dbase_get_record(), dbase_get_record_with_names(), dbase_numfields(), dbase_numrecords()	dbase_close()	Link to Dbase database
dbx_link_object	dbx_connect()	dbx_query()	dbx_close()	dbx connection
dbx_result_object	dbx_query()	()	None	dbx result
domxml attribute				

Resource Type Name	Created By	Used By	Destroyed By	Definition
domxml document				
domxml node				
xpath context				
xpath object				
fbsql database	fbsql_select_db()	()	None	fbsql database
fbsql link	fbsql_change_user(), fbsql_connect()	fbsql_autocommit(), fbsql_change_user(), fbsql_create_db(), fbsql_data_seek(), fbsql_db_query(), fbsql_drop_db(), (), fbsql_select_db(), fbsql_errno(), fbsql_error(), fbsql_insert_id(), fbsql_list_dbs()	fbsql_close()	Link to fbsql database
fbsql plink	fbsql_change_user(), fbsql_pconnect()	fbsql_autocommit(), fbsql_change_user(), fbsql_create_db(), fbsql_data_seek(), fbsql_db_query(), fbsql_drop_db(), (), fbsql_select_db(), fbsql_errno(), fbsql_error(), fbsql_insert_id(), fbsql_list_dbs()	None	Persistent link to fbsql database
fbsql result	fbsql_db_query(), fbsql_list_dbs(), fbsql_query(), fbsql_list_fields(), fbsql_list_tables(), fbsql_tablename()	fbsql_affected_rows(), fbsql_fetch_array(), fbsql_fetch_assoc(), fbsql_fetch_field(), fbsql_fetch_lengths(), fbsql_fetch_object(), fbsql_fetch_row(), fbsql_field_flags(), fbsql_field_name(), fbsql_field_len(), fbsql_field_seek(), fbsql_field_table(), fbsql_field_type(), fbsql_next_result(), fbsql_num_fields(), fbsql_num_rows(), fbsql_result(), fbsql_num_rows()	fbsql_free_result()	fbsql result
fdf	fdf_open()	fdf_create(), fdf_save(), fdf_get_value(), fdf_set_value(), fdf_next_field_name(), fdf_set_ap(), fdf_set_status(), fdf_get_status(), fdf_set_file(), fdf_get_file(), fdf_set_flags(), fdf_set_opt(), fdf_set_submit_form_action(), fdf_set_javascript_action()	fdf_close()	FDF File
ftp	ftp_connect()	ftp_login(), ftp_pwd(), ftp_cdup(), ftp_chdir(), ftp_mkdir(), ftp_rmdir(), ftp_nlist(), ftp_rawlist(), ftp_systype(), ftp_pasv(), ftp_get(), ftp_fget(), ftp_put(), ftp_fput(), ftp_size(), ftp_mdtm(), ftp_rename(), ftp_delete(), ftp_site()	ftp_quit()	FTP stream
gd	imagecreate(), imagecreatefromgif(), imagecreatefromjpeg(), imagecreatefrompng(), imagecreatefromwbmp(), imagecreatefromstring(), imagecreatetruecolor()	imagearc(), imagechar(), imagecharup(), imagecolorallocate(), imagecolorat(), imagecolorclosest(), imagecolorexact(), imagecolorresolve(), imagegammacorrect(), imagegammacorrect(), imagecolorset(), imagecolorsforindex(), imagecolorstotal(), imagecolortransparent(), imagecopy(), imagecopyresized(), imagedashedline(), imagefill(), imagefilledpolygon(), imagefilledrectangle(), imagefilltoborder(), imagegif(), imagepng(), imagejpeg(), imagewbmp(), imageinterlace(), imageline(), imagepolygon(), imagepstext(), imagerectangle(), imagesetpixel(), imagestring(), imagestringup(), imagesx(), imagesy(), imagettftext(), imagefilledarc(), imageellipse(), imagefilledellipse(), imagecolorclosestalpha(), imagecolorexactalpha(),	imagedestroy()	GD Image

Resource Type Name	Created By	Used By	Destroyed By	Definition
		imagecolorresolvealpha(), imagecopymerge(), imagecopymergegray(), imagecopyresampled(), imagetruecolortopalette(), imagesetbrush(), imagesettile(), imagesetthickness()		
gd font	imageloadfont()	imagechar(), imagecharup(), imagefontheight()	None	Font for GD
gd PS encoding				
gd PS font	imagepsloadfont()	imagepstext(), imagepsslantfont(), imagepsextendfont(), imagepsencodefont(), imagepsbbox()	imagepsfreefont()	PS font for GD
GMP integer	gmp_init()	gmp_intval(), gmp_strval(), gmp_add(), gmp_sub(), gmp_mul(), gmp_div_q(), gmp_div_r(), gmp_div_qr(), gmp_div(), gmp_mod(), gmp_divexact(), gmp_cmp(), gmp_neg(), gmp_abs(), gmp_sign(), gmp_fact(), gmp_sqrt(), gmp_sqrtrm(), gmp_perfect_square(), gmp_pow(), gmp_powm(), gmp_prob_prime(), gmp_gcd(), gmp_gcdext(), gmp_invert(), gmp_legendre(), gmp_jacobi(), gmp_random(), gmp_and(), gmp_or(), gmp_xor(), gmp_setbit(), gmp_clrbit(), gmp_scano(), gmp_scan1(), gmp_popcount(), gmp_hamdist()	None	GMP Number
hyperwave document	hw_cp(), hw_docbyanchor(), hw_getremote(), hw_getremotechildren()	hw_children(), hw_childrenobj(), hw_getparents(), hw_getparentsobj(), hw_getchildcoll(), hw_getchildcollobj(), hw_getremote(), hw_getsrcbydestobj(), hw_getandlock(), hw_gettext(), hw_getobjectbyquerycoll(), hw_getobjectbyquerycollobj(), hw_getchilddoccoll(), hw_getchilddoccollobj(), hw_getanchors(), hw_getanchorsobj(), hw_inscoll(), hw_pipedocument(), hw_unlock()	hw_deleteobject()	Hyperwave object
hyperwave link	hw_connect()	hw_children(), hw_childrenobj(), hw_cp(), hw_deleteobject(), hw_docbyanchor(), hw_docbyanchorobj(), hw_errormsg(), hw_edittext(), hw_error(), hw_getparents(), hw_getparentsobj(), hw_getchildcoll(), hw_getchildcollobj(), hw_getremote(), hw_getremotechildren(), hw_getsrcbydestobj(), hw_getobject(), hw_getandlock(), hw_gettext(), hw_getobjectbyquery(), hw_getobjectbyqueryobj(), hw_getobjectbyquerycoll(), hw_getobjectbyquerycollobj(), hw_getchilddoccoll(), hw_getchilddoccollobj(), hw_getanchors(), hw_getanchorsobj(), hw_mv(), hw_incollections(), hw_info(), hw_inscoll(), hw_insdoc(), hw_insertdocument(), hw_insertobject(), hw_mapid(), hw_modifyobject(), hw_pipedocument(), hw_unlock(), hw_who(), hw_getusername()	hw_close(), hw_free_document()	Link to Hyperwave server

Resource Type Name	Created By	Used By	Destroyed By	Definition
hyperwave link persistent	hw_pconnect()	hw_children(), hw_childrenobj(), hw_cp(), hw_deleteobject(), hw_docbyanchor(), hw_docbyanchorobj(), hw_errormsg(), hw_edittext(), hw_error(), hw_getparents(), hw_getparentsobj(), hw_getchildcoll(), hw_getchildcollobj(), hw_getremote(), hw_getremotechildren(), hw_getsrcbydestobj(), hw_getobject(), hw_getandlock(), hw_gettext(), hw_getobjectbyquery(), hw_getobjectbyqueryobj(), hw_getobjectbyquerycoll(), hw_getobjectbyquerycollobj(), hw_getchilddoccoll(), hw_getchilddoccollobj(), hw_getanchors(), hw_getanchorsobj(), hw_mv(), hw_incollections(), hw_info(), hw_inscoll(), hw_insdoc(), hw_insertdocument(), hw_insertobject(), hw_mapid(), hw_modifyobject(), hw_pipedocument(), hw_unlock(), hw_who(), hw_getusername()	None	Persistent link to Hyperwave server
icap	**icap_open()**	**icap_fetch_event(), icap_list_events(), icap_store_event(), icap_snooze(), icap_list_alarms(), icap_delete_event()**	**icap_close()**	Link to icap server
imap	imap_open()	imap_append(), imap_body(), imap_check(), imap_createmailbox(), imap_delete(), imap_deletemailbox(), imap_expunge(), imap_fetchbody(), imap_fetchstructure(), imap_headerinfo(), imap_header(), imap_headers(), imap_listmailbox(), imap_getmailboxes(), imap_get_quota(), imap_status(), imap_listsubscribed(), imap_set_quota(), imap_set_quota(), imap_getsubscribed(), imap_mail_copy(), imap_mail_move(), imap_num_msg(), imap_num_recent(), imap_ping(), imap_renamemailbox(), imap_reopen(), imap_subscribe(), imap_undelete(), imap_unsubscribe(), imap_scanmailbox(), imap_mailboxmsginfo(), imap_fetchheader(), imap_uid(), imap_msgno(), imap_search(), imap_fetch_overview()	imap_close()	Link to IMAP, POP3 server
imap chain persistent				
imap persistent				
ingres	ingres_connect()	ingres_query(), ingres_num_rows(), ingres_num_fields(), ingres_field_name(), ingres_field_type(), ingres_field_nullable(), ingres_field_length(), ingres_field_precision(), ingres_field_scale(), ingres_fetch_array(), ingres_fetch_row(), ingres_fetch_object(), ingres_rollback(), ingres_commit(), ingres_autocommit()	ingres_close()	Link to ingresII base
ingres persistent	ingres_pconnect()	ingres_query(), ingres_num_rows(), ingres_num_fields(), ingres_field_name(), ingres_field_type(), ingres_field_nullable(), ingres_field_length(), ingres_field_precision(),	None	Persistent link to ingresII base

Resource Type Name	Created By	Used By	Destroyed By	Definition
		ingres_field_scale(), ingres_fetch_array(), ingres_fetch_row(), ingres_fetch_object(), ingres_rollback(), ingres_commit(), ingres_autocommit()		
interbase blob				
interbase link	ibase_connect()	ibase_query(), ibase_prepare(), ibase_trans()	ibase_close()	Link to Interbase database
interbase link persistent	ibase_pconnect()	ibase_query(), ibase_prepare(), ibase_trans()	None	Persistent link to Interbase database
interbase query	ibase_prepare()	ibase_execute()	ibase_free_query()	Interbase query
interbase result	ibase_query()	ibase_fetch_row(), ibase_fetch_object(), ibase_field_info(), ibase_num_fields()	ibase_free_result()	Interbase Result
interbase transaction	ibase_trans()	ibase_commit()	ibase_rollback()	Interbase transaction
java				
ldap link	ldap_connect(), ldap_search()	ldap_count_entries(), ldap_first_attribute(), ldap_first_entry(), ldap_get_attributes(), ldap_get_dn(), ldap_get_entries(), ldap_get_values(), ldap_get_values_len(), ldap_next_attribute(), ldap_next_entry()	ldap_close()	ldap connection
ldap result	ldap_read()	ldap_add(), ldap_compare(), ldap_bind(), ldap_count_entries(), ldap_delete(), ldap_errno(), ldap_error(), ldap_first_attribute(), ldap_first_entry(), ldap_get_attributes(), ldap_get_dn(), ldap_get_entries(), ldap_get_values(), ldap_get_values_len(), ldap_get_option(), ldap_list(), ldap_modify(), ldap_mod_add(), ldap_mod_replace(), ldap_next_attribute(), ldap_next_entry(), ldap_mod_del(), ldap_set_option(), ldap_unbind()	ldap_free_result()	ldap search result
ldap result entry				
mcal	mcal_open(), mcal_popen()	mcal_create_calendar(), mcal_rename_calendar(), mcal_rename_calendar(), mcal_delete_calendar(), mcal_fetch_event(), mcal_list_events(), mcal_append_event(), mcal_store_event(), mcal_delete_event(), mcal_list_alarms(), mcal_event_init(), mcal_event_set_category(), mcal_event_set_title(), mcal_event_set_description(), mcal_event_set_start(), mcal_event_set_end(), mcal_event_set_alarm(), mcal_event_set_class(), mcal_next_recurrence(), mcal_event_set_recur_none(), mcal_event_set_recur_daily(), mcal_event_set_recur_weekly(), mcal_event_set_recur_monthly_mday(), mcal_event_set_recur_monthly_wday(), mcal_event_set_recur_yearly(), mcal_fetch_current_stream_event(), mcal_event_add_attribute(), mcal_expunge()	mcal_close()	Link to calendar server
SWFAction				

Resource Type Name	Created By	Used By	Destroyed By	Definition
SWFBitmap				
SWFButton				
SWFDisplayItem				
SWFFill				
SWFFont				
SWFGradient				
SWFMorph				
SWFMovie				
SWFShape				
SWFSprite				
SWFText				
SWFTextField				
mnogosearch agent				
mnogosearch result				
msql link	msql_connect()	msql(), msql_create_db(), msql_createdb(), msql_drop_db(), msql_drop_db(), msql_select_db(), msql_select_db()	msql_close()	Link to mSQL database
msql link persistent	msql_pconnect()	msql(), msql_create_db(), msql_createdb(), msql_drop_db(), msql_drop_db(), msql_select_db(), msql_select_db()	None	Persistent link to mSQL
msql query	msql_query()	msql(), msql_affected_rows(), msql_data_seek(), msql_dbname(), msql_fetch_array(), msql_fetch_field(), msql_fetch_object(), msql_fetch_row(), msql_fieldname(), msql_field_seek(), msql_fieldtable(), msql_fieldtype(), msql_fieldflags(), msql_fieldlen(), msql_num_fields(), msql_num_rows(), msql_numfields(), msql_numrows(), msql_result()	msql_free_result(), msql_free_result()	mSQL result
mssql link	mssql_connect()	mssql_query(), mssql_select_db()	mssql_close()	Link to Microsft SQL Server database
mssql link persistent	mssql_pconnect()	mssql_query(), mssql_select_db()	None	Persistent link to Microsft SQL Server
mssql result	mssql_query()	mssql_data_seek(), mssql_fetch_array(), mssql_fetch_field(), mssql_fetch_object(), mssql_fetch_row(), mssql_field_length(), mssql_field_name(), mssql_field_seek(), mssql_field_type(), mssql_num_fields(), mssql_num_rows(), mssql_result()	mssql_free_result()	Microsft SQL Server result
mysql link	mysql_connect()	mysql_affected_rows(), mysql_change_user(), mysql_create_db(), mysql_data_seek(), mysql_db_name(), mysql_db_query(), mysql_drop_db(), mysql_errno(), mysql_error(), mysql_insert_id(), mysql_list_dbs(), mysql_list_fields(), mysql_list_tables(), mysql_query(), mysql_result(), mysql_select_db(), mysql_tablename(), mysql_get_host_info(), mysql_get_proto_info(), mysql_get_server_info()	mysql_close()	Link to MySQL database

Resource Type Name	Created By	Used By	Destroyed By	Definition
mysql link persistent	mysql_pconnect()	mysql_affected_rows(), mysql_change_user(), mysql_create_db(), mysql_data_seek(), mysql_db_name(), mysql_db_query(), mysql_drop_db(), mysql_errno(), mysql_error(), mysql_insert_id(), mysql_list_dbs(), mysql_list_fields(), mysql_list_tables(), mysql_query(), mysql_result(), mysql_select_db(), mysql_tablename(), mysql_get_host_info(), mysql_get_proto_info(), mysql_get_server_info()	None	Persistent link to MySQL database
mysql result	mysql_db_query(), mysql_list_dbs(), mysql_list_fields(), mysql_list_tables(), mysql_query()	mysql_data_seek(), mysql_db_name(), mysql_fetch_array(), mysql_fetch_assoc(), mysql_fetch_field(), mysql_fetch_lengths(), mysql_fetch_object(), mysql_fetch_row(), mysql_fetch_row(), mysql_field_flags(), mysql_field_name(), mysql_field_len(), mysql_field_seek(), mysql_field_table(), mysql_field_type(), mysql_num_fields(), mysql_num_rows(), mysql_result(), mysql_tablename()	mysql_free_result()	MySQL result
oci8 collection				
oci8 connection	ocilogon(), ociplogon(), ocinlogon()	ocicommit(), ociserverversion(), ocinewcursor(), ociparse(), ocierror()	ocilogoff()	Link to Oracle database
oci8 descriptor				
oci8 server				
oci8 session				
oci8 statement	ocinewdescriptor()	ocirollback(), ocinewdescriptor(), ocirowcount(), ocidefinebyname(), ocibindbyname(), ociexecute(), ocinumcols(), ociresult(), ocifetch(), ocifetchinto(), ocifetchstatement(), ocicolumnisnull(), ocicolumnname(), ocicolumnsize(), ocicolumntype(), ocistatementtype(), ocierror()	ocifreestatement()	Oracle Cursor
odbc link	odbc_connect()	odbc_autocommit(), odbc_commit(), odbc_error(), odbc_errormsg(), odbc_exec(), odbc_tables(), odbc_tableprivileges(), odbc_do(), odbc_prepare(), odbc_columns(), odbc_columnprivileges(), odbc_procedurecolumns(), odbc_specialcolumns(), odbc_rollback(), odbc_setoption(), odbc_gettypeinfo(), odbc_primarykeys(), odbc_foreignkeys(), odbc_procedures(), odbc_statistics()	odbc_close()	Link to ODBC database
odbc link persistent	odbc_connect()	odbc_autocommit(), odbc_commit(), odbc_error(), odbc_errormsg(), odbc_exec(), odbc_tables(), odbc_tableprivileges(), odbc_do(), odbc_prepare(), odbc_columns(), odbc_columnprivileges(), odbc_procedurecolumns(), odbc_specialcolumns(), odbc_rollback(), odbc_setoption(), odbc_gettypeinfo(), odbc_primarykeys(), odbc_foreignkeys(), odbc_procedures(), odbc_statistics()	None	Persistent link to ODBC database
odbc result	odbc_prepare()	odbc_binmode(), odbc_cursor(), odbc_execute(), odbc_fetch_into(), odbc_fetch_row(), odbc_field_name(), odbc_field_num(), odbc_field_type(), odbc_field_len(), odbc_field_precision(),	odbc_free_result()	ODBC result

Resource Type Name	Created By	Used By	Destroyed By	Definition
		odbc_field_scale(), odbc_longreadlen(), odbc_num_fields(), odbc_num_rows(), odbc_result(), odbc_result_all(), odbc_setoption()		
birdstep link				
birdstep result				
OpenSSL key	openssl_get_privatekey(), openssl_get_publickey()	openssl_sign(), openssl_seal(), openssl_open(), openssl_verify()	openssl_free_key()	OpenSSL key
OpenSSL X.509	openssl_x509_read()	openssl_x509_parse(), openssl_x509_checkpurpose()	openssl_x509_free()	Public Key
oracle Cursor	ora_open()	ora_bind(), ora_columnname(), ora_columnsize(), ora_columntype(), ora_error(), ora_errorcode(), ora_exec(), ora_fetch(), ora_fetch_into(), ora_getcolumn(), ora_numcols(), ora_numrows(), ora_parse()	ora_close()	Oracle cursor
oracle link	ora_logon()	ora_do(), ora_error(), ora_errorcode(), ora_rollback(), ora_commitoff(), ora_commiton(), ora_open(), ora_commit()	ora_logoff()	Link to oracle database
oracle link persistent	ora_plogon()	ora_do(), ora_error(), ora_errorcode(), ora_rollback(), ora_commitoff(), ora_commiton(), ora_open(), ora_commit()	None	Persistent link to oracle database
pdf document	pdf_new()	pdf_add_bookmark(), pdf_add_launchlink(), pdf_add_locallink(), pdf_add_note(), pdf_add_pdflink(), pdf_add_weblink(), pdf_arc(), pdf_attach_file(), pdf_begin_page(), pdf_circle(), pdf_clip(), pdf_closepath(), pdf_closepath_fill_stroke(), pdf_closepath_stroke(), pdf_concat(), pdf_continue_text(), pdf_curveto(), pdf_end_page(), pdf_endpath(), pdf_fill(), pdf_fill_stroke(), pdf_findfont(), pdf_get_buffer(), pdf_get_image_height(), pdf_get_image_width(), pdf_get_parameter(), pdf_get_value(), pdf_lineto(), pdf_moveto(), pdf_open_ccitt(), pdf_open_file(), pdf_open_image_file(), pdf_place_image(), pdf_rect(), pdf_restore(), pdf_rotate(), pdf_save(), pdf_scale(), pdf_setdash(), pdf_setflat(), pdf_setfont(), pdf_setgray(), pdf_setgray_fill(), pdf_setgray_stroke(), pdf_setlinecap(), pdf_setlinejoin(), pdf_setlinewidth(), pdf_setmiterlimit(), pdf_setpolydash(), pdf_setrgbcolor(), pdf_setrgbcolor_fill(), pdf_setrgbcolor_stroke(), pdf_set_border_color(), pdf_set_border_dash(), pdf_set_border_style(), pdf_set_char_spacing(), pdf_set_duration(), pdf_set_font(), pdf_set_horiz_scaling(), pdf_set_parameter(), pdf_set_text_pos(), pdf_set_text_rendering(), pdf_set_value(), pdf_set_word_spacing(), pdf_show(), pdf_show_boxed(), pdf_show_xy(), pdf_skew(), pdf_stringwidth(), pdf_stroke(), pdf_translate(),	pdf_close(), pdf_delete()	PDF document

Resource Type Name	Created By	Used By	Destroyed By	Definition
		pdf_open_memory_image()		
pdf image	pdf_open_image(), pdf_open_image_file(), pdf_open_memory_image()	pdf_get_image_height(), pdf_get_image_width(), pdf_open_CCITT(), pdf_place_image()	pdf_close_image()	Image in PDF file
pdf object				
pdf outline				
pgsql large object	pg_lo_open()	pg_lo_open(), pg_lo_create(), pg_lo_read(), pg_lo_read_all(), pg_lo_seek(), pg_lo_tell(), pg_lo_unlink(), pg_lo_write()	pg_lo_close()	PostgreSQL Large Object
pgsql link	pg_connect()	pg_affected_rows(), pg_query(), pg_send_query(), pg_get_result(), pg_connection_busy(), pg_connection_reset(), pg_connection_status(), pg_last_error(), pg_last_notice(), pg_lo_create(), pg_lo_export(), pg_lo_import(), pg_lo_open(), pg_lo_unlink(), pg_host(), pg_port(), pg_dbname(), pg_options(), pg_copy_from(), pg_copy_to(), pg_end_copy(), pg_put_line(), pg_tty(), pg_trace(), pg_untrace(), pg_set_client_encoding(), pg_client_encoding(), pg_metadata(), pg_convert(), pg_insert(), pg_select(), pg_delete(), pg_update()	pg_close()	Link to PostgreSQL database
pgsql link persistent	pg_pconnect()	pg_affected_rows(), pg_query(), pg_send_query(), pg_get_result(), pg_connection_busy(), pg_connection_reset(), pg_connection_status(), pg_last_error(), pg_last_notice(), pg_lo_create(), pg_lo_export(), pg_lo_import(), pg_lo_open(), pg_lo_unlink(), pg_host(), pg_port(), pg_dbname(), pg_options(), pg_copy_from(), pg_copy_to(), pg_end_copy(), pg_put_line(), pg_tty(), pg_trace(), pg_untrace(), pg_set_client_encoding(), pg_client_encoding(), pg_metadata(), pg_convert(), pg_insert(), pg_select(), pg_delete(), pg_update()	None	Persistent link to PostgreSQL database
pgsql result	pg_query(), pg_get_result()	pg_fetch_array(), pg_fetch_object(), pg_fetch_result(), pg_fetch_row(), pg_field_is_null(), pg_field_name(), pg_field_num(), pg_field_prtlen(), pg_field_size(), pg_field_type(), pg_last_oid(), pg_num_fields(), pg_num_rows(), pg_result_error(), pg_result_status()	pg_free_result()	PostgreSQL result
pgsql string				
printer				
printer brush				
printer font				
printer pen				
pspell	pspell_new(), pspell_new_config(), pspell_new_personal()	pspell_add_to_personal(), pspell_add_to_session(), pspell_check(), pspell_clear_session(), pspell_config_ignore(), pspell_config_mode(), pspell_config_personal(), pspell_config_repl(), pspell_config_runtogether(),	None	pspell dictionary

Resource Type Name	Created By	Used By	Destroyed By	Definition
		pspell_config_save_repl(), pspell_save_wordlist(), pspell_store_replacement(), pspell_suggest()		
pspell config	pspell_config_create()	pspell_new_config()	None	pspell configuration
Sablotron XSLT	xslt_create()	xslt_closelog(), xslt_openlog(), xslt_run(), xslt_set_sax_handler(), xslt_errno(), xslt_error(), xslt_fetch_result(), xslt_free()	xslt_free()	XSLT parser
shmop	shmop_open()	shmop_read(), shmop_write(), shmop_size(), shmop_delete()	shmop_close()	
sockets file descriptor set	socket()	accept_connect(), bind(), connect(), listen(), read(), write()	close()	Socket
sockets i/o vector				
dir	dir()	readdir(), rewinddir()	closedir()	Dir handle
file	fopen()	feof(), fflush(), fgetc(), fgetcsv(), fgets(), fgetss(), flock(), fpassthru(), fputs(), fwrite(), fread(), fseek(), ftell(), fstat(), ftruncate(), set_file_buffer(), rewind()	fclose()	File handle
pipe	popen()	feof(), fflush(), fgetc(), fgetcsv(), fgets(), fgetss(), fpassthru(), fputs(), fwrite(), fread()	pclose()	Process handle
socket	fsockopen()	fflush(), fgetc(), fgetcsv(), fgets(), fgetss(), fpassthru(), fputs(), fwrite(), fread()	fclose()	Socket handle
stream				
sybase-db link	sybase_connect()	sybase_query(), sybase_select_db()	sybase_close()	Link to Sybase database using DB library
sybase-db link persistent	sybase_pconnect()	sybase_query(), sybase_select_db()	None	Persistent link to Sybase database using DB library
sybase-db result	sybase_query()	sybase_data_seek(), sybase_fetch_array(), sybase_fetch_field(), sybase_fetch_object(), sybase_fetch_row(), sybase_field_seek(), sybase_num_fields(), sybase_num_rows(), sybase_result()	sybase_free_result()	Sybase result using DB library
sybase-ct link	sybase_connect()	sybase_affected_rows(), sybase_query(), sybase_select_db()	sybase_close()	Link to Sybase database using CT library
sybase-ct link persistent	sybase_pconnect()	sybase_affected_rows(), sybase_query(), sybase_select_db()	None	Persistent link to Sybase database using CT library
sybase-ct result	sybase_query()	sybase_data_seek(), sybase_fetch_array(), sybase_fetch_field(), sybase_fetch_object(), sybase_fetch_row(), sybase_field_seek(), sybase_num_fields(), sybase_num_rows(), sybase_result()	sybase_free_result()	Sybase result using CT library
sysvsem	sem_get()	sem_acquire()	sem_release()	System V Semaphore
sysvshm	shm_attach()	shm_remove(), shm_put_var(), shm_get_var(), shm_remove_var()	shm_detach()	System V Shared Memory
wddx	wddx_packet_start()	wddx_add_vars()	wddx_packet_end()	WDDX packet

Resource Type Name	Created By	Used By	Destroyed By	Definition
xml	xml_parser_create()	xml_set_object(), xml_set_element_handler(), xml_set_character_data_handler(), xml_set_processing_instruction_handler(), xml_set_default_handler(), xml_set_unparsed_entity_decl_handler(), xml_set_notation_decl_handler(), xml_set_external_entity_ref_handler(), xml_parse(), xml_get_error_code(), xml_error_string(), xml_get_current_line_number(), xml_get_current_column_number(), xml_get_current_byte_index(), xml_parse_into_struct(), xml_parser_set_option(), xml_parser_get_option()	xml_parser_free()	XML parser
zlib	gzopen()	gzeof(), gzgetc(), gzgets(), gzgetss(), gzpassthru(), gzputs(), gzread(), gzrewind(), gzseek(), gztell(), gzwrite()	gzclose()	gz-compressed file

Appendix I. List of Supported Protocols/Wrappers

The following is a list of the various URL style protocols that PHP has built-in for use with the filesystem functions such as fopen() and copy(). In addition to these wrappers, as of PHP 4.3, you can write your own wrappers using PHP script and stream_register_wrapper().

HTTP and HTTPS

PHP 3, PHP 4. `https://` since PHP 4.3

- `http://example.com`
- `http://user:password@example.com`
- `https://example.com`
- `https://user:password@example.com`

Allows read-only access to files/resources via HTTP 1.0, using the HTTP GET method. A `Host:` header is sent with the request to handle name-based virtual hosts. If you have configured a user_agent string using your ini file or the stream context, it will also be included in the request.

Redirects have been supported since PHP 4.0.5; if you are using an earlier version you will need to include trailing slashes in your URLs.

The stream allows access to the *body* of the resource; the headers are stored in the `$http_response_header` variable. Since PHP 4.3, the headers are available using stream_get_meta_data().

HTTP connections are read-only; you cannot write data or copy files to an HTTP resource.

> **Note:** HTTPS is supported starting from PHP 4.3, if you have compiled in support for OpenSSL.

FTP and FTPS

PHP 3, PHP 4. `ftps://` since PHP 4.3

- `ftp://example.com/pub/file.txt`
- `ftp://user:password@example.com/pub/file.txt`
- `ftps://example.com/pub/file.txt`

- `ftps://user:password@example.com/pub/file.txt`

Allows read access to existing files and creation of new files via FTP. If the server does not support passive mode ftp, the connection will fail.

You can open files for either reading or writing, but not both simultaneously. If the remote file already exists on the ftp server and you attempt to open it for writing, the connection will fail. If you need to update existing files over ftp, use **ftp_connect()**.

`ftps://` was introduced in PHP 4.3. It is the same as `ftp://`, but attempts to negotiate a secure connection with the ftp server. If the server does not support SSL, then the connection falls back to regular unencrypted ftp.

> **Note:** FTPS is supported starting from PHP 4.3, if you have compiled in support for OpenSSL.

PHP input/output streams

PHP 3.0.13 and up, `php://output` and `php://input` since PHP 4.3

- `php://stdin`
- `php://stdout`
- `php://stderr`
- `php://output`
- `php://input`

`php://stdin`, `php://stdout` and `php://stderr` allow access to the corresponding input or output stream of the PHP process.

`php://output` allows you to write to the output buffer mechanism in the same way as **print()** and **echo()**.

`php://input` allows you to read raw POST data. It is a less memory intensive alternative to `$HTTP_RAW_POST_DATA` and does not need any special `php.ini` directives.

`php://stdin` and `php://input` are read-only, whereas `php://stdout`, `php://stderr` and `php://output` are write-only.

Compression Streams

`zlib:` PHP 4.0.4 - PHP 4.2.3 (systems with fopencookie only)

`compress.zlib://` and `compress.bzip2://` PHP 4.3 and up

- `zlib:`
- `compress.zlib://`
- `compress.bzip2://`

`zlib:` works like **gzopen()**, except that the stream can be used with **fread()** and the other filesystem functions. This is deprecated as of PHP 4.3 due to ambiguities with filenames containing ':' characters; use `compress.zlib://` instead.

`compress.zlib://` and `compress.bzip2://` are equivalent to **gzopen()** and **bzopen()** respectively, and operate even on systems that do not support fopencookie.

Appendix J. List of Parser Tokens

Various parts of the PHP language are represented internally by types like T_SR. PHP outputs identifiers like this one in parse errors, like `"Parse error: unexpected T_SR, expecting ',' or ';' in script.php on line 10."`

You're supposed to know what T_SR means. For everybody who doesn't know that, here is a table with those identifiers, PHP-syntax and references to the appropriate places in the manual.

Table J-1. Tokens

Token	Syntax	Reference
T_AND_EQUAL	&=	assignment operators

Token	Syntax	Reference
T_ARRAY	array()	**array()**, array syntax
T_ARRAY_CAST	(array)	type-casting
T_AS	as	foreach
T_BAD_CHARACTER		anything below ASCII 32 except \t (0x09), \n (0x0a) and \r (0x0d)
T_BOOLEAN_AND	&&	logical operators
T_BOOLEAN_OR	||	logical operators
T_BOOL_CAST	(bool) or (boolean)	type-casting
T_BREAK	break	break
T_CASE	case	switch
T_CHARACTER		
T_CLASS	class	classes and objects
T_CLOSE_TAG	?> or %>	
T_COMMENT	// or #	comments
T_CONCAT_EQUAL	.=	assignment operators
T_CONST	const	
T_CONSTANT_ENCAPSED_STRING	"foo" or 'bar'	string syntax
T_CONTINUE	continue	
T_CURLY_OPEN		
T_DEC	--	incrementing/decrementing operators
T_DECLARE	declare	declare
T_DEFAULT	default	switch
T_DIV_EQUAL	/=	assignment operators
T_DNUMBER	0.12, etc	floating point numbers
T_DO	do	do..while
T_DOLLAR_OPEN_CURLY_BRACES	${	complex variable parsed syntax
T_DOUBLE_ARROW	=>	array syntax
T_DOUBLE_CAST	(real), (double) or (float)	type-casting
T_ECHO	echo	**echo()**
T_ELSE	else	else
T_ELSEIF	elseif	elseif
T_EMPTY	empty	**empty()**
T_ENCAPSED_AND_WHITESPACE		
T_ENDDECLARE	enddeclare	declare, alternative syntax
T_ENDFOR	endfor	for, alternative syntax
T_ENDFOREACH	endforeach	foreach, alternative syntax
T_ENDIF	endif	if, alternative syntax
T_ENDSWITCH	endswitch	switch, alternative syntax
T_ENDWHILE	endwhile	while, alternative syntax
T_END_HEREDOC		heredoc syntax
T_EVAL	eval()	**eval()**
T_EXIT	exit or die	**exit()**, **die()**
T_EXTENDS	extends	extends, classes and objects
T_FILE	__FILE__	constants
T_FOR	for	for
T_FOREACH	foreach	foreach
T_FUNCTION	function or cfunction	functions
T_GLOBAL	global	variable scope
T_IF	if	if
T_INC	++	incrementing/decrementing operators
T_INCLUDE	include()	**include()**
T_INCLUDE_ONCE	include_once()	**include_once()**

Token	Syntax	Reference
T_INLINE_HTML		
T_INT_CAST	(int) or (integer)	type-casting
T_ISSET	isset()	**isset()**
T_IS_EQUAL	==	comparison operators
T_IS_GREATER_OR_EQUAL	>=	comparison operators
T_IS_IDENTICAL	===	comparison operators
T_IS_NOT_EQUAL	!= or <>	comparison operators
T_IS_NOT_IDENTICAL	!==	comparison operators
T_SMALLER_OR_EQUAL	<=	comparison operators
T_LINE	__LINE__	constants
T_LIST	list()	**list()**
T_LNUMBER	123, 012, 0x1ac, etc	integers
T_LOGICAL_AND	and	logical operators
T_LOGICAL_OR	or	logical operators
T_LOGICAL_XOR	xor	logical operators
T_MINUS_EQUAL	-=	assignment operators
T_ML_COMMENT	/* and */	comments
T_MOD_EQUAL	%=	assignment operators
T_MUL_EQUAL	*=	assignment operators
T_NEW	new	classes and objects
T_NUM_STRING		
T_OBJECT_CAST	(object)	type-casting
T_OBJECT_OPERATOR	->	classes and objects
T_OLD_FUNCTION	old_function	old_function
T_OPEN_TAG	<?php, <? or <%	escaping from HTML
T_OPEN_TAG_WITH_ECHO	<?= or <%=	escaping from HTML
T_OR_EQUAL	|=	assignment operators
T_PAAMAYIM_NEKUDOTAYIM	::	::
T_PLUS_EQUAL	+=	assignment operators
T_PRINT	print()	**print()**
T_REQUIRE	require()	**require()**
T_REQUIRE_ONCE	require_once()	**require_once()**
T_RETURN	return	returning values
T_SL	<<	bitwise operators
T_SL_EQUAL	<<=	assignment operators
T_SR	>>	bitwise operators
T_SR_EQUAL	>>=	assignment operators
T_START_HEREDOC	<<<	heredoc syntax
T_STATIC	static	variable scope
T_STRING		
T_STRING_CAST	(string)	type-casting
T_STRING_VARNAME		
T_SWITCH	switch	switch
T_UNSET	unset()	**unset()**
T_UNSET_CAST	(unset)	(not documented; casts to NULL)
T_USE	use	(not implemented)
T_VAR	var	classes and objects
T_VARIABLE	$foo	variables
T_WHILE	while	while, do..while
T_WHITESPACE		
T_XOR_EQUAL	^=	assignment operators

Token	Syntax	Reference
T_FUNC_C	__FUNCTION__	constants, since PHP 4.3.0
T_CLASS_C	__CLASS__	constants, since PHP 4.3.0

Appendix K. About the manual

Formats

The PHP manual is provided in several formats. These formats can be divided into two groups: online readable formats, and downloadable packages.

> **Note:** Some publishers have made available printed versions of this manual. We cannot recommend any of those, as they tend to become out-of-date very quickly.

You can read the manual online at http://www.php.net/ and on the numerous mirror sites. For best performance, you should choose the mirror site closest to you. You can view the manual in either its plain (print-friendly) HTML format or an HTML format that integrates the manual into the look and feel of the PHP website itself.

An advantage of the online manual over most of the offline formats is the integration of user-contributed notes. An obvious disadvantage is that you have to be online to view the manual in the online formats.

There are several offline formats of the manual, and the most appropriate format for you depends on what operating system you use and your personal reading style. For information on how the manual is generated in so many formats, read the 'How we generate the formats' section of this appendix.

The most cross-platform formats of the manual are the HTML and plain-text versions. The HTML format is provided both as a single HTML file and as a package of individual files for each section (which results in a collection of several thousand files). The HTML and plaintext formats are provided as tar files compressed using the bzip2 archiver.

Another popular cross-platform format, and the format most suited to printing, is PDF (also known as Adobe Acrobat). But before you rush to download this format and hit the Print button, be warned that the manual is nearly 2000 pages long, and constantly being revised.

> **Note:** If you do not already have a program capable of viewing PDF format files, you may need to download Adobe Acrobat Reader.

For owners of Palm-compatible handhelds, the Palm document and iSilo formats are ideal for this platform. You can bring your handheld with you on your daily commute and use a DOC or iSilo reader to brush up on your PHP knowledge, or just use it as a quick reference.

For Windows platforms, the Windows HTML Help version of the manual soups up the HTML format for use with the Windows HTML Help application. This version provides full-text search, a full index, and bookmarking. Many popular Windows PHP development environments also integrate with this version of the documentation to provide easy access.

About user notes

The user-contributed notes play an important role in the development of this manual. By allowing readers of the manual to contribute examples, caveats, and further clarifications from their browser, we are able to incorporate that feedback into the main text of the manual. And until the notes have been incorporated, they can be viewed in their submitted form online and in some of the offline formats.

> **Note:** The user-contributed notes are not moderated before they appear online, so the quality of the writing or code examples, and even the veracity of the contribution, cannot be guaranteed. (Not that there is any guarantee of the quality or accuracy of the manual text itself.)
>
> **Note:** For the purposes of license coverage the user-contributed notes are considered part of the PHP manual, and are therefore covered by the same license that covers this documentation (GPL at the moment). For more details see the Manual's Copyright page.

How to read a function definition (prototype)

Each function is documented for quick reference, knowing how to read and understand the manual will make using PHP much

easier. Rather than relying on examples or cut/paste, you want to know how to read function definitions (prototypes). Let's begin:

> **Prerequisite: Basic understanding of types:** Although PHP is a loosly typed language, it's important to have a basic understanding of types as they have important meaning.

Function definitions tell us what type of value is returned, let's use the definition for **strlen()** as our first example:

```
strlen

(PHP 3, PHP 4 >= 4.0.0)
strlen -- Get string length

Description
int strlen ( string str )

Returns the length of string.
```

Table K-1. Explanation of a function definition

Part	Description
strlen	The function name.
(PHP 3, PHP 4 >= 4.0.0)	strlen() has been around in both all of PHP 3 and PHP 4
int	Type of value this function returns, which is an integer (i.e. The length of a string is measured in numbers).
(string str)	The first (and in this case the only) parameter/argument for the function strlen() is named *str*, and it's a string.

We could rewrite the above function definition in a generic way:

```
returned type    function name    ( parameter type   parameter name )
```

Many functions take on multiple parameters, such as **in_array()**. It's prototype is as follows:

```
bool in_array ( mixed needle, array haystack [, bool strict])
```

What does this mean? in_array() returns a boolean value, **TRUE** on success (the *needle* was found in the *haystack*) or **FALSE** on failure (the *needle* was not found in the *haystack*). The first parameter is named *needle* and it can be many different types, so we call it "*mixed*". This mixed *needle* (what we're looking for) can either be a scalar value (string, integer, or float), or an array. *haystack* (the array we're searching in) is the second parameter. The third *optional* parameter is named *strict*. All optional parameters are seen in *[* brackets *]*. The manual states that the *strict* parameter defaults to boolean **FALSE**. See the manual page on each function for details on how they work.

PHP versions documented in this manual

This documentation contains information about PHP 4, with some added migration and compatibility notes regarding PHP 3. Behaviour, parameter, return value and other changes between different PHP versions are documented in notes and inline text in the manual.

You may find documentation pieces for the CVS version of PHP, which always means the very latest development version available through the CVS version handling system. If you are not a developer of PHP itself, and you are not keen on using the very latest development version of PHP, features marked with the "available in CVS" wording are not accessible to you. Though these features will probably be available in the next stable version of PHP. If you would like to download the CVS version, see the anonymous CVS access page.

You may also encounter documentation for a PHP version which is not released (something like PHP 5.0.0 as the latest stable version is 4.3.0). Most of the time, this is not an error in the documentation. Explanation is often added for features not available in the current PHP release, but will be available in a known future PHP version.

How to find more information about PHP

This manual does not attempt to provide instruction about general programming practices. If you are a first-time, or even just a beginning, programmer, you may find it difficult to learn how to program in PHP using just this manual. You may want to seek out a text more oriented towards beginners. You can find a list of PHP-related books at http://www.php.net/books.php.

There are a number of active mailing lists for discussion of all aspects of programming with PHP. If you find yourself stuck on a problem for which you can't find your own solution, you may be able to get help from someone on these lists. You can find a list of the mailing lists at http://www.php.net/support.php, as well as links to the mailing list archives and other online support resources. Furthermore, at http://www.php.net/links.php there is a list of websites devoted to PHP articles, forums, and code galleries.

How to help improve the documentation

There are three ways you can help to improve this documentation.

If you find errors in this manual, in any language, please report them using the bug system at http://bugs.php.net/. Classify the bug as "Documentation Problem". You can also submit problems related to specific manual formats here.

> **Note:** Please don't abuse the bug system by submitting requests for help. Use the mailing lists or community sites mentioned earlier, instead.

By contributing notes, you can provide additional examples, caveats, and clarifications for other readers. But do not submit bug reports using the annotation system please. You can read more about annotations in the 'About user notes' section of this appendix.

If you know English and some foreign language, you may also help out in the translations. If you would like to start a new translation, or help in a translation project, please read http://cvs.php.net/co.php/phpdoc/howto/howto.html.tar.gz.

How we generate the formats

This manual is written in XML using the DocBook XML DTD, using DSSSL (Document Style and Semantics Specification Language) for formatting, and experimentally the XSLT (Extensible Stylesheet Language Transformations) for maintenance and formatting.

Using XML as a source format gives us the ability to generate many output formats from the source files, while only maintaining one source document for all formats. The tools used for formatting HTML and TeX versions are Jade, written by James Clark and The Modular DocBook Stylesheets written by Norman Walsh. We use Microsoft HTML Help Workshop to generate the Windows HTML Help format of the manual, and of course PHP itself to do some additional conversions and formatting.

You can download the manual in various languages and formats, including plain text, plain HTML, PDF, PalmPilot DOC, PalmPilot iSilo and Windows HTML Help, from http://www.php.net/docs.php. The manuals are updated automatically as the text is updated.

You can find more information about downloading the XML source code of this documentation at http://cvs.php.net/. The documentation is stored in the `phpdoc` module.

Translations

The PHP manual is not only available in various formats, it is also available in various languages. The text of the manual is first written in english, then teams of people across the world take care of translating it to their native language. If a translation for a specified function or chapter has not yet been made, the build system of the manual falls back to the english version of it.

Peoples involved in the translations start from the XML source code available from http://cvs.php.net/ and from it they translate to thier mother language. They do *not use* the HTML, the plain text, or the PDF version. It is the build system which takes care of the conversions from XML to human readable formats.

> **Note:** If you would like to help translating the documentation to your native language, please get in touch with the translation/documentation team subscribing to the phpdoc mailinglist: send an empty mail to phpdoc-subscribe@lists.php.net. The mailing list address is `phpdoc@lists.php.net`. State in the message that you are interested in translating the manual to a language and someone will get back to you, helping you start a new language translation or reach the already active documentation team for your language.

At the moment the manual is available, partly or not, in the following languages: Brazilian Portuguese, Chinese, Czech, Dutch, French, German, Hungarian, Italian, Japanese, Korean, Polish and Spanish.

They all can be downloaded here: http://www.php.net/docs.php.

Appendix L. Missing Stuff