



PHP Handbuch

<http://kickme.to/tiger/>

PHP Handbuch

Stig Sæther Bakken

Alexander Aulbach

Egon Schmid

Jim Winstead

Lars Torben Wilson

Rasmus Lerdorf

Zeev Suraski

Herausgegeben von

Egon Schmid

[Copyright](#) © 1997, 1998, 1999, 2000 von der PHP Dokumentations Gruppe

Inhaltsverzeichnis

[Vorwort](#)

[Über dieses Handbuch](#)

I [Einführung](#)

- 1 [Einleitung](#)
- 2 [Installation](#)
- 3 [Configuration](#)
- 4 [Sicherheit](#)

II [Sprachreferenz](#)

- 5 [Grundlagen der Syntax](#)
- 6 [Typen](#)
- 7 [Variablen](#)
- 8 [Konstanten](#)
- 9 [Ausdrücke](#)
- 10 [Operatoren](#)
- 11 [Kontroll-Strukturen](#)
- 12 [Funktionen](#)
- 13 [Klassen und Objekte](#)

III [Features](#)

- 14 [Fehlerbehandlung](#)

- 15 [Creating GIF images](#)
- 16 [HTTP-Authentifizierung mit PHP](#)
- 17 [Cookies](#)
- 18 [Steuerung von Dateiuploads](#)
- 19 [Using remote files](#)
- 20 [Verbindungssteuerung](#)
- 21 [Persistente Datenbankverbindungen](#)

IV [Funktionsreferenz](#)

- I. [Apache-spezifische Funktionen](#)
- II. [Array Funktionen](#)
- III. [Aspell Funktionen](#)
- IV. [Mathematische Funktionen mit beliebiger Genauigkeit](#)
- V. [Kalender-Funktionen](#)
- VI. [COM support functions for Windows](#)
- VII. [Klassen- und Objekt Funktionen](#)
- VIII. [ClibPDF functions](#)
- IX. [Cybercash payment functions](#)
- X. [Database \(dbm-style\) abstraction layer functions](#)
- XI. [Datums- und Zeit-Funktionen](#)
- XII. [dBase Funktionen](#)
- XIII. [dbm functions](#)
- XIV. [Verzeichnis-Funktionen](#)
- XV. [Dynamisch geladene Bibliothek](#)
- XVI. [DOM XML functions](#)
- XVII. [filePro functions](#)
- XVIII. [Funktionen des Dateisystems](#)
- XIX. [Forms Data Format functions](#)
- XX. [FTP-Funktionen](#)
- XXI. [GNU Gettext](#)
- XXII. [HTTP-Funktionen](#)
- XXIII. [Hyperwave functions](#)
- XXIV. [ICAP Functions](#)
- XXV. [Grafik-Funktionen](#)

- XXVI. [IMAP, POP3 und NNTP Funktionen](#)
- XXVII. [Informix functions](#)
- XXVIII. [InterBase-Funktionen](#)
- XXIX. [LDAP functions](#)
- XXX. [Mail Funktionen](#)
- XXXI. [Mathematische Funktionen](#)
- XXXII. [MCAL functions](#)
- XXXIII. [Encryption functions](#)
- XXXIV. [Hash functions](#)
- XXXV. [Microsoft SQL Server functions](#)
- XXXVI. [Sonstige Funktionen](#)
- XXXVII. [mSQL functions](#)
- XXXVIII. [MySQL Funktionen](#)
- XXXIX. [Netzwerk Funktionen](#)
- XL. [ODBC Funktionen](#)
- XLI. [Oracle functions](#)
- XLII. [Oracle 8 functions](#)
- XLIII. [PDF Funktionen](#)
- XLIV. [Payflow Pro functions](#)
- XLV. [PHP Optionen und Informationen](#)
- XLVI. [POSIX functions](#)
- XLVII. [PostgreSQL Funktionen](#)
- XLVIII. [Program Execution functions](#)
- XLIX. [Pspell Functions](#)
- L. [GNU Recode functions](#)
- LI. [Perl-compatible Regular Expression functions](#)
- LII. [Regular expression functions](#)
- LIII. [Semaphor und Shared Memory Funktionen](#)
- LIV. [Session Funktionen](#)
- LV. [Shockwave Flash functions](#)
- LVI. [SNMP functions](#)
- LVII. [String-Funktionen](#)
- LVIII. [Sybase functions](#)

LIX. [URL functions](#)

LX. [Variablen-Functions](#)

LXI. [Vmailmgr functions](#)

LXII. [WDDX functions](#)

LXIII. [XML parser functions](#)

LXIV. [NIS functions](#)

LXV. [Compression functions](#)

V [Anhang](#)

A [Migrating from PHP/FI 2.0 to PHP 3.0](#)

B [PHP development](#)

C [The PHP Debugger](#)

[Vor](#)

Vorwort

Vorwort

Inhaltsverzeichnis

[Über dieses Handbuch](#)

PHP ist die Abkürzung für "PHP: Hypertext Preprocessor" und ist eine Skriptsprache die sich in HTML einbinden lässt. Viele der syntaktischen Möglichkeiten sind den Programmiersprachen C, Java und Perl entnommen und es wurden auch einige PHP spezifische Features entwickelt. Das Ziel der Sprache ist es, das Schreiben von Programmen zur Erzeugung von dynamisch generierten Seiten zu erleichtern und zu beschleunigen.

Über dieses Handbuch

Dieses Handbuch ist in XML geschrieben und verwendet die [DocBook XML DTD](#) und [DSSSL](#) (Document Style and Semantics Specification Language) für die Formatierung. Die Tools zur Transformation in HTML, TeX und RTF sind [Jade](#) von [James Clark](#) und die [Modularen DocBook Stylesheets](#) von [Norman Walsh](#). Die Programme zur Herstellung des PHP Handbuchs wurden von [Stig Sæther Bakken](#) ausgewählt.

Dieses HTML Handbuch wird jeden Tag neu erzeugt und kann bei <http://snaps.php.net/manual/> bezogen werden.

I Einführung

Inhaltsverzeichnis

- 1 [Einleitung](#)
- 2 [Installation](#)
- 3 [Configuration](#)
- 4 [Sicherheit](#)

Kapitel 1 Einleitung

Inhaltsverzeichnis

[Was ist PHP?](#)

[Was kann PHP?](#)

[Eine kurze Entstehungsgeschichte von PHP](#)

Was ist PHP?

PHP (offiziell: "PHP: Hypertext Preprocessor") ist eine server-seitige, in HTML eingebettete Skriptsprache.

Hört sich einfach an, aber was heißt es genau? Ein Beispiel:

Beispiel 1-1 Ein einleitendes Beispiel

```
<html>
  <head>
    <title>Beispiel</title>
  </head>
  <body>
    <?php echo "Hallo, ich bin ein PHP-Skript!"; ?>
  </body>
</html>
```

Dieser Skript unterscheidet sich von einem CGI-Skript, der in einer Sprache wie Perl oder C geschrieben wurde -- anstatt ein Programm mit vielen Anweisungen zur Ausgabe von HTML zu schreiben, schreibt man einen HTML Code mit einigen, eingebetteten Anweisungen, um etwas zu auszuführen (z.B. um -wie oben- Text auszugeben). Der PHP Code steht zwischen speziellen [Anfangs- und Schlusstags](#), mit denen man in den PHP-Modus und zurück wechseln kann.

Was PHP von client-seitigen Sprache wie Javaskript unterscheidet ist, dass der Code vom Server ausgeführt wird. Sollten sie einen Skript wie den obigen auf ihrem Server ausführen, würde der Besucher nur das Ergebnis dessen empfangen, ohne die Möglichkeit zu haben, herauszufinden, wie der zugrundeliegende Code aussieht. Sie können ihren Webserver auch anweisen, alle ihre HTML-Dateien mit PHP zu parsen, denn dann gibt es wirklich nichts, das dem Benutzer sagt, was sie in petto haben.

Was kann PHP?

Prinzipiell kann PHP alles, was jedes andere CGI Programm kann, also z.B. Formulardaten sammeln, dynamischen Inhalt für Websites generieren oder Cookies senden und empfangen.

Vielleicht die größte und bemerkenswerteste Stärke von PHP, ist seine Unterstützung für eine breite Masse von Datenbanken. Eine datenbank-gestützte Website zu erstellen ist unglaublich einfach. Die folgenden Datenbanken werden z.Zt. unterstützt:

Adabas D	InterBase	Solid
dBase	mSQL	Sybase
Empress	MySQL	Velocis
FilePro	Oracle	Unix dbm
Informix	PostgreSQL	

PHP unterstützt auch die Kommunikation mit anderen Diensten durch Protokolle wie IMAP, SNMP, NNTP, POP3 oder auch HTTP. Sie können auch einfache Netzwerk Sockets benutzen und mit Hilfe anderer Protokolle interagieren.

Eine kurze Entstehungsgeschichte von PHP

PHP wurde irgendwann im Herbst des Jahres 1994 von [Rasmus Lerdorf](#) konzipiert. Frühe, nicht veröffentlichte Versionen wurden auf seiner Homepage genutzt, um die Leser seiner Online-Bewerbung festzustellen. Die erste Version, die von anderen genutzt wurde, war Anfang 1995 verfügbar und wurde unter dem Namen "Personal Home Page Tools" bekannt. Sie bestand aus einem extrem simplifizierten Parser, der ausschließlich einige spezielle Macros verstand, und einigen Werkzeugen die damals häufig auf Homepages genutzt wurden: ein Gästebuch, ein Counter und einige andere. Der Parser wurde Mitte 1995 neu programmiert und in PHP/FI umgenannt. Das FI kam von einem anderen Paket, das Rasmus geschrieben hatte und HTML-Formulardaten interpretierte. Er kombinierte die "Personal Home Page Tools"-Skripts mit dem Formular Interpreter und fügte noch mSQL-Unterstützung hinzu und PHP/FI war geboren. PHP/FI wuchs in mit unglaublicher Geschwindigkeit und immer mehr Leute begannen Code beizusteuern.

Es ist schwierig, harte Fakten zu liefern, aber es wird angenommen, dass PHP/FI Ende 1996 für mindestens 15.000 Websites auf ganzen Welt genutzt wurde. Mitte 1997 ist diese Zahl auf über 50.000 angewachsen. Um die gleiche Zeit gab es auch Veränderungen in der Weiterentwicklung von PHP. Es wandelte sich von Rasmus kleinem Privatprojekt, zu dem eine Handvoll Programmierer beigetragen hatte, zu einem wesentlich besser organisierten Anstrengung eines Teams. Von Zeev Suraski und Andi Gutmans wurde der Parser von Grund auf neu geschrieben und bildete die Basis für PHP Version 3. Eine Menge des Codes der PHP/FI-Werkzeuge wurde auf PHP3 portiert und eine Menge wurde komplett neu erstellt.

Heute (Mitte 1999) wird PHP/FI oder PHP3 mit einigen kommerziellen Produkten wie z.B. C2's StrongHold Webserver und RedHat Linux. Eine vorsichtige Schätzung, basierend auf einer Hochrechnung der Zahlen, die von NetCraft veröffentlicht wurden, sagt aus, dass PHP für 150.000 Websites auf der ganzen Welt genutzt wird. Um das in Relation zu setzen: das sind mehr Sites, als alle die, die auf Netscapes Flaggschiff, dem "Enterprise Server", laufen.

Während das hier geschrieben wird, ist PHP auf dem Weg zur nächsten Generation, welche die mächtige [Zend](#) scripting engine nutzen wird, um höhere Geschwindigkeit zu erreichen, und auch mit anderen Servern als dem z.Zt. unterstützten Apache Server, als natives Modul laufen wird.

Kapitel 2 Installation

Inhaltsverzeichnis

[Download der aktuellsten Version](#)

[Installation auf UNIX Systemen](#)

[Installation auf Windows 95/98/NT Systemen](#)

[Probleme?](#)

Download der aktuellsten Version

Den Quellcode sowie die Binärdistributionen für die verschiedenen Plattformen (inklusive Windows) erhalten Sie unter <http://www.php.net/>.

[Zurück](#)

Eine kurze Entstehungsgeschichte
von PHP

[Anfang](#)

[Hoch](#)

[Vor](#)

Installation auf UNIX Systemen

Installation auf UNIX Systemen

Dieser Abschnitt wird Sie durch die Konfiguration und Installation von PHP führen. Erforderliche Kenntnisse und Software:

- Grundlegende UNIX-Kenntnisse (Sie sollten in der Lage sein mit der make-Anweisung und einem C-Compiler umzugehen)
- Einen ANSI C compiler
- Einen Web Server

"Quick Installation"- Anleitung (Apache Modul Version)

```
1.  gunzip apache_1.3.x.tar.gz
2.  tar xvf apache_1.3.x.tar
3.  gunzip php-3.0.x.tar.gz
4.  tar xvf php-3.0.x.tar
5.  cd apache_1.3.x
6.  ./configure --prefix=/www
7.  cd ../php-3.0.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 --prefix=/www --activate-module=src/modules/php3/libphp3.a
13. make
14. make install
```

Sie können aber auch alternativ zu oben beschriebenen Schritt einfach die httpd binary auf Ihre existierende Binary übertragen. In jedem Fall sollte darauf geachtet werden, den Server vorher herunterzufahren.

```
15. cd ../php-3.0.x
16. cp php3.ini-dist /usr/local/lib/php3.ini
```

Sie können nun die php.ini unter /usr/local/lib/php3.ini editieren, um Ihre gewünschten PHP-Optionen einzustellen. Wenn Sie die php.ini lieber in einem anderen Verzeichnis haben möchten, dann sollten Sie Sie in Schritt 8 --with-config-file-path=/path angeben.

17. Editieren Sie nun Ihre httpd.conf or srm.conf folgendermassen:

```
AddType application/x-httpd-php3 .php3
```

Es bleibt Ihnen völlig freigestellt, welche File-Extension Sie hier benutzen. ".php3" wird von uns lediglich empfohlen.

18. Starten Sie den Apache Web Server. Sie müssen den Server starten und beenden, ein einfacher Reload mittels des HUP oder USR1 Signales ist nicht ausreichend.

Configuration

Es gibt zwei Möglichkeiten PHP zu konfigurieren.

- Sie können das mitgelieferte "Setup"- Skript von PHP benutzen. Dieses Skript wird Ihnen eine handvoll Fragen stellen (ähnlich dem "Install"- Skript von PHP/FI 2.0) und am Ende "configure" ausführen. Um dieses Skript auszuführen, geben Sie **./setup** ein.

Dieses Skript wird zusätzlich eine Datei namens "do-conf" erstellen, welches die Konfigurations-Optionen enthält, die "configure" übermittelt wurden. Um später noch Änderungen in der Konfiguration vorzunehmen, können Sie einfach diese Datei editieren, ohne ein erneutes Setup ausführen zu müssen. Geben Sie hierzu **./do-conf** ein, um PHP mit den neuen Optionen zu konfigurieren.

- Die Alternative ist "configure" manuell auszuführen. Um zu sehen, welche Optionen "configure" bietet, geben Sie **./configure --help** ein.

Details über einige der verschiedenen Konfigurations- Optionen sind weiter unten aufgeführt.

Apache Modul

Um PHP als Modul in den Apache einzukompilieren, antworten Sie bitte "yes" auf die Frage "Build as an Apache module?" (die [--with-apache=DIR](#) option zu konfigurieren) und geben Sie das Stammverzeichnis der Apache Distribution an. Wenn Sie also die Apache Distribution in `/usr/local/www/apache_1.2.4`, entpackt haben, dann ist dieses Ihr Stammverzeichnis. Das Standard-Stammverzeichnis ist `/usr/local/etc/httpd`.

fhttpd Module

Um PHP als fhttpd Modul in den Apache einzukompilieren, antworten Sie "yes" auf die Frage "Build as an fhttpd module?" (die [--with-fhttpd=DIR](#) Option zu konfigurieren) und geben Sie das fhttpd source Stammverzeichnis an. Das Standardverzeichnis ist `/usr/local/src/fhttpd`. Wenn Sie fhttpd benutzen, wird PHP als Modul performanter und besser kontrollierbar und besitzt mehr Fähigkeiten zur Remote-Ausführung.

CGI version

Standardmäßig wird PHP als CGI-Programm installiert. Wenn Sie einen Webserver betreiben, auf dem PHP Module unterstützt werden, dann sollten Sie diese Variante aus Performancegründen den Vorzug geben. Wie auch immer, die CGI-Variante erlaubt es dem Apache Benutzer, verschiedene PHP-unterstützte Seiten unter verschiedenen Benutzer ID's zu ermöglichen. Sie sollten jedoch daran denken, sich mit den Sicherheitseinstellungen im [Security chapter](#) vertraut zu machen, wenn Sie PHP in der CGI - Variante einsetzen wollen.

Database Support Options

PHP unterstützt von Haus aus eine große Anzahl an Datenbanken (wie auch ODBC):

Adabas D

```
--with-adabas=DIR
```

Kompiliert mit Adabas D Unterstützung. Parameter sind das Adabas D Installations-Verzeichnis. Standardmäßig ist dieses auf `/usr/local/adabasd` gesetzt.

[Adabas home page](#)

dBase

```
--with-dbase
```

Schaltet die Unterstützung für DBase ein. Externe Bibliotheken werden nicht benötigt.

filePro

```
--with-filepro
```

Schaltet die Unterstützung für FilePro ein. Nur Lese- Unterstützung. Externe Bibliotheken werden nicht benötigt.

mSQL

```
--with-mysql=DIR
```

Schaltet die Unterstützung für mSQL-Datenbanken ein. Das Parameter für diese Option ist das mSQL-Installations-Verzeichnis, standardmäßig unter `/usr/local/Hughes` . Dieses ist das Standardverzeichnis der mSQL2.0 Distribution. **configure** erkennt automatisch die Version von Ihrem mSQL, PHP unterstützt sowohl Version 1.0 wie auch 2.0 . Wenn Sie PHP mit mSQL 1.0 kompilieren, können Sie aber auch nur auf mSQL 1.0 zugreifen und vica versa.

See also [mSQL Configuration](#) Directives in the [configuration file](#).

[mSQL home page](#)

MySQL

```
--with-mysql=DIR
```

Aktiviert die die MySQL-Unterstützung. Als Parameter kann bei dieser Option das MySQL-Installationsverzeichnis angegeben werden. Standardmäßig ist dieses auf `/usr/local` gesetzt.

Siehe auch: [MySQL Konfiguration](#) Direktiven in der [Konfigurations- Datei](#).

[MySQL home page](#)

iODBC

```
--with-iodbc=DIR
```

Aktiviert die iODBC-Unterstützung. Dieses Merkmal wurde erstmals für den iODBC-Treiber-Manager entwickelt, ein frei verteilter ODBC-Treiber-Manager, der unter vielen UNIX-Derivaten läuft. Als Parameter zu dieser Option kann das iODBC-Installations-Verzeichnis angegeben werden. Standardmäßig ist dieses auf `/usr/local` gesetzt.

[FreeODBChome page](#) or [iODBC home page](#)

OpenLink ODBC

```
--with-openlink=DIR
```

Aktiviert die OpenLink ODBC-Unterstützung. Als Parameter kann bei dieser Option das OpenLink ODBC Installations-Verzeichnis angegeben werden. Standardmäßig ist dieses auf `/usr/local/openlink` gesetzt.

[OpenLink Software's home page](#)

Oracle

```
--with-oracle=DIR
```

Aktiviert die Oracle-Unterstützung. Diese Option wurde getestet, und sollte zumindest mit den Oracle-Versionen 7.0 bis 7.3 laufen. Als Parameter kann das `ORACLE_HOME` Verzeichnis angegeben werden. Sie müssen diesen Parameter aber nicht angeben, wenn die Oracle-Umgebung gesetzt wurde.

[Oracle home page](#)

PostgreSQL

```
--with-pgsql=DIR
```

Aktiviert die PostgreSQL-Unterstützung. Als Parameter kann bei dieser Option das PostgreSQL-Basisinstallationsverzeichnis angegeben werden. Standardmäßig ist dieses auf `/usr/local/pgsql` gesetzt.

Siehe auch: [Postgres Konfiguration](#) Direktiven in der [Konfigurations- Datei](#).

[PostgreSQL home page](#)

Solid

```
--with-solid=DIR
```

Aktiviert Solid-Unterstützung. Als Parameter kann das Solid Installationsverzeichnis angegeben werden. Standardmäßig ist dieses auf `/usr/local/solid` gesetzt.

[Solid home page](#)

Sybase

```
--with-sybase=DIR
```

Aktiviert Sybase-Unterstützung. Als Parameter kann das Sybase Installationsverzeichnis angegeben werden. Standardmäßig ist dieses auf `/home/sybase` gesetzt.

Siehe auch: [Sybase Konfiguration](#) Direktives in der [Konfigurations- Datei](#).

[Sybase home page](#)

Sybase-CT

```
--with-sybase-ct=DIR
```

Aktiviert Sybase-CT Unterstützung. Als Parameter kann das Sybase-CT Installationsverzeichnis angegeben werden. Standardmäßig ist dieses auf `/home/sybase` gesetzt.

See also [Sybase-CT Configuration](#) Directives in the [configuration file](#).

Velocis

```
--with-velocis=DIR
```

Aktiviert Velocis Unterstützung. Als Parameter kann das Velocis Installationsverzeichnis angegeben werden. Standardmäßig ist dieses auf `/usr/local/velocis` gesetzt.

[Velocis home page](#)

Eine benutzerdefinierte ODBC Bibliothek

```
--with-custom-odbc=DIR
```

Aktiviert Unterstützung für eine benutzerdefinierte ODBC-Bibliothek. Als Parameter kann das Basisverzeichnis angegeben werden. Standardmäßig ist dieses auf `/usr/local` gesetzt.

Diese option verlangt, dass Sie zuvor eigene `CUSTOM_ODBC_LIBS` definiert haben, bevor Sie das Konfigurationsskript starten. Ebenfalls müssen Sie einen gültiger `odbc.h` Header irgendwo in in Ihrem Include-Pfad haben. Wenn Sie keinen haben, erstellen Sie ihn und schließen Sie diesen spezifischen Header mit ein. Ihr Header mag auch einige extra- Definitionen verlangen, besonders wenn es sich um eine Mehrbenutzer-Plattform handelt. Definieren Sie diese in `CFLAGS`.

Zum Beispiel können Sie Sybase SQL überall auf QNX wie folgt benutzen: `CFLAGS=-DODBC_QNX`
`LDFLAGS=-lunix CUSTOM_ODBC_LIBS="-ldblib -lodbc" ./configure`
`--with-custom-odbc=/usr/lib/sqlany50`

Unified ODBC

```
--disable-unified-odbc
```

Deaktiviert das "Unified ODBC Modul", bei welchem es sich um ein allgemeines Interface für Datenbanken mit ODBC-Interfaces handelt (z.B. für Solid und Adabas D). Es arbeitet auch mit normalen ODBC-Bibliotheken zusammen. Es wurde getestet mit iODBC, Solid, Adabas D und Sybase SQL. Hierbei ist es erforderlich, dass eines (nur eines) dieser Module oder das Velocis Modul aktiviert sind, oder eine benutzerdefinierte ODBC-Bibliothek spezifiziert ist. Disables the Unified ODBC module, which is a common interface to all the databases with ODBC-based interfaces, such as Solid and Adabas D. It also works for normal ODBC libraries. Has been tested with iODBC, Solid, Adabas D and Sybase SQL Anywhere. Requires that one (and only one) of these modules or the Velocis module is enabled, or a custom ODBC library specified. This option is only applicable if one of the following options is used: [--with-iodbc](#), [--with-solid](#), [--with-adabas](#), [--with-velocis](#), or [--with-custom-odbc](#).

Siehe auch: [Unified ODBC Konfiguration](#) Direktiven in der [Konfigurations Datei](#).

LDAP

```
--with-ldap=DIR
```

Aktiviert die LDAP (Lightweight Directory Access Protocol) Unterstützung. Als Parameter kann das LDAP Standard Installationsverzeichnis angegeben werden, welches standardmäßig auf `/usr/local/ldap` gesetzt ist.

Zusätzliche Informationen zu LDAP sind unter [RFC1777](#) und [RFC1778](#) verfügbar.

Weitere Konfigurations-Optionen

--with-mcrypt=*DIR*

```
--with-mcrypt
```

Aktiviert die Unterstützung für die mcrypt Bibliothek. Siehe auch [mcrypt Dokumentation](#) für weitere Informationen. Wenn Sie das optionale *DIR* Argument benutzen, wird PHP in *DIR*/include nach mcrypt.h suchen.

--enable-sysvsem

```
--enable-sysvsem
```

Aktiviert die Unterstützung für Sys V Semaphores (wird von den meisten UNIX-Derivaten unterstützt). Siehe auch [Semaphore und Shared Memory Dokumentation](#) für weitere Informationen.

--enable-sysvshm

```
--enable-sysvshm
```

Aktiviert die Unterstützung für Sys V Shared Memory (wird von den meisten UNIX-Derivaten unterstützt). Siehe auch [Semaphore und Shared Memory Dokumentation](#) für weitere Informationen.

--with-xml

```
--with-xml
```

Aktiviert die Unterstützung für einen "nicht-validierenden" XML-Parser auf Basis von James Clarks's [expat library](#). Siehe auch [XML Funktions-Referenz](#) für mehr Details.

--enable-maintainer-mode

```
--enable-maintainer-mode
```

Aktiviert Extra- Abhängigkeiten und Compiler Warnungen, welche von einigen PHP Entwicklern benutzt werden.

--with-system-regex

```
--with-system-regex
```

Benutzt mehr die System Bibliothek für Regular Expression als als die gebündelte. Wenn Sie PHP als Servermodul benutzen, müssen Sie die gleiche Bibliothek bei der Installation von PHP wie beim Linking des Servers verwenden. Aktivieren Sie diese Option, wenn die System-Bibliothek Features beinhaltet, die Sie unbedingt benötigen. Es wird jedoch empfohlen, die beiliegende Bibliothek wenn möglich zu benutzen.

--with-config-file-path

```
--with-config-file-path=DIR
```

Mit dieser Option kann das Verzeichnis angegeben werden, in dem PHP nach der PHP.INI ([Die Konfigurations Datei](#)) sucht, wenn PHP gestartet wird.

--with-exec-dir

```
--with-exec-dir=DIR
```

Erlaubt die Ausführung von ausführbaren Dateien in einem Verzeichnis nur, wenn PHP im safe mode betrieben wird. Standardmäßig ist dieses Verzeichnis `/usr/local/bin`. Diese Option setzt den Pfad nur auf den Standardwert. Wenn dieser geändert werden soll, kann dieses mit der [safe_mode_exec_dir](#) Direktive in der [Konfigurations Datei](#) später geändert werden.

--enable-debug

```
--enable-debug
```

Aktiviert extra debug Informationen. Dieses ermöglicht es, genauere Informationen zu erhalten, wenn Probleme mit PHP auftreten. (Bedenken Sie, dass diese Option nichts mit Debugger-Fähigkeiten oder Informationen, die zu PHP-Skripten erhältlich sind, zu tun hat).

--enable-safe-mode

```
--enable-safe-mode
```

Aktiviert "safe mode" standardmäßig. Diese Option zieht einige Restriktionen betreffend den Möglichkeiten von PHP nach sich, so z.B. das Öffnen von Dateien nur im DocumentRoot zu erlauben. Lesen Sie auch [Security chapter](#) für weitergehende Informationen. CGI-Benutzer sollten "safe mode" immer aktivieren. Diese Option setzt nur die standardmäßige Behandlung von PHP. Sie kann später per [safe_mode](#) Direktive in der [Konfigurations Datei](#) geändert werden.

--enable-track-vars

```
--enable-track-vars
```

Aktiviert PHP_TRACK_VARS. Ermöglicht PHP zu verfolgen, von wo GET/POST/COOKIES Variablen in den Arrays HTTP_GET_VARS, HTTP_POST_VARS und HTTP_COOKIE_VARS kommen. Diese Option setzt lediglich den Standardwert, welcher später mit der [track_vars](#) Direktive in der [Konfigurations Datei](#) aktiviert bzw. deaktiviert werden kann.

--enable-magic-quotes

```
--enable-magic-quotes
```

Aktiviert magic quotes. Diese Option setzt nur einen Standardwert. Die Option kann später mit der [magic_quotes_runtime](#) Direktive in der [Konfigurations Datei](#) aktiviert bzw. deaktiviert werden. Siehe auch die [magic_quotes_gpc](#) und die [magic_quotes_sybase](#) Direktiven.

--enable-debugger

```
--enable-debugger
```

Aktiviert die interne PHP Debugger Unterstützung. Dieses Merkmal ist immer noch im Experimentier- Stadium. Siehe auch [Debugger Konfiguration](#) Direktiven in der [Konfigurationsdatei](#).

--enable-discard-path

```
--enable-discard-path
```

Wenn diese Option aktiviert ist, kann die PHP CGI-Binär-Datei sicher ausserhalb des Web-Verzeichnisses plaziert werden, und niemand wird .htaccess-Sicherheit umgehen können. Lesen Sie die [Sektion im Sicherheitskapitel](#) über

diese Option.

--enable-bcmath

```
--enable-bcmath
```

Aktiviert **bc** mathematische Funktionen zur Präzision. Siehe auch [bcmath.scale](#) Option in der [Konfigurationsdatei](#).

--enable-force-cgi-redirect

```
--enable-force-cgi-redirect
```

Aktiviert den Sicherheits-Check für interne Server-Redirects. Sie sollten dieses Option aktivieren, wenn Sie die CGI-Version mit dem Apache benutzen.

Wenn Sie PHP als CGI Binärdatei benutzen, überprüft PHP selber zuerst immer, ob es als Redirect benutzt wird (z.B. unter Apache mittels Action Direktiven). Dieses stellt sicher, dass PHP in dieser Variante nicht missbraucht werden kann, um Authentifizierungsmodelle des Webservers über direkte Aufrufe wie `http://my.host/cgi-bin/php/secret/doc.html` zu umgehen. Dieses Beispiel greift zwar auf `http://my.host/secret/doc.html` zu, aber berücksichtigt keine Sicherheitseinstellungen die durch die Directory-Einstellungen `/secret` in der `httpd` gesetzt wurden.

Wenn Sie diese option nicht aktivieren, erlauben Sie das Umgehen von `httpd` Sicherheits- und Authentifizierungseinstellungen. Tun Sie dieses nur, wenn Ihre Serversoftware nicht fähig ist, zu erkennen, ob ein sicherer Redirect erfolgt ist und alle ihre Dateien unterhalb des Document- Root und Benutzer-Verzeichnisse für jedermann zugänglich sein sollen / können.

Lesen Sie die [Sektion im Sicherheits-Kapitel](#) um mehr über diese option zu erfahren.

--disable-short-tags

```
--disable-short-tags
```

Deaktiviert die Kurzform `<? ?>` PHP Tags. Sie müssen diese Option deaktivieren, wenn Sie PHP im Zusammenhang mit XML benutzen wollen. Wenn Sie die Kurzform deaktiviert haben, ist die einzige gültige Syntax für PHP-Code-Tags `<?php ?>`. Diese Option setzt nur den Standartwert. Die Option kann später in der [short_open_tag](#) Direktive in der [Konfigurationsdatei](#) geändert werden.

--enable-url-includes

```
--enable-url-includes
```

Erlaubt es, Programme direkt von anderen HTTP- oder FTP- Servern mittels [include\(\)](#) laufen zu lassen. Siehe auch

[include_path](#) Option in der [Konfigurationsdatei](#).

--disable-syntax-hl

```
--disable-syntax-hl
```

Deaktiviert Syntax-Highlighting.

CPPFLAGS and LDFLAGS

Um die PHP-Installation dazu zu veranlassen in verschiedenen Verzeichnissen für Header und Bibliotheken zu suchen, modifizieren Sie die CPPFLAGS und LDFLAGS Umgebungsvariablen. Wenn Sie eine sensible shell benutzen, sollte es möglich sein folgendes zu tun: **LDFLAGS=-L/my/lib/dir CPPFLAGS=-I/my/include/dir ./configure**

Building

Wenn PHP fertig konfiguriert ist, können Sie die CGI-ausführbare Datei oder PHP-Bibliothek erstellen. Das Kommando **make** sollte das für Sie erledigen. Wenn dieser Weg fehlschlägt, und Sie nicht herausfinden können warum, schauen Sie hier nach [Problem- Sektion](#).

Testing

Wenn Sie PHP als CGI-Programm hergestellt haben, können Sie das Programm mit **make test** testen. Es ist immer eine gute Idee, Ihr Programm zu testen. So können Sie ein Problem möglicherweise früher erkennen und beseitigen.

Benchmarking

Wenn Sie PHP als CGI-Programm hergestellt haben, können Sie mit **make bench** ein Benchmark Test machen. Bedenken Sie, dass wenn der safe_mode standardmäßig aktiviert wurde, es sein kann, dass der Benchmark nicht fähig ist, den Benchmark zu vollenden, wenn der Vorgang länger als 30 Sekunden (die erlaubt sind) dauert. Dieses kommt daher, weil die Funktion [set_time_limit\(\)](#) im safe_mode nicht benutzt werden kann. Benutzen Sie die [max_execution_time](#) Konfigurationseinstellungen um die Zeit für Ihre eigenen Skripte einzustellen. **make bench** ignoriert die [Konfigurationsdatei](#).

[Zurück](#)

Installation

[Anfang](#)

[Hoch](#)

[Vor](#)

Installation auf Windows
95/98/NT Systemen

Installation auf Windows 95/98/NT Systemen

Diese Installationsanleitung soll Ihnen helfen, PHP auf Ihrem Windows 9x/NT Webserver zu installieren und zu konfigurieren. Diese Anleitung wurde von [Bob Silva](#) erstellt. Um die aktuellste Fassung (Englisch) zu erhalten, klicken Sie bitte hier: <http://www.umesd.k12.or.us/php/win32install.html>.

Diese Anleitung umfasst die Installation folgender Komponenten:

- Personal Web Server (Neueste Version empfohlen)
- Internet Information Server 3 oder 4
- Apache 1.3.x
- Omni HTTPd 2.0b1

Grundlegende Installations- Schritte

Die folgenden Schritte sollten in jedem Fall und unabhängig vom bevorzugten Webserver ausgeführt werden.

- Entpacken Sie die Dateien, die mit der Distribution mitgeliefert werden in ein Verzeichnis Ihrer Wahl. Wir empfehlen "C:\PHP3".
- Kopieren Sie die 'php3.ini-dist'- Datei in Ihr '%WINDOWS%' - Verzeichnis und benennen Sie sie um in 'php3.ini'. Ihr Windows-Verzeichnis ist im Allgemeinen:
c:\windows für Windows 95/98
c:\winnt oder c:\winnt40 für NT servers
- Editieren Sie nun Ihre 'php3.ini' -Datei mit Ihrem bevorzugtem Editor:
 - Sie müssen den Schlüssel 'extension_dir' so verändern, dass er auf Ihr PHP-Verzeichnis zeigt, bzw. dorthin, wo Sie die zu ladenden Erweiterungen (Module) abgelegt haben. Z.B.:
c:\php3
 - Wenn Sie den Omni Httpd-Server verwenden, überspringen Sie den nächsten Schritt. Geben Sie bei dem Schlüssel 'doc_root' das Stammverzeichnis an, in dem Ihre PHP-Skripte liegen. Sie sollten diese Angabe aus Sicherheits-gründen auf das Dokumentenverzeichnis des Webserver setzen: Apache: '%apache\htdocs' IIS: '%inetpub\wwwroot' PWS: '%webshare\wwwroot'.

- Wählen Sie nun die Erweiterungen (Module) aus, die Sie benutzen möchten, wenn PHP startet. Dieses geschieht, indem Sie die jeweiligen Schlüssel 'extension=php3_*.dll' entkommentieren (Entfernen Sie das ';' vor den Erweiterungen). Einige Erweiterungen stehen in Abhängigkeiten zu zusätzlichen Bibliotheken (Libraries), welche auf Ihrem System installiert sein müssen, um korrekt zu funktionieren. Die [FAQ](#) - Sektion von PHP liefert hier weitere Informationen über Bezugsquellen weiterer Bibliotheken. Bei selten benutzten Modulen empfiehlt es sich diese im Hinblick auf Systemressourcen dynamisch in das einzelne Skript zu laden. Module können Sie dynamisch mittels folgender Anweisung laden: **dl("php_*.dll");**
- Wenn Sie den PWS oder IIS benutzen, können Sie die browscap.ini auf folgende Verzeichnisse setzen: 'c:\windows\system\inetsrv\browscap.ini' für Windows 95/98 und 'c:\winnt\system32\inetsrv\browscap.ini' für NT Server. Zusätzliche Informationen zur Benutzung der browscap.ini und Ihrer Funktionalität erhalten Sie hier: [mirror](#), klicken Sie auf den "source" button um ein Anwendungsbeispiel zu erhalten.

Den DLLs für PHP Erweiterungen sind allen das Prefix 'php3_' vorangestellt. Dieses soll Unklarheiten zwischen PHP Erweiterungen und ihren unterstützenden Bibliotheken vorbeugen.

Windows 95/98/NT und PWS/IIS 3

Die empfohlene Methode, um diese Server zu konfigurieren ist es, die 'php_iis_reg.inf'- Datei zu benutzen. Sie können diese Datei editieren, um sie Ihren Wünschen anzupassen. Sie können aber die Konfiguration auch "von Hand" vornehmen.

(!)WARNUNG: Diese Schritte erfordern einen Eingriff in die Registrierungsdatei von Windows. Sie sollten auf jeden Fall zuvor eine Sicherheitskopie Ihrer Registrierungsdatei erstellen. Sollte Ihnen bei der Bearbeitung der Registrierungsdatei ein Fehler unterlaufen, kann dieses ein instabiles oder nicht mehr startendes System zur Folge haben. Wir lehnen jede Verantwortung für Schäden und Verluste ab. Bitte seien Sie also bei der folgenden Methode sehr aufmerksam.

- Starten Sie regedit.exe
- Gehen Sie zu HKEY_LOCAL_MACHINE /System /CurrentControlSet /Services /W3Svc /Parameters /ScriptMap.
- Wählen Sie aus dem 'Bearbeiten-Menü' folgendes aus: Neu->Zeichenfolge.
- Geben Sie hier die Erweiterung Ihrer .php3- Dateien an. Bsp.: .php3
- Klicken Sie nun doppelt auf den Wert der Zeichenkette, und geben Sie hier als zugeordnetes Programm Ihre php.exe an. Bsp.: c:\php3\php.exe %s %s. Die Folge '%s %s' ist unbedingt einzufügen, PHP würde sonst nicht korrekt arbeiten.
- Wiederholen Sie diese Schritte für jede weitere Datei-Erweiterung, die sie mit PHP verbinden wollen.
- Wechseln Sie zu: HKEY_CLASSES_ROOT

- Wählen Sie aus dem 'Bearbeiten-Menü' folgendes aus: Neu -> Schlüssel.
- Benennen Sie den Schlüssel wie Ihre Datei-Erweiterung (.php3)
- Wählen Sie nun den neu erstellten Schlüssel aus, und weisen Sie ihm den Standardwert `phpfile` zu.
- Wiederholen Sie diese Schritte, wenn Sie weitere Datei-Erweiterungen konfigurieren wollen.
- Erzeugen Sie nun einen weiteren Schlüssel Neu->schlüssel im Verzeichnis `HKEY_CLASSES_ROOT` und benennen Sie ihn mit `phpfile`.
- Wählen Sie den neuen Schlüssel `phpfile` aus, und weisen Sie ihm den Standardwert `PHP Script` zu.
- Erstellen Sie nun unterhalb von `phpfile` mittels Rechtsklick einen neuen Schlüssel und nennen Sie ihn `Shell`.
- Unterhalb von `Shell` erzeugen Sie nun einen weiteren Schlüssel und nennen ihn `open`.
- Unterhalb von `open` erzeugen Sie jetzt noch einen Schlüssel und nennen ihn `command`.
- Wählen Sie den Schlüssel `command` aus, und geben Sie ihm als Standardwert den Pfad zur `php.exe` mit folgender Schreibweise an: `c:\php3\php.exe -q %1`. (vergessen Sie bitte nicht das `%1`).
- Beenden Sie Regedit.

PWS and IIS 3 Benutzer sollten nun ein voll funktionsfähiges System haben. IIS 3 Benutzer können ein kleines [Tool](#) von Steven Genusa benutzen, um Ihre Skript-Maps zu editieren.

Windows NT und IIS 4

Um PHP auf Windows NT und dem IIS4 zu installieren, folgen Sie dieser Anleitung:

- Im Internet Service Manager (MMC) selektieren Sie die Webseite oder ein Ausgangsverzeichnis Ihrer Anwendungen.
- Öffnen Sie die Eigenschaften Ihres Verzeichnisses (Rechtsklick), danach klicken Sie auf die Registerkarte Homeverzeichnis, Virtuelles Verzeichnis oder Verzeichnis.
- Klicken Sie auf die Schaltfläche Konfiguration und anschliessend auf die Registerkarte Anwendungs-Mapping.
- Klicken Sie auf Hinzufügen und geben Sie in der Ausführungsbox ein:
`c:\path-to-php-dir\php.exe %s %s`. Sie müssen `%s %s` unbedingt am Ende eingeben, sonst wird PHP nicht korrekt arbeiten.
- In der Ausführungsbox geben Sie die Dateierweiterungen an, mit der Ihre PHP-Skripte verbunden werden sollen. (Sie müssen Schritt 5 und 6 für jede zusätzlich gewünschte Erweiterung wiederholen. `.php3` und `.html` sind die gebräuchlichsten.)

- Stellen Sie im Internet Service Manager die geeigneten Sicherheitsmechanismen ein. Falls Ihr NT-Server das NTFS- Dateisystem benutzt, dann fügen Sie bitte Ausführungsrechte für I_USR_ dem Verzeichnis hinzu, das das Programm `php.exe` enthält.

Windows 9x/NT and Apache 1.3.x

Um den Apache zu konfigurieren, müssen Sie die Datei `httpd.conf` zum Bearbeiten öffnen (Bei älteren Apache-Versionen sind noch weitere Konfigurations-Dateien vorhanden; hier muss dann auch die `srml.conf` editiert werden).

Obwohl der Apache auf mannigfaltige Weise konfiguriert werden kann, sollte die hier vorgestellte Art selbst für einen "Newcomer" keine grösseren Hindernisse darstellen. Für weitere Konfigurationsmöglichkeiten konsultieren Sie bitte die Apache Docs.

- `ScriptAlias /php3/ "c:/path-to-php-dir/"`
- `AddType application/x-httpd-php3 .php3`
- `AddType application/x-httpd-php3 .phtml`
- `Action application/x-httpd-php3 "/php3/php.exe"`

Das Quell-Code-Highlighting Feature nutzen sie mit der Funktion `showSource()`. Erstellen Sie z.B. ein PHP-Skript in welchem Sie folgenden Code eingeben: `<?php show_source ("original_php_script.php3"); ?>`. Ersetzen Sie `original_php_script.php3` durch den Namen des Skriptes, dessen Quellcode Sie anzeigen wollen (Dieses ist nur ein Weg, das Feature zu nutzen). Die Farben der Syntax-Highlighting können Sie in der `php.ini` im Abschnitt 'Languages Options' einstellen. *Bemerkung:* Auf Win-Apache müssen alle Backslashes in einer Verzeichnis-Anweisung in Forward-Slashes (normaler Slash) konvertiert werden.

Omni HTTPd 2.0b1 für Windows

Dieser ist von allen Server der am einfachsten zu konfigurierende:

Schritt 1: Installieren Sie den Omni server

Schritt 2: Rechtsklick auf das blaue OmniHTTPd Icon in der System Tray und selektieren Sie **Eigenschaften**

Schritt 3: Klicken Sie auf **Web Server Global Settings**

Schritt 4: Bei der Registerkarte 'External', geben Sie folgendes ein: `virtual = .php3 | actual = c:\path-to-php-dir\php.exe`

Schritt 5: Bei der Registerkarte **Mime** geben Sie folgendes ein: `virtual = wwwserver/stdcgi | actual = .php3`

Schritt 6: Klicken Sie auf **OK**. Fertig :-)

Wiederholen Sie Schritt 2-6 für jede Dateierweiterung, die Sie mit PHP verbinden wollen.

PHP Module

Tabelle 2-1 PHP Module

php3_calendar.dll	Kalender - Funktionen
php3_crypt.dll	Verschlüsselungs - Funktionen (MD5)
php3_dbase.dll	dBase - Funktionen
php3_dbm.dll	GDBM Emulation via Berkely DB2 library
php3_filepro.dll	READ ONLY Zugang zur Filepro Datenbank
php3_gd.dll	GD Library - Funktionen für gif-Manipulation
php3_hyperwave.dll	HyperWave - Funktionen
php3_imap4r2.dll	IMAP 4 - Funktionen
php3_ldap.dll	LDAP - Funktionen
php3_mysql1.dll	mSQL 1 client
php3_mysql2.dll	mSQL 2 client
php3_mssql.dll	MSSQL client (benötigt MSSQL DB-Bibliotheken)
php3_mysql.dll	MySQL - Funktionen
php3_nsmail.dll	Netscape Mail - Funktionen
php3_oci73.dll	Oracle - Funktionen
php3_snmp.dll	SNMP get and walk Funktionen (nur NT !)
php3_zlib.dll	ZLib - Funktionen

[Zurück](#)

Installation auf UNIX Systemen

[Anfang](#)

[Hoch](#)

[Vor](#)

Probleme?

Probleme?

Lesen Sie die FAQ

Einige Problem tauchen öfters auf als andere. Die am häufigsten auftretenden Probleme sind in der PHP-FAQ aufgelistet: <http://www.php.net/FAQ.php>

Bug Reports

Wenn Sie glauben, einen Bug gefunden zu haben, melden Sie ihn uns bitte. Die PHP-Entwickler kennen ihn wahrscheinlich noch nicht, und wenn Sie ihn nicht melden, stehen die Chancen schlecht, das er ausgemerzt wird. Sie können einen Bug mittels des "bug-tracking system" unter <http://www.php.net/bugs.php> melden.

Andere Probleme

Wenn Ihnen trotzdem noch nicht geholfen werden konnte, könnte es sein, das jemand auf der Mailingliste Ihnen helfen kann. Bevor Sie jedoch ein Problem an die Mailingliste weitergeben, sollten Sie zuerst das Archiv der Liste durchsuchen. Das Archiv finden Sie unter <http://www.php.net/>. Um sich auf der Mailingliste anzumelden, reicht es, eine leere Email an php3-subscribe@lists.php.net zu schicken. Die Adresse der Mailingliste ist `php3@lists.php.net`.

Wenn Sie Hilfe von der Mailingliste haben möchten, dann achten Sie bitte auf folgende Verhaltensmaßnahmen:

Beschreiben Sie Ihr Problem so präzise wie möglich, und fügen Sie Details Ihrer Entwicklungsumgebung hinzu (Betriebssystem, Server, PHP-Version, CGI oder Modul- Version etc). Fügen Sie genug Quellcode hinzu, so dass andere Ihr Problem nachvollziehen können

Verschicken Sie keine html-codierte Emails. Solche Emails können nicht von allen Email-Programmen gelesen werden, was zur Folge hat, das Ihre Email unter Umständen gleich im Papierkorb landet.

Halten Sie die Netiquette ein.

Kapitel 3 Configuration

Inhaltsverzeichnis

[Die Konfigurationsdatei](#)

Die Konfigurationsdatei

Die Konfigurationsdatei (`php3.ini` in PHP 3.0.x, und `php.ini` in PHP 4.0) wird geladen, wenn PHP gestartet wird. Wurde PHP als Modul in den Webserver einkompiliert, dann geschieht dieses nur wenn der Server gestartet wird. Ist PHP als CGI-Version konfiguriert worden, dann geschieht dieses bei jedem Aufruf.

Wenn Sie PHP in der Modul-Variante benutzen, können Sie die Konfigurationseinstellungen auch mittels der Apache-Konfigurations- datei bzw. mittels `.htaccess`-Dateien ändern.

Bei PHP 3.0.x existieren Apache-Anweisungen, die mit jeder Konfigurationseinstellung der `php3.ini` korrespondieren, ausgenommen Einstellungen mit dem Prefix "`php3_`".

Bei PHP 4.0 gibt es nur noch wenige Apache-Anweisungen, die es Ihnen erlauben, die Konfigurationseinstellungen zu ändern.

`php_value` *name value*

Dieses setzt den Wert der spezifizierten Variablen.

`php_flag` *name on | off*

Dieser Schalter wird benutzt, um die Boolean-Konfigurations- Option zu aktivieren.

`php_admin_value` *name value*

Dieser Wert setzt den Wert der spezifischen Variablen. "Admin" Konfigurations-Einstellungen können nur innerhalb der Haupt-Konfigurationsdatei des Apache gesetzt werden, nicht etwa über eine `.htaccess` Datei.

`php_admin_flag` *name on | off*

Dieser Schalter wird benutzt, um die Boolean-Konfigurations- Option zu aktivieren.("Admin")

Die Konfigurationseinstellungen können Sie in der Ausgabe der [phpinfo\(\)](#) Datei einsehen. Ebenfalls können Sie Zugang zu den individuellen Konfigurationseinstellungen über [get_cfg_var\(\)](#) bekommen.

Allgemeine Konfigurationseinstellungen

asp_tags boolean

Dieser Schalter aktiviert die Unterstützung von ASP `<% %>` Tags als Erweiterung zu den üblichen `<?php ?>` Tags. Dieses beinhaltet auch die Kurzform der Variablenausgabe `<%= $value %>`.

Weitere Informationen finden Sie hier: [Escaping from HTML](#).

Anmerkung: Die Unterstützung für ASP -Tags wurde in Version 3.0.4. hinzugefügt

auto_append_file string

Hier können Sie eine Datei angeben, die automatisch nach der Haupt-Datei aufgerufen wird. Die Datei wird automatisch in die aufgerufene Datei eingebettet, als wenn Sie die Datei mittels der [include\(\)](#) Funktion eingebunden hätten. So wird [include_path](#) benutzt.

Der Wert none deaktiviert auto-appending.

Anmerkung: Wird das Skript mit der Funktion [exit\(\)](#) beendet, wird auto-append *not* aktiv.

auto_prepend_file string

Hier können Sie eine Datei angeben, die automatisch vor der Hauptdatei aufgerufen wird. Die Datei wird automatisch in die aufgerufene Datei eingebettet, als wenn Sie die Datei mittels der [include\(\)](#) Funktion eingebunden hätten. So wird [include_path](#) benutzt.

Der Wert none deaktiviert auto-prependung.

cgi_ext string

display_errors boolean

Dieser Wert muß "on" sein, damit Fehlermeldungen an die Konsole (Prompt oder Browser) gesendet werden können.

doc_root string

Hier wird das Stammverzeichnis der PHP-Skripte eingegeben. Üblicherweise handelt es sich hier um das DocumentRoot des Servers (Apache: htdocs) Diese Angabe wird nur benutzt, wenn Sie einen Wert enthält. Wenn PHP mit [safe mode](#) konfiguriert wurde, werden alle PHP-Skripte außerhalb dieses Directorys ignoriert.

engine boolean

Diese Option ist in erster Linie nur sinnvoll, wenn PHP als Modul in den Apache einkompiliert wurde. Sie wird von Seiten benutzt, die den PHP-Parser auf einer per-directory oder per-virtual Server Basis aus bzw. einschalten wollen. Wenn Sie **engine off** in den dafür vorgesehenen Blöcken in der `httpd.conf` Datei benutzen, kann PHP aktiviert bzw. deaktiviert werden.

error_log string

Hier können Sie die Datei angeben, in der Skript-Fehler protokolliert werden sollen. Wenn Sie statt eines Dateinamens `sys log` eintragen, wird stattdessen das Ereignisprotokoll von

WindowsNT genutzt. Auf UNIX Systemen syslog(3) verwendet. Windows 95/98 unterstützen dieses nicht.

error_reporting integer

Hier können Sie die Genauigkeit der Fehlermeldungen einstellen. Der eingetragene Wert ist ein Bitwert, und wird als Summe folgender Werte des Error Reporting Levels gebildet:

Tabelle 3-1 Error Reporting Levels

bit value	Aktivierte Protokollierung
1	normale Fehler
2	normale Warnungen
4	Fehler des parsers, meistens Syntaxfehler
8	unkritische Warnungen, die ignoriert werden können. Sinnvoll in der Entwicklungsphase.

Standart Wert ist hier 7 (normale Fehler, normale Warnungen, Fehler des Parsers werden angezeigt).

open_basedir string

Limitiert die Anzahl der Dateien, die von PHP in einem bestimmten Verzeichnis geöffnet werden können.

Wenn ein Skript versucht, eine Datei mit z.B. fopen oder gzopen zu öffnen, wird der Ort der Datei überprüft. Wenn sich die Datei außerhalb des spezifizierten Verzeichnisses befindet, wird PHP es nicht öffnen. Alle symbolischen Links sind hier mit eingeschlossen, so das es auch nicht möglich ist dieses Verbot mittels symlink zu umgehen.

Der Wert . gibt an, das das Verzeichnis, in dem das Skript abgespeichert ist, als Basis-Verzeichnis genutzt wird.

Unter Windows werden Verzeichnisse mit Semikolon getrennt, unter allen anderen Betriebssystemen mit einem Doppelpunkt. Wenn PHP als Modul in den Apache einkompiliert wurde, werden open_basedir paths von "Eltern-" Verzeichnissen nun automatisch vererbt.

Anmerkung: Die Unterstützung für multiple Verzeichnisse wurde in Version 3.0.7. hinzugefügt

Die Standarteinstellung ist, alle Dateien zum parsen freizugeben.

gpc_order string

Beeinflußt die Reihenfolgen von GET/POST/COOKIE Variablen parsing. Die Standarteinstellung ist GPC. Wenn Sie diesen Wert auf z.B."GP" setzen, ignoriert PHP Cookies, und wird jede GET Methoden Variable mit POST Methoden Variablen des gleichen Namens überschreiben.

ignore_user_abort string

Standartmäßig auf ON. Wenn Sie diese Einstellung auf OFF setzen, werden PHP-Skripte beendet, sobald sie versuchen eine Ausgabe zu erzeugen, nachdem der Client die Verbindung beendet hat.

[ignore_user_abort\(\)](#).*include_path* string

Hier können Sie ein Verzeichnis angeben, in dem die [require\(\)](#), [include\(\)](#) and [fopen_with_path\(\)](#) Funktionen nach Dateien suchen. Das Format ist ähnlich den System's PATH Umgebungsvariablen. Eine Liste von Verzeichnissen, getrennt durch einen Doppelpunkt bei UNIX oder ein Semikolon bei WINDOWS.

Beispiel 3-1 UNIX include_path

```
include_path=.: /home/httpd/php-lib
```

Beispiel 3-2 Windows include_path

```
include_path=".;c:\www\phplib"
```

Der Standartwert für diese Option ist . (nur das aktuelle Verzeichnis).

isapi_ext string*log_errors* boolean

Dieser Schalter entscheidet, ob Skript- Fehler im Fehler-Logfile des Servers protokolliert werden sollen. Diese Option ist also Server-spezifisch.

magic_quotes_gpc boolean

Dieser Schalter setzt den magic_quotes Zustand für GPC (Get/Post/Cookie) Operationen. Wenn magic_quotes auf ON stehen, werden automatisch alle ' (single-quote), " (double quote), \ (backslash) und NUL's mit einem backslash versehen. Wenn auch magic_quotes_sybase auf ON steht, wird ein single-quote mit einem weiteren single-quote anstatt eines backslashes versehen.

magic_quotes_runtime boolean

Wenn der Schalter *magic_quotes_runtime* aktiviert ist, werden Anführungszeichen der meisten Funktionen welche Daten aus jeglicher Art von externer Quelle, eingeschlossen Datenbanken und Textdateien, mit einem Backslash versehen. Wenn auch der Schalter *magic_quotes_sybase* aktiviert ist, wird ein einfaches Anführungszeichen mit einem einfachen Anführungszeichen anstatt eines Backslashes versehen.

magic_quotes_sybase boolean

Wenn der Schalter *magic_quotes_sybase* zusätzlich zu *magic_quotes_gpc* oder *magic_quotes_runtime* aktiviert ist, werden einfache Anführungszeichen mit einfachen Anführungszeichen anstatt eines Backslashes versehen.

max_execution_time integer

Dieser Wert gibt die maximale Ausführungslänge eines Skriptes in Sekunden an, bevor es vom Parser abgebrochen wird. Dieses soll Abstürzen des Servers durch schlecht geschriebene/ falsche Skripte vorbeugen. Bei komplexen Datenbankabfragen sollte man jedoch bedenken, das der Standartwert unter Umständen zu klein sein kann.

memory_limit integer

Dieser Wert gibt den maximal erlaubten Speicherplatzverbrauch eines PHP Skriptes an. Diese Einstellung soll den Server vor Speicherverschwendung durch schlecht programmierte Skripte bewahren.

nsapi_ext string*short_open_tag* boolean

Gibt an, ob die Kurzform (**<? ?>**) eines öffnenden PHP Tags erlaubt ist. Wenn Sie PHP in Kombination mit XML nutzen, müssen sie diese Option deaktivieren. Wenn Sie die Option deaktiviert haben, müssen sie die Langform des öffnenden Tags (**<?php ?>**) verwenden.

sql.safe_mode boolean*track_errors* boolean

Wenn dieses Option aktiviert ist, wird der letzte Fehler immer in der Umgebungsvariablen \$php_errormsg abgelegt sein.

track_vars boolean

Wenn dieser Schalter aktiviert ist, werden GET, POST und Cookie - Werte in den Umgebungsvariablen-Arrays \$HTTP_GET_VARS, \$HTTP_POST_VARS und \$HTTP_COOKIE_VARS abgelegt.

upload_tmp_dir string

Hier wird das temporäre Verzeichnis angegeben, in welchem Dateien gespeichert werden, die mittels file upload auf den Server geladen werden. Dieses Verzeichnis muss schreibbar sein.

user_dir string

Wenn Sie PHP für User hosten, dann können persönliche Verzeichnisse der Art ~username angelegt werden. Der Wert user_dir zeigt auf das Stammverzeichnis, unter dem diese Nutzerverzeichnisse liegen. z.B. public_html.

warn_plus_over_loading boolean

Wenn dieser Schalter aktiviert ist, gibt PHP eine Warnung aus, wenn der plus (+) Operator anstatt des Punkt (.) Operators für die String-Concanetation benutzt wurde.

Mail Konfiguration Direktiven

SMTP string

Dieser Wert muss nur in der Windows-Umgebung gesetzt werden. Hier wird der DNS-Name oder die IP-Adresse des SMTP-Servers eingetragen, wenn mails mit der [mail\(\)](#) Funktion verschickt werden sollen.

sendmail_from string

Hier können Sie den Absender der Emails eintragen, wenn Sie von Windows aus Emails verschicken wollen.

sendmail_path string

Hier geben Sie den Pfad zum **sendmail** Programm an. Für gewöhnlich ist dies `/usr/sbin/sendmail` oder `/usr/lib/sendmail`. **configure** versucht zwar den Pfad zum sendmail Programm selbstständig herauszufinden, aber für den Fall, das dieses nicht funktioniert, können Sie den Pfad hier eintragen.

Auf Systemen, wo Sendmail nicht benutzt wird, sollten sie -wenn vorhanden- diesen Pfad auf den Sendmail Wrapper/ Ersatz setzen. Zum Beispiel: [Qmail](#) Benutzer könne den Pfad normalerweise wie folgt setzen: `/var/qmail/bin/sendmail`.

Safe Mode Konfiguration Direktiven

safe_mode boolean

Wenn Sie diesen Schalter auf ON setzen, können Sie PHP im sicheren Modus betreiben. Mehr dazu erfahren Sie unter: [Security chapter](#)

safe_mode_exec_dir string

Wenn Sie PHP im sicheren Modus betreiben, werden die [system\(\)](#) Funktion und andere Systemprogramme ausführende Funktionen nicht gestartet, wenn Sie sich nicht in diesem Verzeichnis befinden.

Debugger Konfiguration Direktiven

debugger.host string

DNS name oder IP Adresse des Host, der vom Debugger benutzt wird.

debugger.port string

Port Nummer, die vom Debugger benutzt wird.

debugger.enabled boolean

Hier können Sie den Debugger aktivieren bzw. deaktivieren.

Extension Loading Direktiven

enable_dl boolean

Diese Option ist eigentlich nur sinnvoll, wenn Sie PHP als Apache- Modul benutzen. Dann können Sie das "Einladen" der PHP- Extensionen dynamisch mit der Funktion [dl\(\)](#) aktivieren bzw. deaktivieren per Virtual Server or per Directory.

Der Hauptgrund das dynamische Laden von extensionen zu deaktivieren ist Sicherheit. Wenn die Option aktiviert ist, ist es möglich, `safe_mode` und `open_basedir` Restriktionen zu ignorieren.

Standartmäßig wird das dynamische Laden erlaubt, außer wenn Sie PHP im `safe_mode` betreiben. Im `safe_mode` ist es nie möglich die die [dl\(\)](#) Funktion zu benutzen.

extension_dir string

Hier geben Sie das Verzeichnis an, indem PHP dynamisch zu ladenden Extensionen findet.

extension string

Hier geben Sie an, welche Extensionen geladen werden sollen, wenn PHP gestartet wird.

MySQL Konfigurations Direktiven

mysql.allow_persistent boolean

Erlaubt persistente Verbindungen.

mysql.default_host string

Hier wird der Host (Computername) des Computers eingetragen, auf dem MySQL installiert ist (wird benutzt, falls kein anderer Name angegeben wird).

mysql.default_user string

Hier wird der Name des Standartbenutzers eingegeben (wird benutzt, falls kein anderer Name angegeben wird).

mysql.default_password string

Hier wird das Standartpasswort eingegeben (wird benutzt, falls kein anderes Passwort angegeben wurde).

mysql.max_persistent integer

Hier geben Sie die maximal erlaubte Anzahl persistenter MySQL- Verbindungen pro Prozess an.

mysql.max_links integer

Hier geben sie die maximale Anzahl der gesamten Verbindungen pro Prozess an (persistenten und normale).

mSQL Konfigurations Directives

msql.allow_persistent boolean

Erlaubt persistente mSQL-Verbindungen.

msql.max_persistent integer

Hier geben Sie die maximal erlaubte Anzahl persistenter mSQL- Verbindungen pro Prozess an.

msql.max_links integer

Hier geben sie die maximale Anzahl der gesamten Verbindungen pro Prozess an (persistenten und normale).

Postgres Konfigurations Direktiven

pgsql.allow_persistent boolean

Erlaubt persistente Postgres - Verbindungen.

pgsql.max_persistent integer

Hier geben Sie die maximal erlaubte Anzahl persistenter Postgres- Verbindungen pro Prozess an.

pgsql.max_links integer

Hier geben sie die maximale Anzahl der gesamten Verbindungen pro Prozess an (persistenten und normale).

Sybase Konfigurations Direktiven

sybase.allow_persistent boolean

Erlaubt persistente Sybase - Verbindungen.

sybase.max_persistent integer

Hier geben Sie die maximal erlaubte Anzahl persistenter Sybase- Verbindungen pro Prozess an.

sybase.max_links integer

Hier geben sie die maximale Anzahl der gesamten Verbindungen pro Prozess an (persistenten und normale).

Sybase-CT Konfigurations Direktiven

sybct.allow_persistent boolean

Erlaubt persistente Sybase-CT Verbindungen. Standartwert ist ON.

sybct.max_persistent integer

Hier geben Sie die maximal erlaubte Anzahl persistenter Sybase- Verbindungen pro Prozess an. Der Standartwert ist -1 (unbegrenzt).

sybct.max_links integer

Hier geben sie die maximale Anzahl der gesamten Verbindungen pro Prozess an (persistenten und normale). Standartwert ist -1 (unbegrenzt).

sybct.min_server_severity integer

Server Nachrichten mit einer höherem oder gleichem Gewicht wie *sybct.min_server_severity* werden als Warnung ausgegeben. Dieser Wert kann auch in einem Skript mittels der **sybase_min_server_severity()** Funktion gesetzt werden. Standartwert ist 10, wodurch Fehler mit Informations- "Gewicht" oder höher ausgegeben werden.

sybct.min_client_severity integer

Client library Nachrichten mit höherem oder gleichem Gewicht wie *sybct.min_client_severity* werden als Warnungen ausgegeben. Dieser Wert kann auch in einem Skript mit der Funktion **sybase_min_client_severity()** gesetzt werden. Der Standartwert ist hier 10, was eine Ausgabe im Endeffekt deaktiviert.

sybct.login_timeout integer

Hier können Sie die maximale Zeit in Sekunden angeben, die gewartet wird bis eine Verbindung erfolgreich aufgebaut wurde (also bevor eine Fehlermeldung erscheint). Beachten Sie, das wenn die *max_execution_time* für einen Verbindungsversuch vorbei ist, Ihr Skript beendet wird, bevor es eine Fehlermeldung (Oder andere Aktion) ausgeben kann. Der Standartwert ist hier 1 Minute.

sybct.timeout integer

Hier können Sie die maximale Zeit in Sekunden angeben, die für einen erfolgreichen *select_db* oder andern DB-Query gewartet wird bis eine Fehlermeldung erscheint. Beachten Sie, das wenn die *max_execution_time* für einen DB-Query vorbei ist, Ihr Skript beendet wird, bevor es eine Fehlermeldung (Oder andere Aktion) ausgeben kann. Der Standartwert ist hier unbegrenzt.

sybct.hostname string

Hier können Sie den Host angeben, von dem die Verbindung ausgehen soll. Sie können diesen Wert mit *sp_who* anzeigen. Standartwert ist none.

Informix Konfigurations Direktiven

ifx.allow_persistent boolean

Erlaubt persistente Informix Verbindungen.

ifx.max_persistent integer

Hier geben Sie die maximal erlaubte Anzahl persistenter Informix- Verbindungen pro Prozess an.

ifx.max_links integer

Hier geben sie die maximale Anzahl der gesamten Verbindungen pro Prozess an (persistenten und normale).

ifx.default_host string

Hier geben Sie den Standart Host für eine Verbindung ein, der benutzt wird, wenn kein anderer Host in [ifx_connect\(\)](#) oder [ifx_pconnect\(\)](#) angegeben wurde.

ifx.default_user string

Hier geben Sie den Standart Benutzer für eine Verbindung ein, das benutzt wird, wenn kein anderer Benutzer in [ifx_connect\(\)](#) oder [ifx_pconnect\(\)](#) angegeben wurde.

ifx.default_password string

Hier geben sie das Standart Passwort für eine Verbindung ein, das benutzt wird, wenn kein anderes

Passwort in [ifx_connect\(\)](#) oder [ifx_pconnect\(\)](#) angegeben wurde.

ifx.blobinfile boolean

Setzen Sie diesen Wert auf true, wenn sie Blob-Felder in einer Datei zurückgegeben haben wollen, auf false wenn Sie sie im Speicher haben wollen. Diesen Eintrag können Sie mit [ifx_blobinfile_mode\(\)](#) während der Ausführung überschreiben.

ifx.textasvarchar boolean

Setzen Sie diesen Wert auf true, wenn Sie TEXT-Felder als normale Strings in SELECT-Anweisungen zurückgegeben haben wollen, auf false, wenn Sie sie BLOB-ID Parameter benutzen wollen. Diesen Eintrag können Sie mit [ifx_textasvarchar\(\)](#) während der Ausführung überschreiben.

ifx.byteasvarchar boolean

Setzen Sie diesen Wert auf true, wenn Sie BYTE-Felder in SELECT Abfragen als normale Strings zurückgeliefert haben wollen, auf false, wenn Sie BLOB- ID Parameter benutzen wollen. Diesen Eintrag können Sie mit [ifx_textasvarchar\(\)](#) während der Ausführung überschreiben.

ifx.charasvarchar boolean

Setzen Sie diesen Wert auf true, wenn Sie vorangehende Spaces von CHAR-Feldern bei der Abfrage abschneiden wollen.

ifx.nullformat boolean

Setzen Sie diesen Wert auf true, wenn Sie NULL-Felder als String "NULL", auf false, wenn Sie diese als leeren String "" zurückgeliefert haben wollen. Sie können diesen Wert während der Ausführung mit der Funktion [ifx_nullformat\(\)](#) überschreiben.

BC Math Konfigurations Direktiven

bcmath.scale integer

Anzahl der Nachkommastellen der bcmath-Funktion.

Browser Capability Konfigurations Direktiven

browscap string

Name der browser_capabilities Datei. Mehr hierüber erfahren Sie unter [get_browser\(\)](#).

Unified ODBC Configuration Directives

uodbc.default_db string

ODBC Standartquelle, die benutzt wird, wenn keine andere in [odbc_connect\(\)](#) oder

[odbc_pconnect\(\)](#) angegeben wurde.

uodbc.default_user string

ODBC Standart- Benutzername, der benutzt wird, wenn kein anderer Benutzer in [odbc_connect\(\)](#) oder [odbc_pconnect\(\)](#) definiert wurde.

uodbc.default_pw string

ODBC Standart- Passwort, das benutzt wird, wenn kein anderes Passwort in [odbc_connect\(\)](#) oder [odbc_pconnect\(\)](#) angegeben wurde.

uodbc.allow_persistent boolean

Erlaubt persistente ODBC- Verbindungen.

uodbc.max_persistent integer

Hier geben Sie die maximal erlaubte Anzahl persistenter ODBC- Verbindungen pro Prozess an.

uodbc.max_links integer

Hier geben Sie die maximal erlaubte Anzahl der gesamten ODBC- Verbindungen pro Prozess an (persistente und normale).

[Zurück](#)

Probleme?

[Anfang](#)

[Hoch](#)

[Vor](#)

Sicherheit

Kapitel 4 Sicherheit

Inhaltsverzeichnis

[CGI-Version](#)

[Apache-Modul](#)

PHP ist eine mächtige Sprache, und der Interpreter, der in einen Webserver als Modul oder als separate CGI-Version eingebunden werden kann, kann auf Dateien zugreifen, Befehle ausführen und Netzwerkverbindungen zu einem Server herstellen. Diese Eigenschaften können einen Webserver unsicher machen, wenn man es bei den Voreinstellungen belässt. PHP wurde besonders dafür entwickelt, um eine sicherere Sprache als Perl oder C für die Erstellung von CGI-Programmen bereitzustellen. Mit der richtigen Wahl der Einstellungen beim Kompilieren und zur Laufzeit bietet PHP genau die Kombination aus Freiheit und Sicherheit, die gerade benötigt wird.

Da es sehr viele verschiedene Möglichkeiten gibt, PHP zu nutzen, gibt es viele Konfigurationseinstellungen, die das Verhalten von PHP beeinflussen. Eine große Auswahl an Einstellungen garantiert, daß man PHP für vielerlei Zwecke einsetzen kann, allerdings bedeutet es auch, dass es Kombinationen gibt, die zur Folge haben, daß die Installation nicht genug Sicherheit bietet. Dieses Kapitel beschreibt die verschiedenen Kombinationen der Konfigurationseinstellungen und unter welchen Gegebenheiten sie sicher genutzt werden können.

CGI-Version

Mögliche Angriffe

PHP als CGI zu nutzen ist eine Möglichkeit für Installationen, bei denen aus irgendwelchen Gründen kein Modul in die Serversoftware eingebunden werden soll (wie beim Apache) oder für Systeme, bei denen verschiedene CGI-Wrapper genutzt werden sollen, um sichere chroot und setuid Umgebungen für Scripts zu schaffen. Bei dieser Konfiguration wird das ausführbare PHP-Binary üblicherweise im cgi-bin Verzeichnis des Webserver installiert. CERT advisory [CA-96.11](#) spricht sich gegen die Plazierung von Interpretern im cgi-bin Verzeichnis aus. Obwohl das PHP-Binary als standalone Interpreter verwendet werden kann, wurde PHP entwickelt, um Angriffe, die durch diese Konfiguration möglich werden, abzuwehren:

- Sytemdatenzugriff: `http://my.host/cgi-bin/php?/etc/passwd`

Die Abfrageinformation in einer URL, die auf ein Fragezeichen (?) folgt, wird durch das CGI-Interface als Kommandozeilenargument an den Interpreter weitergereicht. Üblicherweise wird von Interpretern die Datei geöffnet und ausgeführt, die als erstes Argument in der

Kommandozeile steht.

Beim Aufruf als CGI-Binary verweigert PHP die Interpretierung der Kommandozeilenargumente.

- Zugriff auf beliebige Web-Dokumente auf dem Server:
`http://my.host/cgi-bin/php/secret/doc.html`

Die Pfadinformation, Teil der URL nach dem PHP-Binarynamen, `/secret/doc.html` wurde ursprünglich benutzt, um den Namen der Datei zu übergeben, die durch das CGI-Programm geöffnet und interpretiert werden soll. Normalerweise werden einige Einträge in der Konfigurationsdatei des Webserver benutzt (Apache:Action), um Aufrufe von Dokumenten wie `http://my.host/secret/script.php3` an den PHP-Interpreter umzuleiten. Bei dieser Konfiguration überprüft der Webserver zuerst die Zugriffsrechte im Verzeichnis `/secret` und erstellt anschließend den umgeleiteten Aufruf

`http://my.host/cgi-bin/php/secret/script.php3`. Unglücklicherweise wird, wenn der Aufruf bereits in dieser Form geschieht, vom Webserver keine Zugriffsüberprüfung der Datei `/secret/script.php3`, sondern lediglich der Datei `/cgi-bin/php` vorgenommen. So ist jeder Benutzer, der auf `/cgi-bin/php` zugreifen darf, in der Lage, sich zu jedem geschützten Dokument auf dem Webserver Zugriff zu verschaffen.

Bei PHP kann beim compilieren die Konfigurationsoption [--enable-force-cgi-redirect](#) und zur Laufzeit die Direktive [doc_root](#) and [user_dir](#) benutzt werden, um diesen Angriff zu verhindern, falls der Verzeichnisbaum des Servers Verzeichnisse mit Zugriffsbeschränkungen beinhaltet. Ausführliche Informationen über die verschiedenen Kombinationen siehe weiter unten.

Fall 1: only public files served

Wenn der Server keine Inhalte hat, die durch Passwort oder IP-basierte Zugriffskontrolle geschützt sind, gibt es für diese Konfiguration keinen Grund. Wenn der Webserver keine Redirects erlaubt oder keine Möglichkeit hat, auf einer sicher umgeleiteten Anfrage mit dem PHP-Binary Verbindung aufzunehmen, kann die Option [--disable-force-cgi-redirect](#) im configure-Script angegeben werden. Nichtsdestotrotz müssen Sie sicherstellen, daß Ihre PHP-Skripte nicht auf die eine oder anderen Art des Aufrufs angewiesen sind, weder direkt durch `http://my.host/cgi-bin/php/dir/script.php3` noch durch einen Redirect `http://my.host/dir/script.php3`.

Beim Apache kann der Redirect durch den Gebrauch von AddHandler und Action konfiguriert werden (siehe unten).

Fall 2: --enable-force-cgi-redirect benutzen

Diese Option, die beim Kompilieren verwendet wird, verhindert grundsätzlich den Aufruf von PHP mit einer URL wie beispielsweise `http://my.host/cgi-bin/php/secret_dir/script.php3`. Stattdessen parst PHP in diesem Modus nur dann, wenn der Aufruf durch einen korrekten Redirect des Webserver erfolgt.

Normalerweise wird der Redirect in der Apache-Konfiguration mit den folgenden Einträgen festgelegt:

```
Action php3-script /cgi-bin/php
AddHandler php3-script .php3
```

Diese Option wurde nur mit dem Apache Webserver getestet und ist abhängig davon, wie Apache die nicht standardmäßige CGI-Umgebungsvariable `REDIRECT_STATUS` bei Redirect-Anfragen setzt. Sollte Ihr Webserver keine Möglichkeit unterstützen, zu übermitteln, ob es sich um einen direkte Aufruf oder einen Redirect handelt, können Sie diese Option nicht verwenden und müssen eine der anderen hier beschriebenen Wege gehen, die CGI-version zu nutzen.

Fall 3: `doc_root` oder `user_dir` festlegen

Aktiven Inhalt, wie beispielsweise Skripts und ausführbare Dateien in den Dokumentverzeichnissen des Webserver abulegen, wird manchmal als unsichere Methode angesehen. Wenn, beispielsweise aufgrund von Konfigurationsfehlern, die Skripte nicht ausgeführt, sondern als reguläres HTML-Dokument angezeigt werden kann dies den Verlust von geistigem Eigentum und Sicherheit (Passwörter!) zur Folge haben. Von daher ziehen es viele Sysadmins vor, eine zweite Verzeichnisstruktur für Skripte, auf die nur durch das PHP-CGI zugegriffen werden soll, einzurichten. Diese werden dann stets interpretiert und nicht angezeigt.

Auch wenn die Methode, sicherzustellen dass die Anfragen nicht umgeleitet werden (wie im vorangegangenen Kapitel beschrieben), nicht verfügbar ist, ist es notwendig, ein `doc_root` für Scripts zusätzlich zum Dokumentenverzeichnis einzurichten.

Sie können Das PHP-Skriptverzeichnis durch den Eintrag [doc_root](#) in der [Konfigurationsdatei](#) ändern, oder Sie setzen die Umgebungsvariable `PHP_DOCUMENT_ROOT`. Wenn sie gesetzt ist, wird die CGI-Version von PHP den Namen der zu öffnenden Datei stets mit `doc_root` und der Pfadinformation der Anfrage zusammensetzen, so daß man sicher sein kann, daß ausserhalb dieses Verzeichnisses keine Skripte ausgeführt werden (außer `user_dir`, siehe unten).

Eine weitere hier nützliche Option ist [user_dir](#). Wenn das `user_dir` nicht gesetzt ist, hat nur `doc_root` Einfluß auf die zu öffnende Datei. Der Aufruf einer URL wie `http://my.host/~user/doc.php3` hat nicht zum Ergebnis, daß eine Datei im Home-Verzeichnis des Benutzers geöffnet wird, sondern eine Datei namens `~user/doc.php3` unterhalb des `doc_root` (Ja, ein Verzeichnisname, der mit einer Tilde anfängt [~]).

Ist das `user_dir` beispielsweise auf `public_php` gesetzt, wird eine Anfrage wie `http://my.host/~user/doc.php3` eine Datei namens `doc.php3` im Verzeichnis `public_php` im Heimatverzeichnis des Benutzers öffnen. Wenn das Heimatverzeichnis des Benutzers `/home/user` ist, so ist die ausgeführte Datei `/home/user/public_php/doc.php3`.

Die `user_dir` Erweiterung erfolgte im Hinblick auf die `doc_root` Einstellung. So können Zugriffe auf das Dokumenten- und Benutzerverzeichnis separat gesteuert werden.

Fall 4: PHP-Parser außerhalb des Webverzeichnisbaums

Eine sehr sichere Sache ist es, das PHP-Parser-Binary irgendwo außerhalb des Webverzeichnisbaums zu plazieren, beispielsweise in `/usr/local/bin`. Der einzige Nachteil dieses Verfahrens ist, dass eine Zeile, ähnlich der folgenden:

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

als erste Zeile in jeder Datei, die PHP-Tags enthält, stehen muss. Ausserdem muss die Datei ausführbar sein. Ansonsten ist sie genauso zu behandeln wie ein beliebiges CGI-Script in Perl oder sh oder anderen gebräuchlichen Scriptsprachen, die den `#!` shell-escape Mechanismus nutzen, um sich selbst aufzurufen.

Damit PHP bei dieser Konfiguration `PATH_INFO` und `PATH_TRANSLATED` Informationen korrekt auswertet, sollte der PHP-Parser mit der Option [--enable-discard-path](#) kompiliert werden.

[Zurück](#)

Configuration

[Anfang](#)[Hoch](#)[Vor](#)

Apache-Modul

Apache-Modul

Wenn PHP als Apache-Modul eingesetzt wird, übernimmt es die Benutzerrechte des Apache (üblicherweise user "nobody").

II Sprachreferenz

Inhaltsverzeichnis

5 [Grundlagen der Syntax](#)

6 [Typen](#)

7 [Variablen](#)

8 [Konstanten](#)

9 [Ausdrücke](#)

10 [Operatoren](#)

11 [Kontroll-Strukturen](#)

12 [Funktionen](#)

13 [Klassen und Objekte](#)

Kapitel 5 Grundlagen der Syntax

Inhaltsverzeichnis

[Den HTML-Bereich der Datei verlassen](#)

[Abgrenzung von Anweisungen](#)

[Kommentare](#)

Den HTML-Bereich der Datei verlassen

Es gibt vier Möglichkeiten, den HTML-Bereich einer Skript-Datei zu verlassen und in den "PHP-Modus" zu wechseln:

Beispiel 5-1 Möglichkeiten, den HTML-Bereich zu verlassen

```
1.  <? echo ("Das ist die einfachste Moeglichkeit:eine SGML-´processing
instruction´\n"); ?>

2.  <?php echo("Um XML-konforme Dokumente herzustellen, benutzt man diese Syntax\n");
?>

3.  <script language="php">
        echo ("Manche Editoren (z. B. Frontpage) moegen keine ´processing
instructions´\n");
    </script>

4.  <% echo ("Optional können auch Tags im ASP-Stil verwendet werden"); %>
    <%= $variable; # Das ist ein Abkuerzung fuer "<?echo .." %>
```

Die erste Methode ist nur verfügbar, wenn ´short tags´ aktiviert sind. Dies kann durch die **short_tags()**-Funktion geschehen, durch Aktivieren der [short_open_tag](#)-Einstellung in der PHP-Konfigurationsdatei oder durch das übergeben der ´--enable-short-tags'-Option an den **configure**-Befehl beim Kompilieren des PHP-Parsers.

Die vierte Methode ist nur verfügbar, wenn Tags im ASP-Stil (ASP-style tags) aktiviert sind. Das ist möglich, indem man die [asp_tags](#)-Konfigurationseinstellung aktiviert.

Anmerkung: Die Unterstützung der Tags im ASP-Stil wurde in Version 3.0.4. hinzugefügt.

Der schließende Tag für den PHP-Block schließt eine sofort folgende folgende Zeilenschaltung mit ein, falls diese vorhanden ist.

[Zurück](#)

Sprachreferenz

[Anfang](#)

[Hoch](#)

[Vor](#)

Abgrenzung von Anweisungen

Abgrenzung von Anweisungen

Einzelne Anweisungen werden genauso voneinander abgetrennt wie in C oder Perl - jedes Statement wird mit einem Semikolon beendet.

Auch der schließende Tag (?>) bedeutet das Ende eines Statements, deshalb sind die folgenden Beispiele gleichwertig:

```
<?php
    echo "Dies ist ein Test";
?>

<?php echo "Dies ist ein Test" ?>
```

Kommentare

PHP unterstützt 'C', 'C++' und Unix-shell-artige Kommentare. Beispiele:

```
<?php
    echo "Dies ist ein Test"; // Dies ist ein einzeliger Kommentar im c++-Stil.
    /* Dies ist ein mehrzeiliger Kommentar
       und noch eine Kommentar-Zeile */
    echo "Dies ist noch ein Test";
    echo "... und ein abschliessender Test"; # Dies ist ein Shell-artiger Kommentar.
?>
```

Die einzeligen Kommentar-Arten kommentieren sämtlichen Text bis zum Zeilenende oder bis zum Ende des aktuellen PHP-Blocks aus, je nachdem, was zuerst eintritt.

```
<h1>Dies ist ein <?# echo "einfaches";?> Beispiel.</h1>
<p>Obige Ueberschrift wird lauten: 'Dies ist ein Beispiel.'
```

Man sollte darauf achten, nicht mehrere ´C´-artigen Kommentare ineinander zu setzen, was vorkommen kann, wenn man große Blocks auskommentiert.

```
<?php
/*
    echo "Dies ist ein Test"; /* Dieser Kommentar wird ein Problem verursachen. */
*/
?>
```

Kapitel 6 Typen

Inhaltsverzeichnis

[Integer-Typen](#)

[Fließkomma-Zahlenwerte](#)

[Strings / Zeichenketten](#)

[Arrays](#)

[Objects](#)

[Typen-Tricks](#)

PHP unterstützt die folgenden Typ-Deklarationen:

- [Array](#)
- [Fließkomma-Zahl](#)
- [Integer](#)
- [Object](#)
- [String / Zeichenkette](#)

Der Typ einer Variablen wird normalerweise nicht vom Programmierer bestimmt; vielmehr wird dies zur Laufzeit von PHP entschieden, abhängig vom Zusammenhang in dem die Variable benutzt wird.

Wenn sie die Umwandlung in einen bestimmten Variablen-Typ erzwingen wollen, können sie dies entweder per [cast](#) oder durch Gebrauch der Funktion [settype\(\)](#).

qBeachten Sie, dass eine Variable je nach Gebrauch und Situation auf unterschiedliche Art und Weise typisiert sein kann. Weitere Informationen sehen sie unter [Typ-Veränderung](#).

Integer-Typen

Integer-Typen können durch Gebrauch einer der folgenden Zuweisungsarten angegeben werden:

```
$a = 1234; # Zahlenwert im dezimal-Format
$a = -123; # ein negativer Zahlenwert
$a = 0123; # Oktal-Zahl (83 im dezimal-Format) Achtung:
          # Fehlerquelle bei Strings mit führenden Nullen!
$a = 0x12; # Zahlenwert im hexadezimal-Format
          # (entspricht 18 im dezimal-Format)
```

[Zurück](#)

Kommentare

[Anfang](#)[Hoch](#)[Vor](#)

Fließkomma-Zahlenwerte

Fließkomma-Zahlenwerte

Fließkomma-Zahlenwerte ("doubles") können durch eine der folgenden Anweisungen zugewiesen werden:

```
$a = 1.234;  
$a = 1.2e3;
```


Strings / Zeichenketten

Strings können durch eines von zwei der folgenden Sets von Begrenzungszeichen zugewiesen werden:

Wenn der String eingeschlossen wird von doppelten Anführungszeichen ("), werden die Variablen innerhalb der Zeichenkette ausgewertet (begrenzt durch einige Einschränkungen beim "Parsen" (Ausführen) des PHP- Scripts). Wie in C und Perl kann das Backslash-Zeichen ("\") zur An- / Ausgabe spezieller Zeichen benutzt werden:

Tabelle 6-1 Nicht ausgewertete / übergangene Zeichen:

Zeichenfolge	Bedeutung
\n	neue Zeile
\r	Wagenrücklauf / an Zeilenanfang
\t	horizontaler Tabulator
\\	Backslash / Rückstrich
\\$	Dollar-Symbol
\"	doppelte Anführungszeichen
\[0-7]{1,3}	die Zeichenfolge, die dem regulären Ausdruck entspricht, ist in Oktal-Schreibweise
\x[0-9A-Fa-f]{1,2}	die Zeichenfolge, die dem regulären Ausdruck entspricht ist, in Hexadezimal-Schreibweise

Sie können ein beliebiges anderes Zeichen übergehen bzw. von der Auswertung ausschliessen; in der höchsten Warn-Stufe wird aber eine Warnmeldung ausgegeben.

Die zweite Möglichkeit eine(n) Zeichenkette / String zu begrenzen, ist der Gebrauch des einfachen Anführungszeichens (''), auch Single-Quote genannt. Hier werden einzig die "\" und '\"-Zeichenfolgen von der Auswertung ausgeschlossen. Dies dient der Erleichterung, so dass sie Single-Quotes und Backslashes in einem durch einfache Anführungszeichen begrenzten String benutzen können. Variablen innerhalb von durch Single- Quotes begrenzten Strings werden *nicht* ausgewertet.

Eine andere Möglichkeit Strings einzufassen, besteht im Gebrauch der "here doc"-Syntax ("<<<"). Hierfür ist nach <<< ein Merker zu setzen. Nun folgt der eigentliche String und dann der selbe Merker um den String abzuschliessen. Der schliessende Merker *muss* in der ersten Spalte der Zeile stehen.

Beispiel 6-1 Beispiel zur String-Festlegung per "here doc"-Methode

```
$str = <<<EOD
Beispiel eines Strings
über mehrere Script-Zeilen
durch Gebrauch der here doc-Syntax.
EOD;
```

Anmerkung: Die "here-doc"-Unterstützung wurde in PHP 4 eingeführt.

Zeichenketten / Strings können mittels des '.'-Operators miteinander verbunden werden. Beachten sie, dass hier der '+' (Additions)-Operator nicht funktioniert. Bitte beachten sie auch [String-Operatoren](#) für weitergehende Informationen. Zeichen in einem String können wie ein numerisch-indiziertes Array von Zeichen benutzt werden. Die Syntax ist der von C ähnlich. Sehen sie hierzu die folgenden Beispiele:

Beispiel 6-2 Einige String-Beispiele

```
<?php
/* Zuweisung eines Strings. */
$str = "Dies ist eine Zeichenkette";

/* Erweiterung dieses Strings. */
$str = $str . " mit etwas zusätzlichem Text";

/* andere Möglichkeit zum Erweitern incl. Anweisung für Neue-Zeile. */
$str .= " und einer neuen Zeile am Ende.\n";

/* Dieser String wird am Ende zu: '<p>Nummer: 9</p>' */
$num = 9;
$str = "<p>Nummer: $num</p>";

/* Dieser wird zu '<p>Nummer: $num</p>' */
$num = 9;
$str = '<p>Nummer: $num</p>';
?>

/* Das erste Zeichen eines Strings */
$str = 'Das ist ein Test.'
$first = $str[0];

/* Das letzte Zeichen eines Strings */
$str = 'Das ist immer noch ein Test.'
$last = $str[strlen($str)-1];
?>
```

Umwandlung von Zeichenketten / Strings

Sobald ein String als Zahlenwert angesehen wird, wird der resultierende Wert und Typ wie folgt festgelegt:

Der String wird als double angesehen, wenn er eines der Zeichen '.', 'e', oder 'E' enthält. Ansonsten wird er als Integer-Wert interpretiert.

Der Inhalt wird durch den Anfangsteil des Strings vorgegeben. Sofern der String mit numerischen Daten beginnt, wird ein Zahlen-Wert angenommen. Andererseits kann der Wert auch 0 (Null) sein. Gültig sind auch Werte mit einem optionalen Vorzeichen, gefolgt von einer oder mehreren Zahlen (optional mit einem Dezimalpunkt). Wahlweise kann auch ein Exponent angegeben werden. Dieser besteht aus einem 'e' oder 'E', gefolgt von einer oder mehreren Zahlen.

Sobald der erste Ausdruck ein String ist, hängt der Typ der Variablen vom zweiten Ausdruck ab.

```
$foo = 1 + "10.5";           // $foo ist double (11.5)
$foo = 1 + "-1.3e3";         // $foo ist double (-1299)
$foo = 1 + "bob-1.3e3";      // $foo ist ein Integer-Wert (1)
$foo = 1 + "bob3";           // $foo ist ein Integer-Wert (1)
$foo = 1 + "10 Small Pigs";  // $foo ist ein Integer-Wert (11)
$foo = 1 + "10 Little Piggies"; // $foo ist ein Integer-Wert (11)
$foo = "10.0 pigs " + 1;      // $foo ist ein Integer-Wert (11)
$foo = "10.0 pigs " + 1.0;    // $foo ist double (11)
```

Mehr Informationen über solche Umwandlungen können sie im UNIX-Manual unter strtod(3) finden.

Wenn sie alle Beispiele dieses Abschnitts testen wollen, können sie sie per Cut und Paste in ihr Script übertragen und mit der folgenden Zeile selbst sehen, was passiert:

```
echo "\$foo==\$foo; Typ ist " . gettype( $foo ) . "<br>\n";
```

[Zurück](#)

Fließkomma-Zahlenwerte

[Anfang](#)
[Hoch](#)
[Vor](#)

Arrays

Arrays

Arrays reagieren wie Hash-Tabellen (assoziative Arrays) und indizierte Arrays (Vektoren).

Eindimensionale Arrays

PHP unterstützt sowohl skalare als auch assoziative Arrays. Letztendlich gibt es keinen Unterschied zwischen den beiden. Sie können ein Array erzeugen, indem sie die [list\(\)](#) oder [array\(\)](#) Funktionen benutzen, oder sie können explizit den Wert eines jeden Array-Elements festlegen.

```
$a[0] = "abc";  
$a[1] = "def";  
$b["foo"] = 13;
```

Sie können ein Array auch erzeugen, indem sie einfach Werte einem Array hinzu fügen. Wenn sie einer Array-Variablen einen Wert weisen, indem sie leere Klammern benutzen, wird der Wert am Ende des Arrays hinzugefügt.

```
$a[] = "hello"; // $a[2] == "hello"  
$a[] = "world"; // $a[3] == "world"
```

Arrays können durch Gebrauch der [asort\(\)](#), [arsort\(\)](#), [ksort\(\)](#), [rsort\(\)](#), [sort\(\)](#), [uasort\(\)](#), [usort\(\)](#), und [uksort\(\)](#)-Funktionen sortiert werden (in Abhängigkeit von der gewünschten Sortierung).

Sie können die Anzahl der Elemente eines Arrays mittels der [count\(\)](#)-Funktion ermitteln.

Durchlaufen können sie ein Array per [next\(\)](#) und [prev\(\)](#)-Funktion. Eine weitere Möglichkeit ist die Funktion [each\(\)](#).

Mehrdimensionale Arrays

Mehrdimensionale Arrays sind einfach. Für jede Dimension des Arrays fügen sie einen anderen [key]-Wert ans Ende an:

```

$a[1]      = $f;           # Ein-Dimensionales Beispiel
$a["foo"]  = $f;

$a[1][0]   = $f;           # Zwei-Dimensional
$a["foo"][2] = $f;         # (sie können numerische und
                           assoziative Indizes so mischen)
$a[3]["bar"] = $f;         # (oder so)

$a["foo"][4]["bar"][0] = $f; # Vier-Dimensional!

```

In PHP ist es nicht möglich, mehrdimensionale Arrays direkt in Strings zu referenzieren. Deshalb erzeugt auch das nachfolgende Beispiel nicht das gewünschte Ergebnis:

```

$a[3]['bar'] = 'Bob';
echo "Dies funktioniert nicht: $a[3][bar]";

```

In PHP wird das o.a. Beispiel folgendes ausgeben: Dies funktioniert nicht: Array[bar] . Jedoch kann der String-Verbindungs-Operator benutzt werden, dies zu umgehen:

```

$a[3]['bar'] = 'Bob';
echo "Das funktioniert: " . $a[3][bar];

```

In PHP4 kann das Problem umgangen werden, indem die Array-Referenz in geschweiften Klammern eingeschlossen wird:

```

$a[3]['bar'] = 'Bob';
echo "Das funktioniert in PHP4: {$a[3][bar]}";

```

Man kann multi-dimensionale Arrays auf viele Arten füllen, aber der trickreichste Weg geht über das Verständnis, wie der [array\(\)](#)-Befehl für assoziative Arrays zu benutzen ist. Die folgenden Code-Schnipsel füllen das ein-dimensionale Array über den gleichen Weg:

Example 1:

```
$a["color"]      = "red";
$a["taste"]      = "sweet";
$a["shape"]      = "round";
$a["name"]       = "apple";
$a[3]            = 4;
```

Example 2:

```
$a = array(
    "color" => "red",
    "taste" => "sweet",
    "shape" => "round",
    "name"  => "apple",
    3       => 4
);
```

Die [array\(\)](#)-Funktion kann eingefügt werden für mehrdimensionale Arrays:

```
<?php
$a = array(
    "apple" => array(
        "color" => "red",
        "taste" => "sweet",
        "shape" => "round"
    ),
    "orange" => array(
        "color" => "orange",
        "taste" => "tart",
        "shape" => "round"
    ),
    "banana" => array(
        "color" => "yellow",
        "taste" => "paste-y",
        "shape" => "banana-shaped"
    )
);

echo $a["apple"]["taste"];    # dies wird "sweet" ausgegeben
?>
```


Objects

Objekt-Initialisierung

Um ein Objekt zu initialisieren benutzen sie die Angabe `new`, dadurch wird das Objekt einer Variablen-Instanz zugewiesen.

```
<?php
class foo {
    function do_foo () {
        echo "Doing foo.";
    }
}

$bar = new foo;
$bar->do_foo ();
?>
```

Für eine ausführliche Besprechung lesen Sie im Kapitel [Klassen und Objekte](#) nach.

Typen-Tricks

PHP erfordert (bzw. unterstützt) keine explizite Typ-Definitionen bei der Deklaration von Variablen; der Typ einer Variablen wird bestimmt durch den Zusammenhang in dem die Variable benutzt wird. Das bedeutet, dass bei der Zuweisung einer Zeichenkette / eines Strings zu einer Variablen *var* diese Variable *var* den Typ String erhält. Sollten sie danach der Variablen *var* einen Integer-Wert zuweisen, wird sie zu einem Integer-Typ.

Ein Beispiel für die automatische Typ-Konvertierung von PHP ist der Plus-Operator '+'. Ist einer der zu addierenden Werte vom Typ double, werden alle Werte als double-Typ gehandhabt. Auch das Ergebnis der Addition wird vom Typ double sein. Ist dies nicht der Fall, werden Werte als Integer-Typen angesehen und das Ergebnis wird ebenfalls vom Typ Integer sein. Beachten sie, dass hierdurch nicht der Typ der Additions-Elemente selbst beeinflusst wird; der Unterschied liegt einzig und allein in der Auswertung dieser Elemente.

```
$foo = "0"; // $foo ist vom Typ String (ASCII 48)
$foo++;    // $foo ist immer noch vom Typ String,
           // Inhalt "1" (ASCII 49)
$foo += 1; // $foo ist jetzt vom Typ Integer (2)
$foo = $foo + 1.3; // $foo ist nun vom Typ double (3.3)
$foo = 5 + "10 Little Piggies"; // $foo ist vom Typ Integer (15)
$foo = 5 + "10 Small Pigs";     // $foo ist vom Typ Integer (15)
```

Wenn für sie die letzten beiden Beispiele gleich sind, beachten sie [String-Umwandlung](#).

Wenn sie die Auswertung einer Variablen als ein spezieller Typ erzwingen wollen, beachten sie den Abschnitt in [Typ-Umwandlung](#). Wenn sie den Typ einer Variablen wechseln wollen, sehen sie bitte [settype\(\)](#) nach.

Um die Beispiele dieses Kapitels auszuprobieren, können sie diese per "Copy und Paste" übertragen und die folgende Zeile einfügen, um zu sehen, was passiert:

```
echo "\$foo==\$foo; ist vom Typ " . gettype( $foo ) . "<br>\n";
```

Anmerkung: Das Verhalten einer automatischen Umwandlung ist zur Zeit undefiniert.

```
$a = 1;           // $a ist ein Integer-Wert
$a[0] = "f";     // $a wird zum Array, $a[0] enthält "f"
```

Weil das o.a. Beispiel so aussieht, als ob \$a ein Array wird, dessen erstes Element ein 'f' enthält, beachten sie folgendes:

```
$a = "1";        // $a ist ein String
$a[0] = "f";     // Was ist mit den String-Offsets? Was passiert?
```

Weil PHP die Indizierung innerhalb von Strings mittels der gleichen Syntax wie bei der Array-Indizierung unterstützt, führt das o.a. Beispiel zu einem Problem: Soll \$a zu einem Array werden, dessen erstes Element ein "f" ist oder soll "f" das erste Zeichen des Strings \$a werden?

Aus diesem Grund ist das Ergebnis der automatischen Umwandlung seit PHP 3.0.12 und auch in PHP 4.0b3-RC4 undefiniert. Lösungen werden noch diskutiert.

Typ-Umwandlung

Typ-Umwandlung in PHP funktioniert vielfach wie in C: Der Name des geforderten Typs wird vor der umzuwandelnden Variablen in Klammern gesetzt.

```
$foo = 10;       // $foo ist ein Integer-Wert
$bar = (double) $foo; // $bar ist vom Typ double
```

Folgende Umwandlungen sind möglich:

- (int), (integer) - Umwandlung in Integer-Wert
- (real), (double), (float) - hin zu double
- (string) - hin zu String
- (array) - hin zum Array
- (object) - Wandlung zum Objekt

Beachten sie, dass Tabulatoren und Leerzeichen innerhalb der Klammern erlaubt sind. Deshalb sind die folgenden Beispiel identisch:

```
$foo = (int) $bar;
$foo = ( int ) $bar;
```

Es ist nicht immer offenkundig, was bei der Typ-Umwandlung geschieht. Zum besseren Verständnis sollte das Folgende beachtet werden:

Bei der Umwandlung einer skalaren oder String-Variablen wird die Variable das erste Element des Arrays:

```
$var = 'ciao';  
$arr = (array) $var;  
echo $arr[0]; // gibt 'ciao' aus
```

Sobald eine skalare oder String-Variable in ein Objekt gewandelt wird, wird die Variable zu einem Attribut des Objekts; der Eigenschafts-Name wird 'scalar':

```
$var = 'ciao';  
$obj = (object) $var;  
echo $obj->scalar; // gibt 'ciao' aus
```

[Zurück](#)[Objects](#)[Anfang](#)[Hoch](#)[Vor](#)[Variablen](#)

Kapitel 7 Variablen

Inhaltsverzeichnis

[Grundlegendes](#)[Vordefinierte Variablen](#)[Geltungsbereich von Variable](#)[Variable Variablen](#)[Variablen ausserhalb von PHP](#)

Grundlegendes

Variablen werden in PHP dargestellt durch ein Dollar-Zeichen (\$) gefolgt vom Namen der Variablen. Bei Variablen-Namen wird zwischen Groß- und Kleinschreibung unterschieden (case-sensitive).

Variablen-Namen werden in PHP nach den gleichen Regeln wie die andere Bezeichner erstellt. Ein gültiger Variablen-Name beginnt mit einem Buchstaben oder einem Unterstrich ("_"), gefolgt von einer beliebigen Anzahl von Buchstaben, Zahlen oder Unterstrichen. Als regulärer Ausdruck (regular expression) würde das wie folgt ausgedrückt: '[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\xzf-\xff]*'.

Anmerkung: Unserem Zweck entspricht also ein Buchstabe von a bis z bzw. A bis Z oder einem ASCII-Zeichen von 127 bis 255 (0x7f bis 0xff).

```
$var = "Du";  
$vaR = "und";  
$Var = "ich";  
$vAr = "wir lernen PHP"  
echo "$var $vaR $Var, $vAr"; // gibt "Du und ich, wir lernen PHP" aus  
  
$4site = 'nicht jetzt'; // ungültig, da Anfang eine Zahl  
$_4site = 'nicht jetzt'; // gültig, da Unterstrich am Anfang  
$täbyte = 'irgendwas'; // gültig, da 'ä' dem ASCII-Wert 228 entspricht
```

Variablen werden in PHP3 durch ihren Wert bestimmt. Das heisst, wenn sie einer Variablen einen Ausdruck zuweisen, wird der gesamte Inhalt des Originalausdrucks in die Zielvariable kopiert. Die Folge ist, dass eine Variable, die ihren Inhalt von einer anderen Variablen erhalten hat, ihren Inhalt behält, auch wenn sie danach den Inhalt der anderen (Quell- / Ursprungs-)Variablen ändern. Die Inhalte der Ziel- und Quellvariablen sind also insoweit unabhängig voneinander. Für weitere Informationen lesen sie bitte [Expressions / Ausdrücke](#).

PHP4 bietet eine andere Möglichkeit der Wertzuweisung bei Variablen: *Zuweisung durch Referenzierung*. Das bedeutet, dass der Wert der neuen Variablen eine Referenz zur Ursprungs-Variablen darstellt (mit anderen Worten: Der Wert ist ein Alias bzw. Zeiger auf den Inhalt der Ursprungsvariablen). Beide Variablen zeigen also

auf die selbe(n) Speicherstelle(n). Änderungen der neuen Variablen ändern auch deren Ursprungs- Variable und umgekehrt. Der Wert / Inhalt wird also nicht kopiert. Die Übertragung geschieht dadurch auch schneller als in PHP3. Dies wird sich aber nur bei umfangreichen Schleifen oder bei der Übertragung von grossen Arrays oder Objekten bemerkbar machen.

Für die Zuweisung per Referenz müssen sie lediglich ein "&" plus "amp;" (&) der (Ausgangs-, Quell-) Variablen voranstellen, die sie einer anderen Variablen zuweisen wollen. Der folgende Skript- Ausschnitt wird zweimal 'Mein Name ist Bob' ausgegeben:

```
<?php
$foo = 'Bob';           // 'Bob' der Variablen $foo zuweisen.
$bar = &$foo;           // Zeiger auf $foo in $bar erzeugen.
$bar = "My name is $bar"; // $bar verändern...
echo $foo;              // $foo wurde dadurch ebenfalls verändert.
echo $bar;
?>
```

Zu beachten ist, dass nur Variablenbezeichner referenziert werden können.

```
<?php
$foo = 25;
$bar = &$foo;           // Gültige Zuweisung.
$bar = &(24 * 7);        // Ungültig, da kein Variablenbezeichner
                        zugewiesen wird.

function test() {
    return 25;
}

$bar = &test();          // Ungültig.
?>
```

[Zurück](#)

Typen-Tricks

[Anfang](#)[Hoch](#)[Vor](#)

Vordefinierte Variablen

Vordefinierte Variablen

PHP bietet jedem ausgeführtem Skript eine Vielzahl von vordefinierten Variablen an. Viele dieser Variablen können jedoch nicht vollständig erläutert werden, da sie abhängig sind vom Web-Server, der Version und dem Setup des Web-Servers sowie weiteren Faktoren. Einige dieser Variablen stehen nicht zur Verfügung, wenn PHP-Skripte per Kommando- Zeilen-Aufruf ausgeführt werden.

Ungeachtet dieser Faktoren folgt nun eine Liste von vordefinierten Variablen innerhalb einer Basis-PHP3- Installation als Modul und einer Basis-Installation des [Apache](#)-Web-Servers in der Vers. 1.3.6.

Eine Liste aller vordefinierten Variablen (und weitere nützliche Informationen) erhalten sie durch Lesen der Infos zu (und Gebrauch) der [phpinfo\(\)](#)-Funktion.

Anmerkung: Diese Liste ist weder vollständig noch erhebt sie Anspruch auf Vollständigkeit. Sie dient lediglich als Anleitung zu den möglicherweise zu erwartenden Arten von vordefinierten Variablen und dem Zugriff darauf.

Apache-Variablen

Diese Variablen werden durch den [Apache](#)-Web-Server erzeugt. Sollten sie einen anderen Web-Server nutzen, gibt es keine Garantie, dass dieser die selben Variablen unterstützt. Es könnten einige sein; es könnten aber auch hier nicht aufgeführte sein. Viele dieser Variablen werden in den [CGI 1.1 Spezifikationen](#) aufgeführt. Darauf sollten sie vorbereitet sein.

Einige sind beim Aufruf von PHP per Kommandozeile verfügbar. Sie können unterschiedliche Bedeutungen haben.

GATEWAY_INTERFACE

Die Revision der CGI-Spezifikation, die der Web-Server benutzt, z.B. 'CGI/1.1'.

SERVER_NAME

Der Host-Name des Web-Servers, der das Skript ausführt. Das kann auch der Name eines virtuellen Servers sein.

SERVER_SOFTWARE

Der Identifikations-String des Web-Servers (aus den Headern), sobald er Anforderungen beantwortet.

SERVER_PROTOCOL

Name und Revision des Informations-Protokolls, über das die Seite angefordert wurde, z.B.

'HTTP/1.0'.

REQUEST_METHOD

Welche Methode zum Zugriff auf die Seite benutzt wurde, z.B. 'GET', 'HEAD', 'POST', 'PUT'.

QUERY_STRING

Der Abfrage-(Query-)String (falls vorhanden), mit dem auf die Seite zugegriffen wurde.

DOCUMENT_ROOT

Das Verzeichnis des gerade ausgeführten Skripts aus Sicht des in der Konfigurations-Datei des Servers definierten Dokument-Wurzel-Verzeichnisses.

HTTP_ACCEPT

Inhalt des `Accept` :-Headers der aktuellen Anforderung (so es einen solche Header gibt).

HTTP_ACCEPT_CHARSET

Inhalt des `Accept-Charset` :-Headers der aktuellen Anforderung (sofern es einen gibt).
Beispiel: 'iso-8859-1,*,utf-8'.

HTTP_ENCODING

Inhalt des `Accept-Encoding` :-Headers der aktuellen Anforderung (wenn es einen gibt).
Beispiel: 'gzip'.

HTTP_ACCEPT_LANGUAGE

Inhalt des `Accept-Language` :-Headers in der aktuellen Anforderung, sofern dieser Header existiert. Beispiel: 'en'.

HTTP_CONNECTION

Inhalt des `Connection` :-Headers des aktuellen Request, so er vorhanden ist. Beispiel: 'Keep-Alive'.

HTTP_HOST

Inhalt des `Host` :-Headers der aktuellen Anforderung, wenn er existiert.

HTTP_REFERER

Die Adresse (URL) - sofern vorhanden - der Seite, von der aus auf die aktuelle Seite gesprungen wurde. Diese wird vom Browser gesetzt. Nicht alle unterstützen dies.

HTTP_USER_AGENT

Inhalt der `User-Agent` :-Header-Angabe der aktuellen Anfrage (wenn eine Angabe existiert). Dabei handelt es sich um eine Zeichenkette, welche das Browser- Programm benennt, mit dem die aktuelle Seite aufgerufen wurde, z.B. `Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586)`. Abgesehen von anderen Dingen können sie diesen Wert zusammen mit der [get_browser\(\)](#)-Funktion dazu verwenden, ihre Web-Seiten in Abhängigkeit von den Möglichkeiten des jeweils benutzten Browsers anzeigen zu lassen.

REMOTE_ADDR

Die IP-Adresse, von der aus gerade auf die Web-Seite zugegriffen wird.

REMOTE_PORT

Der Port, der zum Zugriff auf ihren Web-Server seitens des Anwender-Computers benutzt wird.

SCRIPT_FILENAME

Der absolute Pfadname des gerade ausgeführten Skripts.

SERVER_ADMIN

Der Inhalt der in der Konfigurations-Datei des Web-Servers (Apache) stehenden Angabe zum SERVER_ADMIN. Bei virtuellen Hosts der dafür eingetragene Wert.

SERVER_PORT

Der vom Server für den Web-Server benutzte Kommunikations- Port (normalerweise '80'). Verwenden sie z.B. SSL, wird dieser Port derjenige sein, den sie für sicheres HTTP definiert haben.

SERVER_SIGNATURE

Sofern eingeschaltet, enthält diese Zeichenkette die Server- Version und den virtuellen Host-Namen. Sie wird den durch den Server generierten Seiten hinzu gefügt.

PATH_TRANSLATED

In Abhängigkeit vom Datei- / File-System der Pfad des aktuellen Skripts, nachdem der Server das virtuelle Mapping in ein reales Mapping umgesetzt hat (nicht der Dokument-Wurzel-Pfad).

SCRIPT_NAME

Enthält den Pfad des aktuellen Skripts. Nützlich für Seiten, die auf sich selbst verweisen müssen.

REQUEST_URI

Die URI, die durch den Zugriff auf die aktuelle Seite gegeben ist, z.B. '/index.html'.

Umgebungs- / Environment-Variablen

Diese Variablen werden aus der Umgebung, in der PHP läuft, in den globalen Namensbereich von PHP importiert. Viele werden durch die jeweilige Shell, in der PHP läuft, unterstützt bzw. gebildet. Da es verschiedenste Systemumgebungen mit den unterschiedlichsten Shell`s gibt, ist es nicht möglich, eine abschließende Liste der definierten Umgebungs-Variablen aufzustellen. Lesen sie deshalb in der Anleitung zu ihrer Shell nach, um eine Liste dieser systembezogenen Variablen zu erhalten.

Andere Umgebungs-Variablen beinhalten die CGI-Variablen, die ohne Rücksicht darauf, ob PHP als Web-Server-Modul oder im CGI-Modus läuft, gesetzt werden.

PHP-Variablen

Diese Variablen werden durch PHP selbst erzeugt.

argv

Ein Array von Informationen, die dem Skript übergeben werden. Wird das Skript an der Befehlszeile aufgerufen, ermöglicht dies C-ähnlichen Zugriff auf die Kommando-Zeilen-Parameter. Beim Aufruf per GET-Methode enthält dieses Array die Abfragewerte.

argc

Anzahl der per Kommando-Zeile dem Skript übergebenen Parameter (wenn von dort aufgerufen).

PHP_SELF

Der Datei-Name des gerade ausgeführten Skripts, relativ zum Wurzel-Verzeichnis des Dokuments. Bei Kommando-Zeilen- Aufrufen ist diese Variable nicht verfügbar.

HTTP_COOKIE_VARS

Ein assoziatives Array von Variablen, das dem aktuellen Skript über HTTP-Cookies übergeben wurde. Nur verfügbar, wenn die Option "Variablen-Tracking" per [track_vars](#)-Konfigurations- oder per `<?php_track_vars?>`- Anweisung eingeschaltet wurde.

HTTP_GET_VARS

Ein assoziatives Array von Variablen, das dem aktuellen Skript per HTTP-GET-Methode übergeben wurde. Nur möglich, wenn "Variablen-Tracking" per [track_vars](#)-Konfigurations- oder per `<?php_track_vars?>`- Anweisung eingeschaltet ist.

HTTP_POST_VARS

Ein assoziatives Array aus Variablen, welches dem aktuellen Skript per HTTP-POST-Methode übergeben wurde. Nur möglich, wenn "Variablen-Tracking" per [track_vars](#)-Konfigurations- oder per `<?php_track_vars?>`- Anweisung eingeschaltet ist.

[Zurück](#)

Variablen

[Anfang](#)

[Hoch](#)

[Vor](#)

Geltungsbereich von Variable

Geltungsbereich von Variable

Der Geltungsbereich einer Variablen ergibt sich aus dem Zusammenhang, in dem sie definiert wurde. Meistens besteht dieser aus einem einzigen Bereich. Dieser beinhaltet auch den Bereich für Dateien, die per "Include"- oder "Require"-Anweisung eingebunden wurden, z.B.:

```
$a = 1;  
include "b.inc";
```

Die Variable \$a ist auch in der eingebundenen Datei b.inc verfügbar. In benutzer-definierten Funktionen wird ein auf die Funktion beschränkter Geltungsbereich eingeführt. Jede in einer Funktion benutzte Variable ist zunächst auf den lokalen Bereich der Funktion beschränkt, z.B.:

```
$a = 1; /* globaler Bereich */  
  
function test () {  
    echo $a; /* Referenz auf einen lokalen Variablen-Bereich */  
}  
  
test ();
```

Dieses Skript erzeugt keine Bildschirm-Ausgabe, da sich die Echo- Anweisung auf eine lokale Variable namens \$a bezieht und dieser kein Wert im lokalen Bezug zugewiesen worden ist. Dies ist ein kleiner Unterschied zu C, wo globale Variablen auch in Funktionen vorhanden sind, es sei denn, sie werden durch eine funktionsinterne Definition überschrieben. Das kann zu Problemen führen, denn in PHP müssen global geltende Variablen innerhalb von Funktionen als solche definiert werden. Ein Beispiel:

```

$a = 1;
$b = 2;

function sum () {
    global $a, $b;
    $b = $a + $b;
}

sum ();
echo $b;

```

Das obige Skript gibt "3" aus. Durch das Deklarieren der Variablen \$a und \$b innerhalb der Funktion, weisen alle Referenzen zu beiden Variablen auf die nun globalen Werte. Es gibt keine Beschränkungen bei der Anzahl an globalen Variablen, die durch eine Funktion verändert werden können.

Eine weitere Möglichkeit besteht in der Verwendung des speziellen \$GLOBALS PHP-Array. Das obige Beispiel kann damit auch so geschrieben werden:

```

$a = 1;
$b = 2;

function sum () {
    $GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"];
}

sum ();
echo $b;

```

Das \$GLOBALS-Array ist ein assoziatives Array mit dem Bezeichner der globalen Variablen als Schlüssel und dem Inhalt dieser Variablen als Wert des Array-Elements.

Ein weiterer wichtiger Anwendungszweck von Variablen-Bereichen ist die *static*-Variable. Eine statische- Variable existiert nur in einem lokalen Funktions-Bereich, der Wert geht beim Verlassen dieses Bereichs aber nicht verloren. Schauen sie das folgende Beispiel an:

```

function test () {
    $a = 0;
    echo $a;
    $a++;
}

```

Diese Funktion ist sinnlos, da sie bei jedem Aufruf \$a auf 0 setzt und "0" ausgibt. Die Anweisung \$a++, welche den Wert erhöht, macht keinen Sinn, da der Wert von \$a beim Verlassen der Funktion verloren geht. Um eine sinnvolle Zählfunktion zu implementieren, die ihren aktuell gesetzten Wert nicht vergisst, müssen sie die \$a-Variable als "static" deklarieren:

```
function test () {  
    static $a = 0;  
    echo $a;  
    $a++;  
}
```

Jetzt wird bei jedem Aufruf der test()-Funktion der aktuelle Wert von \$a ausgegeben und dann um 1 erhöht.

Static-Variablen ermöglichen auch einen Weg des Umgangs mit rekursiven Funktionen. Das sind Funktionen, die sich selbst aufrufen. Hierbei besteht die Gefahr, so genannte Endlos- Schleifen zu programmieren. Sie müssen also einen Weg vorsehen, diese Rekursion zu beenden. Die folgende einfache Funktion zählt rekursiv bis 10. Die Variable \$count wird benutzt, um die Rekursion zu beenden:

```
function test () {  
    static $count = 0;  
  
    $count++;  
    echo $count;  
    if ($count < 10) {  
        test ();  
    }  
    $count--;  
}
```

[Zurück](#)

Vordefinierte Variablen

[Anfang](#)[Hoch](#)[Vor](#)

Variable Variablen

Variable Variablen

Manchmal ist es erforderlich, variable Variablen-Bezeichner zu benutzen. Das bedeutet, einen Variablen-Namen zu setzen und dynamisch zu gebrauchen. Eine normale Variable wird wie folgt gebildet:

```
$a = "hallo";
```

Eine variable Variable nimmt den Wert einer Variablen und behandelt ihn als Bezeichner der Variablen. Im obigen Beispiel kann *hallo* als Variablen-Name gebraucht werden, indem man zwei \$-Zeichen benutzt, also schreibt:

```
$$a = "welt";
```

Nun existieren in der PHP-Symbol-Struktur zwei definierte und gespeicherte Variablen: \$a mit dem Inhalt "Hallo" und \$hello mit dem Inhalt "Welt". Deshalb wird die Anweisung

```
echo "$a ${$a}";
```

zur gleichen Ausgabe führen wie:

```
echo "$a $hallo";
```

also zu: *hallo Welt*.

Wenn sie variable Variablen mit Arrays verwenden, müssen sie eine Unklarheit beachten. Wenn sie nämlich \$\$a[1] schreiben, dann muss der Parser wissen, ob sie \$a[1] als Variable oder \$\$a als Variable und dann [1] als Index dieser Variablen verwenden wollen bzw. gemeint haben. Die Lösung: Verwenden sie \${\$a[1]} für den ersten Fall und \${\$a}[1] für den zweiten.

Variablen ausserhalb von PHP

HTML-Formulare (GET and POST)

Sobald ein Formular an ein PHP-Skript übergeben wird, wird jede Variable dieses Formulars dem Skript automatisch verfügbar gemacht. Dafür sorgt PHP. Ein Beispiel:

Beispiel 7-1 Einfache Formular-Variablen

```
<form action="foo.php3" method="post">  
  Name: <input type="text" name="name"><br>  
  <input type="submit">  
</form>
```

Bei der Formular-Übergabe (hier per POST) erzeugt PHP die Variable `$name`, die genau das enthält, was sie in das *Name*:-Feld des Formulars eingegeben haben.

Im Zusammenhang mit Formular-Variablen versteht PHP auch Arrays, aber nur eindimensionale. Deshalb müssen sie die betreffenden Variablen vorher gruppieren oder das folgende Beispiel auf Mehrfach-Auswahl-Bereiche anwenden:

Beispiel 7-2 Komplexere Formular-Variablen

```
<form action="array.php" method="post">  
  Name: <input type="text" name="personal[name]"><br>  
  Email: <input type="text" name="personal[email]"><br>  
  Bier: <br>  
  <select multiple name="bier[]">  
    <option value="binding">Binding  
    <option value="warsteiner">Warsteiner  
    <option value="stauder">Stauder  
    <option value="stuttgarter">Stuttgarter Schwabenbräu  
  </select>  
  <input type="submit">  
</form>
```

Ist das PHP-"`track_vars`"-Feature eingeschaltet (entweder durch [track_vars](#)-Konfigurations- Einstellung oder per `<?php_track_vars?>` -Anweisung), können die durch die POST- oder GET-Methode übertragenen Variablen in den globalen assoziativen Arrays `$HTTP_POST_VARS` und

\$HTTP_GET_VARS wieder gefunden werden.

IMAGE SUBMIT Variablen-Bezeichner

Zur Übertragung eines Formulars kann auch ein Bild (Image) statt eines Übertragungs-Schalters (Submit-Button) benutzt werden, dessen Tag wie folgt aussieht:

```
<input type=image src="image.gif" name="sub">
```

Klickt der Benutzer irgendwo auf das Bild, wird das entsprechende Formular an den Web-Server übertragen. Hierbei sind zwei zusätzliche Variablen vorhanden, sub_x und sub_y. Diese enthalten die Koordinaten des Klick-Punktes innerhalb des Bildes. Die Erfahreneren werden sagen, dass dabei kein Unterstrich vorhanden ist (da ein Punkt enthalten ist, also z.B. "sub.x"). Dieser wird aber automatisch von PHP erzeugt.

HTTP-Cookies

PHP unterstützt HTTP-Cookies, wie sie in [Netscape's Spec](#) definiert sind. Cookies ermöglichen die Daten-Speicherung innerhalb der jeweiligen Browser-Umgebung zur Weiterleitung oder wiederholten Identifikation von Benutzern. Sie können Cookies erzeugen, indem sie die **SetCookie()**- Funktion benutzen. Cookies sind Teil des HTTP-Headers, deshalb muss die SetCookie-Funktion aufgerufen werden, bevor irgendeine Ausgabe an den Browser gesendet wird. Dabei handelt es sich um die gleiche Einschränkung, die auch für die **Header()**-Funktion gilt. Alle an sie gesendeten Cookies werden in PHP-Variablen umgesetzt, genau wie die Daten bei den GET- und POST-Methoden.

Wenn sie einem einzelnen Cookie verschiedene Wert zuweisen wollen müssen sie dem Cookie-Namen [] hinzufügen. Z.B.:

```
SetCookie ("MeinCookie[]", "Ich teste", time()+3600);
```

Bedenken sie, dass ein Cookie ein vorhergehendes Cookie gleichen namens überschreibt, es sei denn, der Pfad oder die Domain ist anders. Für eine Warenkorb-Anwendung müssen sie deshalb z.B. einen Zähler bilden und diesen weiter leiten:

Beispiel 7-3 SetCookie-Beispiel

```
$Count++;  
SetCookie ("Count", $Count, time()+3600);  
SetCookie ("Cart[$Count]", $item, time()+3600);
```

Umgebungs- / Environment-Variablen

PHP sorgt automatisch für die Verfügbarkeit der Umgebungs- Variablen als normale PHP-Variablen.

```
echo $HOME; /* Zeigt die HOME-Umgebungs-Variable (sofern gesetzt). */
```

Da Informationen per GET, POST und Cookie-Mechanismen übergeben werden, ist es manchmal das Beste, Umgebungs-Variablen explizit auszulesen. Dadurch wird die richtige Version eingelesen. Hierfür kann die [getenv\(\)](#)-Funktion genutzt werden. Den Wert einer Umgebungs-Variablen können sie per [putenv\(\)](#)- Funktion setzen.

Punkte in eingelesenen Variablen-Bezeichnern

Normalerweise verändert PHP nicht die Variablen-Bezeichner, wenn sie einem Skript übergeben werden. Es sollte aber beachtet werden, dass der Punkt (".") kein gültiger Bestandteil eines Variablen-Bezeichners ist. Deshalb achten sie auf folgendes:

```
$varname.ext; /* ungültiger Variablen-Bezeichner */
```

Der PHP-Parser sieht eine Variable namens \$varname, gefolgt von einem Zeichenketten-Verbindungs-Operator, dieser wiederum gefolgt von der offenen Zeichenkette 'ext' (also nicht eingegrenzt durch "" und auch keinem Schlüssel oder reserviertem Bezeichner entsprechend). Das kann natürlich nicht zum gewünschten Ergebnis führen.

Deshalb ist es wichtig zu wissen, dass PHP in den ihm übergebenen Variablen alle Punkte (.) automatisch durch einen Unterstrich (_) ersetzt.

Bestimmung des Variablen-Typs

Da PHP den Typ einer Variablen bestimmt und grundsätzlich selbst eine entsprechende Umformung vornimmt, ist es nicht immer klar, welchen Typ eine Variable gerade hat. PHP beinhaltet einige Funktionen, die dies herausfinden. Als da sind: [gettype\(\)](#), [is_long\(\)](#), [is_double\(\)](#), [is_string\(\)](#), [is_array\(\)](#) und [is_object\(\)](#).

[Zurück](#)

Variable Variablen

[Anfang](#)[Hoch](#)[Vor](#)

Konstanten

Kapitel 8 Konstanten

PHP definiert eine Reihe von Konstanten und stellt einen Mechanismus zur Verfügung, mit dem man zusätzliche Konstanten zur Laufzeit definieren kann. Konstanten sind Variablen sehr ähnlich, bis auf die Tatsache, dass sie mit der [define\(\)](#)-Funktion definiert werden müssen und später nicht mehr mit einem anderen Wert versehen werden können.

Folgende vordefinierten Variablen sind immer verfügbar:

`__FILE__`

Der Name der Skript-Datei, die gerade geparsed wird. Wird diese Konstante in einer Datei verwendet, die per [include\(\)](#) oder [require\(\)](#) eingebunden wurde, liefert sie den Namen der eingebundenen Datei, nicht den der aufrufenden Datei.

`__LINE__`

Die Nummer der Zeile im laufenden Skript, die gerade geparkt wird. Wird diese Konstante in einer Datei benutzt, die per [include\(\)](#) oder [require\(\)](#) eingebunden wurde, liefert sie die Zeilennummer innerhalb der eingebundenen Datei.

`PHP_VERSION`

Ein String, der die Versionsnummer des PHP-Parsers enthält, der gerade verwendet wird; z. B. `'3.0.8-dev'`.

`PHP_OS`

Der Name des Betriebssystems, auf dem der PHP-Parser ausgeführt wird; z. B. `'Linux'`.

`TRUE`

Der Wert `'wahr'`.

`FALSE`

Der Wert `'falsch'`.

`E_ERROR`

Bedeutet einen Fehler, der sich von einem `'parsing error'` unterscheidet. Die Ausführung des Skriptes wird beendet.

`E_WARNING`

Gibt einen Zustand zurück, durch den PHP weiß, dass etwas nicht in Ordnung ist, das aktuelle Skript aber trotzdem weiter ausführt; dies kann vom Skript selbst aufgefangen werden. Ein Beispiel wäre ein ungültiger regulärer Ausdruck (regex) in der Funktion [ereg\(\)](#).

`E_PARSE`

Der Parser hat Probleme mit ungültiger Syntax in der Skript-Datei. Die Ausführung des Skriptes wird

beendet.

E_NOTICE

Etwas ist aufgetreten, das ein Fehler sein kann oder nicht. Das aktuelle Skript wird weiter ausgeführt. Beispiele hierfür sind ein nicht gequoteter string als Hash-Index oder der Zugriff auf eine Variable, die nicht gesetzt wurde.

Die E_*-Konstanten werden typischerweise mit der [error_reporting\(\)](#)-Funktion benutzt, um das Fehlermeldungs-Niveau festzusetzen.

Zusätzliche Konstanten können mithilfe der [define\(\)](#)-Funktion definiert werden.

Zu beachten ist, dass dies Konstanten sind, und keine Makros, wie man sie von C her kennt; nur gültige Skalar-Daten können von einer Konstante vertreten werden.

Beispiel 8-1 Konstanten definieren

```
<?php
define("CONSTANT", "Hallo Welt.");
echo CONSTANT; // gibt "Hallo Welt." aus.
?>
```

Beispiel 8-2 Die Benutzung von __FILE__ und __LINE__

```
<?php
function report_error($file, $line, $message) {
    echo "Ein Fehler ist aufgetreten in $file in Zeile $line: $message.";
}

report_error(__FILE__, __LINE__, "Irgendetwas ist falsch gelaufen!");
?>
```

[Zurück](#)

Variablen ausserhalb von PHP

[Anfang](#)[Hoch](#)[Vor](#)

Ausdrücke

Kapitel 9 Ausdrücke

Ausdrücke (Expressions) sind die wichtigsten Bausteine von PHP. In PHP ist fast alles, was geschrieben wird, ein Ausdruck. Die einfachste, aber auch zutreffenste Definition für einen Ausdruck ist "alles, was einen Wert hat".

Die grundlegendste Formen von Ausdrücken sind Konstanten und Variablen. Wenn man "\$a = 5" schreibt, weist man \$a den Ausdruck '5' zu. '5' hat offensichtlich den Wert 5. Anders ausgedrückt: '5' ist ein Ausdruck mit dem Wert 5 (in diesem Fall ist '5' eine Integer-Konstante).

Nach dieser Zuweisung würde man erwarten, dass der Wert von \$a nun ebenfalls 5 ist, wenn man also \$b = \$a schreibt, sollte dasselbe dabei herauskommen, als hätte man \$b = 5 geschrieben. Anders ausgedrückt: \$a wäre ebenfalls ein Ausdruck mit dem Wert 5. Wenn alles richtig funktioniert, wird genau das passieren.

Etwas kompliziertere Beispiele für Ausdrücke sind Funktionen:

```
function foo () {  
    return 5;  
}
```

Angenommen, Sie sind mit dem Konzept von Funktionen vertraut (wenn Sie es nicht sind, lesen Sie das Kapitel über Funktionen), dann würden Sie annehmen, dass die Eingabe von \$c = foo() grundsätzlich daselbe bedeutet, als würde man schreiben \$c = 5, und genau das trifft zu. Funktionen sind Ausdrücke mit dem Wert ihres Rückgabewertes. Da foo() den Wert 5 zurückgibt, ist der Wert des Ausdruckes 'foo()' 5. Normalerweise geben Funktionen nicht einfach einen statischen Wert zurück, sondern berechnen irgendetwas.

Natürlich müssen Werte in PHP keine Integer-Zahlen sein, und oft sind sie es auch nicht. PHP unterstützt drei skalare Datentypen: integer values (Integer-Zahlen), floating point values (Fließkommazahlen) und string values (Zeichenketten). (Skalare sind Datentypen, die man nicht in kleinere Stücke 'brechen' kann, im Gegensatz zu Arrays). PHP unterstützt auch zwei zusammengesetzte (nicht-skalare) Datentypen: Arrays und Objekte. Jeder dieser Datentypen kann Variablen zugewiesen und von Funktionen zurückgegeben werden.

Bis hierhin sollten Benutzer von PHP/FI 2 keine Veränderung bemerkt haben. PHP fasst den Begriff 'Ausdruck' aber noch viel weiter, wie es auch andere Programmiersprachen tun. PHP ist in dem Sinne eine ausdrucksorientierte Sprache, dass fast alles ein Ausdruck ist. Zurück zu dem Beispiel, mit dem wir uns schon beschäftigt haben: '\$a = 5'. Es ist einfach zu erkennen, dass hier zwei Werte enthalten sind: Der Wert der Integer-Konstanten '5' und der Wert von \$a, der auf 5 geändert wird. In Wirklichkeit ist aber noch ein zusätzlicher Wert enthalten, nämlich der Wert der Zuweisung selbst. Die Zuweisung selbst enthält den zugewiesenen Wert, in diesem Fall 5. In der Praxis bedeutet dies, dass '\$a = 5', egal was es tut, immer einen Ausdruck mit dem Wert 5 darstellt. Folglich ist '\$b = (\$a = 5)' gleichbedeutend mit '\$a = 5; \$b = 5;' (ein Semikolon markiert das Ende einer Anweisung). Da Wertzuweisungen von rechts nach links geparkt werden, kann man auch '\$b = \$a = 5' schreiben.

Ein anderes gutes Beispiel für die ausdrucksorientierung von PHP sind Prä- und Post-Inkrement sowie die entsprechenden Dekremente. Benutzer von PHP/FI 2 und vielen anderen Sprachen sind vermutlich mit den Notationen 'variable++' und 'variable--' vertraut. Dies sind Inkrement- und Dekrement-Operatoren. IN PHP/FI 2 hat das Statement '\$a++' keinen Wert (es ist kein Ausdruck) und daher kann man es nicht als Wert zuweisen oder in irgendeiner Weise benutzen. PHP erweitert die Eigenschaften von Dekrement und Inkrement, indem es die beiden ebenfalls zu Ausdrücken macht. In PHP gibt es, wie in C, zwei Arten von Inkrementen - Prä-Inkrement und Post-Inkrement. Grundsätzlich erhöhen sowohl Prä- als auch Post-Inkrement den Wert der Variable, und der Effekt auf die Variable ist derselbe. Der Unterschied ist der Wert des Inkrement-Ausdruckes. Das Prä-Inkrement, das '++\$variable' geschrieben wird, enthält als Wert den Wert der erhöhten

Variabe (PHP erhöht den Wert der Variable, bevor es ihren Wert ausliest, daher der Name 'PRÄ-Inkrement'). Das Post-Inkrement, dass '\$variable++' geschrieben wird, enthält dagegen den ursprünglichen Wert der Variablen vor der Erhöhung (PHP erhöht den Wert der Variablen, nachdem es ihren Wert ausgelesen hat, daher der Name 'POST-Inkrement').

Ein sehr gebräuchlicher Typ von Ausdrücken sind Vergleichsausdrücke. Diese Ausdrücke geben entweder 0 (=FALSCH) oder 1 (=WAHR) zurück. PHP unterstützt > (größer), >= (größer oder gleich), == (gleich), != (ungleich), < (kleiner), und <= (kleiner oder gleich). Diese Ausdrücke werden meistens in bedingten Anweisungen, wie z. B. in if-Anweisungen, verwendet.

Im letzten Beispiel für Ausdrücke befassen wir uns mit kombinierten Zuweisungs- und Operator-Ausdrücken. Wir haben schon gezeigt, wie man den Wert von \$a um 1 erhöht, nämlich indem man einfach '\$a++' oder '++\$a' schreibt. Aber was tut man, wenn man den Wert um mehr als eins erhöhen will, z. B. um 3? Man könnte mehrer Male '\$a++' schreiben, aber das ist offensichtlich weder effizient noch sehr komfortabel. Viel üblicher ist es, einfach '\$a = \$a + 3' zu schreiben. '\$a + 3' gibt den Wert von \$a plus 3 zurück, dieser wird wieder \$a zugewiesen, was dazu führt, dass \$a nun um 3 erhöht wurde. In PHP - wie in einigen anderen Programmiersprachen, z. B. in C - kann man dies aber noch kürzer schreiben, was mit der Zeit klarer wird und auch einfacher zu verstehen ist. Um 3 zu dem aktuellen Wert hinzuzufügen, schreibt man '\$a += 3'. Das bedeutet exakt: "Nimm' den Wert von \$a, addiere 3 hinzu und weise \$a den entstandenen Wert zu". Zusätzlich dazu, dass diese Schreibweise kürzer und klarer ist, resultiert sie auch in einer schnelleren Ausführung. Der Wert von '\$a += 3' ist, wie der Wert einer regulären Zuweisung, der zugewiesene Wert. Es ist zu beachten, dass dieser Wert NICHT 3, sondern dem kombinierten Wert von \$a plus 3 entspricht (Das ist der Wert, der \$a zugewiesen wird). Jeder Operator, der zwei Elemente verbindet, kann in dieser Schreibweise verwendet werden, z. B. '\$a -= 5' (vermindert den Wert von \$a um 5) oder '\$a *= 7' (multipliziert den Wert von \$a mit 7 und weist das Ergebnis \$a zu), usw.

Es gibt einen weiteren Ausdruck, der Ihnen vielleicht seltsam vorkommt, wenn Sie ihn bisher noch in keiner Programmiersprache kennengelernt haben, den dreifach konditionalen Operator:

```
$eins ? $zwei : $drei
```

Wenn der Wert des ersten Sub-Ausdruckes (hier: \$eins) wahr ist (d. h. nicht NULL), dann wird der Wert des zweiten Subausdrucks (hier: \$zwei) zurückgeben und ist das Ergebnis des konditionalen Ausdrucks. Andernfalls (d. h. wenn der erste Ausdruck falsch ist), wird der Wert des dritten Subausdruckes (hier: \$drei) zurückgeben.

Das folgende Beispiel sollte das Verständnis von Prä- und Post-Inkrement und von Ausdrücken im allgemeinen erleichtern:

```
function verdoppeln($i) {
    return $i*2;
}
$b = $a = 5;          /* Weise den Variablen $a und $b beiden den Wert 5 zu */
$c = $a++;            /* Post-Inkrement, der urspruengliche Wert von $a (also 5)
                        wird $c zugewiesen. */
$e = $d = ++$b;       /* Prae-Inkrement, der erhöhte Wert von $b (= 6) wird $d und
                        $e zugewiesen. */

/* An diesem Punkt sind $d und $e beide gleich 6 */

$f = verdoppeln($d++); /* Weise $f den doppelten Wert von $d vor
                        der Erhöhung um eins zu, 2*6 = 12 */
$g = double(++$e);     /* Weise $g den doppelten Wert von $e nach
                        der Erhoehung zu, 2*7 = 14 to $g */
$h = $g += 10;         /* Zuerst wie $g um 10 erhöht und hat schliesslich den Wert
                        24. Der Wert dieser Zuweisung (24) wird dann $h zugewiesen,
                        womit $h ebenfalls den Wert von 24 hat. */
```

Am Anfang dieses Kapitels hatten wir gesagt, wir würden die verschiedenen Arten von Anweisungen beschreiben und, wie versprochen, Ausdrücke können Anweisungen sein. Trotzdem ist nicht jeder Ausdruck eine Anweisung. In diesem Fall hat eine Anweisung die Form 'Ausdr' ';', d. h. ein Ausdruck gefolgt von einem Semikolon. In '\$b=\$a=5;' ist \$a=5 ein gültiger Ausdruck, aber für sich allein keine Anweisung. '\$b=\$a=5;' ist jedoch eine gültige Anweisung.

Ein letzter Punkt, der noch zu erwähnen ist, ist der 'wahr'-Wert von Ausdrücken. In vielen Fällen, hauptsächlich in bedingten Anweisungen und Schleifen, ist man nicht am spezifischen Wert eines Ausdrucks interessiert, sondern kümmert sich nur darum, ob er WAHR oder FALSCH bedeutet (PHP hat keinen speziellen boolean-Datentyp). Der Wahrheitswert eines Ausdrucks in in PHP wird ähnlich bestimmt wie in Perl. Jeder numerische Wert, der nicht Null ist, bedeutet WAHR, Null bedeutet FALSCH. Es ist zu beachten, dass negative Werte nicht Null sind und deshalb als WAHR aufgefasst werden! Eine leere Zeichenkette und die Zeichenkette "0" sind FALSCH; alle anderen Zeichenketten sind WAHR. Nicht-skalare Datentypen (Arrays und Objekte) werden als FALSCH betrachtet, wenn sie keine Elemente enthalten, andernfalls geben sie WAHR zurück.

PHP stellt eine vollständige und mächtige Implementat von Ausdrücken bereit und, deren vollständige Dokumentation den Rahmen dieses Manuals sprengen würde. Die obigen Beispiele sollten Ihnen einen guten Eindruck darüber verschaffen, was Ausdrücke sind und wie man nützliche Ausdrücke konstruieren kann. Im Rest dieses Manuals werden wir *ausdr* schreiben, um ausdrücken, dass an dieser Stelle jeder gültige PHP-Ausdruck stehen kann.

[Zurück](#)

Konstanten

[Anfang](#)

[Hoch](#)

[Vor](#)

Operatoren

Kapitel 10 Operatoren

Inhaltsverzeichnis

[Arithmetische Operatoren](#)[Zuweisungsoperatoren](#)[Bit-Operatoren](#)[Vergleichs-Operatoren](#)[Fehler-Kontroll-Operatoren](#)[Operatoren zur Programmausführung](#)[Inkrementierungs- bzw. Dekrementierungsoperatoren](#)[Logische Operatoren](#)[Operator-Rangfolge](#)[Zeichenketten-Operatoren](#)

Arithmetische Operatoren

Erinnerst du dich noch an die Grundrechenarten aus der Schule? Die arithmetischen Operatoren funktionieren genauso:

Tabelle 10-1 Arithmetic Operators

Beispiel	Name	Ergebnis
$\$a + \$b$	Addition	Summe von $\$a$ und $\$b$.
$\$a - \$b$	Subtraktion	Differenz von $\$a$ und $\$b$.
$\$a * \$b$	Multiplikation	Produkt von $\$a$ und $\$b$.
$\$a / \$b$	Division	Quotient von $\$a$ und $\$b$.
$\$a \% \$b$	Modulus	Rest von $\$a$ geteilt durch $\$b$.

Zuweisungsoperatoren

Der einfachste Zuweisungsoperator ist "=". Wahrscheinlich kommt man als erstes auf die Idee, ihn mit "ist gleich" zu bezeichnen. Das ist falsch. In Wirklichkeit bedeutet er, dass dem linken Operanden der Wert des rechten Operanden zugewiesen wird (man müsste ihn also mit "wird gesetzt auf den Wert von" übersetzen).

Der Wert eines Zuweisungs-Ausdruckes ist der zugewiesene Wert. D. h. der Wert des Ausdruckes "\$a = 3" ist 3. Das erlaubt es, einige raffinierte Dinge zu tun:

```
$a = ($b = 4) + 5 // $a ist nun gleich 9 und $b wurde auf den Wert 4 gesetzt.
```

Zusätzlich zu dem oben vorgestellten Zuweisungsoperator "=" gibt es "kombinierte Operatoren" für alle binären, arithmetischen und String-Operatoren, die es erlauben, den Wert einer Variablen in einem Ausdruck zu benutzen, und dieser anschließend das Ergebnis des Ausdrucks als neuen Wert zuzuweisen. Beispiel:

```
$a = 3;  
$a += 5; // setzt $a auf den Wert 8, als ob wir geschrieben haetten: $a = $a + 5;  
$b = "Hallo ";  
$b .= "Du!"; // setzt $b auf den Wert "Hallo Du!", aequivalent zu $b = $b . "There!";
```

Man beachte, dass die Zuweisung nur den Wert der Ursprungsvariable der neuen Variable zuweist (assignment by value)(sie "kopiert"), weshalb sich Änderungen an der einen Variablen nicht auf die andere auswirken werden. Das kann wichtig sein wenn man einen großen Array o. ä. in einer Schleife kopieren muss. PHP4 unterstützt 'assignment by reference' (Zuweisung als Verweis), mithilfe der `$var = &$othervar`; -Syntax, dies funktioniert jedoch nicht in PHP3. 'Assignment by reference' bedeutet, dass beide Variablen nach der Zuweisung die selben Daten repräsentieren. Nach der Zuweisung verändert eine Änderung der Variable `$var` auch den Inhalt der Variable `&$othervar`; , nichts wird kopiert, stattdessen wird ein "Verweis" auf die erste Variable angelegt.

Bit-Operatoren

Bit-Operatoren erlauben es, in einem Integer bestimmte Bits "ein- oder auszuschalten" (auf 0 oder 1 zu setzen).

Tabelle 10-2 Bit-Operatoren

Beispiel	Name	Ergebnis
$\$a \& \$b$	Und	Die Bits sind in \$a und in \$b gesetzt.
$\$a \$b$	Oder	Die Bits sind entweder in \$a oder in \$b gesetzt.
$\$a \wedge \$b$	Entweder oder (Xor)	Die Bits sind in \$a oder in \$b gesetzt, aber nicht in beiden.
$\sim \$a$	Nicht	Die Bits, die in \$a nicht gesetzt sind, sind gesetzt, und umgekehrt.
$\$a \ll \$b$	Nach links verschieben	Verschiebung der Bits von \$a um \$b Stellen nach links (Jede Stelle entspricht einer Multiplikation mit zwei).
$\$a \gg \$b$	Nach rechts verschieben	Verschiebt die Bits von \$a um \$b Stellen nach rechts (jede Stelle entspricht einer Division durch zwei).

Vergleichs-Operatoren

Vergleichs-Operatoren erlauben es - wie der Name schon sagt - zwei Werte zu vergleichen.

Tabelle 10-3 Vergleichsoperatoren

Beispiel	Name	Ergebnis
<code>\$a == \$b</code>	Gleich	Gibt "wahr" zurück, von <code>\$a</code> gleich <code>\$b</code> ist.
<code>\$a === \$b</code>	Identisch	Gibt "wahr" zurück wenn <code>\$a</code> gleich <code>\$b</code> ist und beide vom gleichen Typ sind(funktioniert nur in PHP4)
<code>\$a != \$b</code>	Ungleich	Gibt "wahr" zurück, wenn <code>\$a</code> nicht gleich <code>\$b</code> ist.
<code>\$a < \$b</code>	Kleiner Als	Gibt "wahr" zurück, wenn <code>\$a</code> kleiner als <code>\$b</code> ist.
<code>\$a > \$b</code>	Größer Als	Gibt "wahr" zurück, wenn <code>\$a</code> größer als <code>\$b</code> ist.
<code>\$a <= \$b</code>	Kleiner Gleich	Gibt "wahr" zurück", wenn <code>\$a</code> kleiner oder gleich <code>\$b</code> ist.
<code>\$a >= \$b</code>	Größer Gleich	Gibt "wahr" zurück, wenn <code>\$a</code> größer oder gleich <code>\$b</code> ist.

Ein weiterer Vergleichs-Operator ist der `"?:"`- oder Trinitäts-Operator. Er arbeitet genauso wie in C und vielen anderen Programmier-Sprachen.

```
(ausdr1) ? (ausdr2) : (ausdr3);
```

Dieser Ausdruck gibt *ausdr2* zurück, wenn *ausdr1* "wahr" zurückgibt, und *ausdr3*, wenn *ausdr1* "falsch" zurückgibt.

Fehler-Kontroll-Operatoren

PHP unterstützt einen Operator zur Fehlerkontrolle: Das @-Symbol. Stellt man das @ in PHP vor einen Ausdruck werden alle Fehlermeldungen, die von diesem Ausdruck erzeugt werden könnten, ignoriert.

Ist das [track_errors](#)-Feature aktiviert, werden alle Fehlermeldungen, die von diesem Ausdruck erzeugt werden, in der globalen Variablen \$php_errormsg gespeichert. Da diese Variable mit jedem neuen Auftreten eines Fehlers überschrieben wird, sollte man sie möglichst bald nach Verwendung des Ausdrucks überprüfen, wenn man mit ihr arbeiten will.

```
<?php
/* Beabsichtigter SQL-Fehler (zusätzliches Anfuehrungszeichen): */
$res = @mysql_query( "select name, code from 'namelist' ) or
    die( "SQL-Anfrage fehlgeschlagen: Fehlermeldung: '$php_errormsg'" );
?>
```

Siehe auch [error_reporting\(\)](#).

Operatoren zur Programmausführung

PHP unterstützt einen Operator zur Ausführung externer Programme: Die sog. Backticks (`). Achtung: Die Backticks sind keine einfachen Anführungszeichen, sondern entsprechen auf den meisten Tastaturen den Tasten für die frz. Accents. PHP versucht, den Text zwischen den Backticks als Kommandozeilen-Befehl auszuführen. Zurückgegeben wird der die Ausgabe des aufgegebenen Programms (d. h. die Ausgabe wird nicht einfach gelöscht, sondern kann einer Variablen zugewiesen werden).

```
$output = `ls -al`;  
echo "<pre>$output</pre>";
```

Siehe auch [system\(\)](#), [passthru\(\)](#), [exec\(\)](#), [popen\(\)](#), und [escapeshellcmd\(\)](#).

Inkrementierungs- bzw. Dekrementierungsoperatoren

PHP unterstützt Prä- und Post-Inkrementierungs- und Dekrementierungsoperatoren im Stil der Programmiersprache C.

Tabelle 10-4 Inkrement- und Dekrementoperatoren

Beispiel	Name	Auswirkung
++\$a	Prä-Inkrement	Erhöht den Wert von \$a um eins (inkrementiert \$a), und gibt anschließend den neuen Wert von \$a zurück.
\$a++	Post-Inkrement	Gibt zuerst den aktuellen Wert von \$a zurück und erhöht dann den Wert von \$a um eins.
--\$a	Prä-Dekrement	Vermindert den Wert von \$a um eins (dekrementiert \$a), und gibt anschließend den neuen Wert von \$a zurück.
\$a--	Post-Dekrement	Gibt zuerst den aktuellen Wert von \$a zurück und erhöht dann den Wert von \$a um eins.

Ein einfach Beispiel-Skript:

```
<?php
echo "<h3>Post-Inkrement</h3>";
$a = 5;
echo "Sollte 5 sein: " . $a++ . "<br>\n";
echo "Sollte 6 sein: " . $a . "<br>\n";

echo "<h3>Prae-Inkrement</h3>";
$a = 5;
echo "Sollte 6 sein: " . ++$a . "<br>\n";
echo "Sollte 6 sein: " . $a . "<br>\n";

echo "<h3>Post-Dekrement</h3>";
$a = 5;
echo "Sollte 5 sein: " . $a-- . "<br>\n";
echo "Sollte 4 sein: " . $a . "<br>\n";

echo "<h3>Predecrement</h3>";
$a = 5;
```

```
echo "Sollte 4 sein: " . --$a . "<br>\n";  
echo "Sollte 4 sein: " . $a . "<br>\n";  
?>
```

[Zurück](#)

Operatoren zur
Programmausführung

[Anfang](#)

[Hoch](#)

[Vor](#)

Logische Operatoren

Logische Operatoren

Tabelle 10-5 Logische Operatoren

Beispiel	Name	Ergebnis
\$a and \$b	Und	Wahr sowohl \$a als auch \$b wahr ist.
\$a or \$b	Oder	Wahr wenn \$a oder \$b wahr ist.
\$a xor \$b	Entweder Oder	Wahr wenn entweder \$a oder \$b wahr ist, aber nicht beide.
! \$a	Nicht	Wahr wenn \$a nicht wahr ist.
\$a && \$b	Und	Wahr sowohl \$a als auch \$b wahr ist.
\$a \$b	Oder	Wahr wenn \$a oder \$b wahr ist.

Der Grund dafür, dass es je zwei unterschiedliche Operatoren für die "Und"- und die "Oder"-Verknüpfung gibt ist, dass die beiden Operatoren jeweils einen unterschiedliche Plätze in der Operator-Rangfolge haben. (siehe auch [Operator-Rangfolge](#).)

Operator-Rangfolge

Die Operator-Rangfolge legt fest, wie "eng" ein Operator zwei Ausdrücke miteinander verbindet. Das Ergebnis des Ausdruckes $1 + 5 * 3$ ist z. B. 16 und nicht 18, da der Multiplikations-Operator (*) in der Rangfolge höher steht als der Additions-Operator (+).

Die folgende Tabelle zeigt die Rangfolge der Operatoren, oben steht der Operator mit dem niedrigsten Rang.

Tabelle 10-6 Operator-Rangfolge

Associativity	Operator
links	,
links	or
links	xor
links	and
rechts	print
links	= += -= *= /= .= %= &= = ^= ~= <<= >>=
links	? :
links	
links	&&
links	
links	^
links	&
keine Richtung	== != ===
keine Richtung	< <= > >=
links	<< >>
links	+ - .
links	* / %
rechts	! ~ ++ -- (int) (double) (string) (array) (object) @
rechts	[
keine Richtung	new

Zeichenketten-Operatoren

Es gibt in PHP zwei Operatoren für Zeichenketten. Der erste ist Vereinigungs-Operator. ('.'), dessen Rückgabewert eine Zeichenkette, zusammengesetzt aus dem rechten und dem linken Argument, ist. Der zweite ist der Vereinigungs-Zuweisungsoperator ('.='). Siehe [Zuweisungs-Operatoren](#) für weitere Informationen.

```
$a = "Hallo ";  
$b = $a . "Welt!"; // $b enthaelt jetzt den Text "Hallo Welt!"  
  
$a = "Hallo ";  
$a .= "Welt!"; // $a enthaelt jetzt den Text "Hallo Welt!"
```

Kapitel 11 Kontroll-Strukturen

Inhaltsverzeichnis

[if](#)[else](#)[elseif](#)[Alternative Syntax für Kontroll-Strukturen](#)[while](#)[do..while](#)[for](#)[foreach](#)[break](#)[continue](#)[switch](#)[require\(\)](#)[include\(\)](#)[require_once\(\)](#)[include_once\(\)](#)

Jedes PHP-Skript besteht aus einer Reihe von Anweisungen. Eine Anweisung kann aus einem Funktions-Aufruf, einer Schleife, einer bedingten Anweisung oder einem Befehl, der nichts macht (eine leere Anweisung), bestehen. Jeder Befehl endet gewöhnlich mit einem Semikolon. Darüber hinaus können Befehle zu einer Anweisungsgruppe zusammengefasst werden, welche durch geschweifte Klammern begrenzt wird. Eine Anweisungsgruppe ist auch eine Anweisung. Die unterschiedlichen Arten von Anweisungen werden in diesem Abschnitt erläutert.

if

Der `if`-Befehl ist eine der wichtigsten Möglichkeiten vieler Programmier-Sprachen, PHP eingeschlossen. Er erlaubt die bedingte Ausführung von Programmteilen. PHP beinhaltet eine `if`-Struktur, die ähnlich der von C ist:

```
if (expr)
    Anweisung
```

Wie im Abschnitt über Expressions / Ausdrücke beschrieben, wird `expr` auf seinen wirklichen Wertinhalt ausgewertet. Wenn *expr* TRUE entspricht, wird PHP Anweisung ausführen, falls nicht - sie also FALSE ist - wird Anweisung übergangen.

Das folgende Beispiel wird `a ist grösser als b` anzeigen, wenn `$a` grösser ist als `$b`:

```
if ($a > $b)
    print "a ist grösser als b";
```

Oft werden sie die bedingte Ausführung von mehr als einer Anweisung wollen. Selbstverständlich ist es nicht erforderlich, jede Anweisung mit einer `if`-Bedingung zu versehen. Statt dessen können sie mehrere Anweisungen in Gruppen zusammenfassen. Z.B. wird der folgende Programm-Code `a ist grösser als b` anzeigen, wenn `$a` grösser ist als `$b`. Danach wird der Wert von `$a` in `$b` gespeichert:

```
if ($a > $b) {
    print "a ist grösser als b";
    $b = $a;
}
```

If-Anweisungen können ohne Einschränkung innerhalb anderer `if`-Anweisungen definiert werden. Das ermöglicht ihnen völlige Flexibilität bei der bedingten Ausführung verschiedenster Programmteile.

[Zurück](#)[Zeichenketten-Operatoren](#)[Anfang](#)[Hoch](#)[Vor](#)[else](#)

else

Häufig ist es erforderlich, eine Anweisung auszuführen, wenn eine bestimmte Bedingung erfüllt ist und eine andere Anweisung, falls sie nicht erfüllt ist. Dafür gibt es `else`. `Else` erweitert eine `if`-Anweisung um die Ausführung von Anweisungen, sobald der Ausdruck der `if`-Anweisung als `FALSE` angesehen wird. Der folgende Code wird z.B. `a ist grösser als b` anzeigen, wenn `$a` grösser ist als `$b`, anderenfalls `a ist NICHT grösser als b`:

```
if ($a > $b) {  
    print "a ist grösser als b";  
} else {  
    print "a ist NICHT grösser als b";  
}
```

Die `else`-Anweisung wird nur ausgeführt, wenn der `if`-Ausdruck als `FALSE` ausgewertet wurde und wenn bei vorhandenen `elseif`-Ausdrücken diese ebenfalls `FALSE` sind (siehe [elseif](#)).

elseif

`Elseif` ist, wie der Name schon sagt, eine Verbindung von `if` und `else`. Wie `else` erweitert sie eine `if`-Anweisung um die Ausführung anderer Anweisungen, sobald die normale `if`-Bedingung als `FALSE` angesehen wird. Anders als bei `else` wird die Ausführung dieser anderen Anweisungen nur durchgeführt, wenn die bei `elseif` angegebene alternative Bedingung als `TRUE` angesehen wird. Der folgende Code wird z.B. `a ist grösser als b`, `a ist gleich b` oder `a ist kleiner als b` ausgeben:

```
if ($a > $b) {  
    print "a ist grösser als b";  
} elseif ($a == $b) {  
    print "a ist gleich b";  
} else {  
    print "a ist kleiner als b";  
}
```

Es kann mehrere `elseif`-Anweisungen innerhalb einer `if`-Anweisung geben. Die erste `elseif`-Bedingung (falls vorhanden), die `true` ist, wird ausgeführt. In PHP kann man auch 'else if' schreiben (zwei Wörter). Das Verhalten ist identisch zu 'elseif' (ein Wort), genau wie in C.

Die `elseif`-Anweisung wird nur ausgeführt, wenn die vorausgehende `if`-Bedingung sowie jede vorherige `elseif`-Bedingung `FALSE` ist und die aktuelle `elseif`-Bedingung `TRUE` ist.

Alternative Syntax für Kontroll-Strukturen

PHP bietet eine alternative Syntax für einige seiner Kontroll-Strukturen; als da sind `if`, `while`, `for` und `switch`. Die Grundform der alternativen Syntax besteht immer aus dem Wechsel der öffnenden Klammer gegen einen Doppelpunkt (`:`) und der schliessenden Klammer gegen ein `endif;`, `endwhile;`, `endfor;` bzw. `endswitch;`.

```
<?php if ($a == 5): ?>
    A ist gleich 5
<?php endif; ?>
```

Im obigen Beispiel ist der HTML-Bereich eingebettet in eine `if`-Anweisung mit alternativer Syntax. Der HTML-Bereich wird nur ausgegeben, wenn `$a` gleich 5 ist.

Die alternative Syntax kann auch auf `else` und `elseif` angewendet werden. Es folgt eine `if`-Struktur mit `elseif` und `else` im alternativen Format:

```
if ($a == 5):
    print "a ist gleich 5";
    print "...";
elseif ($a == 6):
    print "a ist gleich 6";
    print "!!!";
else:
    print "a ist weder 5 noch 6";
endif;
```

Siehe auch [while](#), [for](#) und [if](#) für weitere Beispiele.

[Zurück](#)

elseif

[Anfang](#)[Hoch](#)[Vor](#)

while

while

`while`-Schleifen sind die einfachste Form von Schleifen in PHP. Sie funktionieren genau wie in C. Die Grundform einer `while`-Anweisung lautet:

```
while (expr) Anweisung
```

Die Bedeutung einer `while`-Anweisung ist einfach. Sie weist PHP an, einen in ihr eingebetteten Befehl so lange zu wiederholen, bis die `while`-Bedingung `TRUE` geworden ist. Der Wert der Bedingung wird immer am Anfang der Schleife geprüft. Wird der Wert während der Ausführung der Befehle innerhalb der `while`-Schleife geändert, endet die Ausführung dieses Befehls-Blocks nicht vor einem neuem Schleifen-Durchlauf (Iteration). Jeder Schleifendurchlauf ist eine Iteration. Falls die `while`-Bedingung bereits zu Beginn `FALSE` ist, werden die Anweisungen der `while`-Schleife nicht ein einziges Mal durchlaufen.

Wie bei der `if`-Anweisung kann man mehrere Befehle innerhalb einer Schleife angeben, indem man sie mit geschweiften Klammern umschliesst oder die alternative Syntax gebraucht:

```
while (expr): statement ... endwhile;
```

Die folgenden Beispiele sind identisch; beide geben Zahlen von 1 bis 10 aus:

```
/* Beispiel 1 */

$i = 1;
while ($i <= 10) {
    print $i++; /* es wird erst $i ausgegeben,
                  bevor der Wert erhöht wird
                  (post-increment) */
}

/* Beispiel 2 */

$i = 1;
while ($i <= 10):
    print $i;
    $i++;
endwhile;
```

[Zurück](#)

Alternative Syntax für
Kontroll-Strukturen

[Anfang](#)

[Hoch](#)

[Vor](#)

do..while

do..while

Do..while-Schleifen sind den while-Schleifen sehr ähnlich, ausser dass die Erfüllung des Ausdrucks am Ende jedes Durchlaufs geprüft wird (statt am Anfang). Der Hauptunterschied zu gewöhnlichen while-Schleifen ist, dass der erste Schleifen-Durchlauf bei do..while in jedem Fall statt findet, wogegen es bei while-Schleifen durchaus passieren kann, dass die Schleife nie durchlaufen wird, wenn die am Anfang zu prüfende Bedingung schon zu Beginn FALSE ist.

Es gibt nur eine Syntax für do..while-Schleifen:

```
$i = 0;
do {
    print $i;
} while ($i>0);
```

Die obige Schleife wird genau einmal durchlaufen, da nach der ersten Wiederholung die Erfüllung der Bedingung geprüft wird. Da diese aber nicht erfüllt, also FALSE ist (\$i ist nicht grösser als 0), wird die Schleifenausführung beendet.

Erfahrene C-Anwender kennen auch die Möglichkeit, Programm-Blöcke mit do..while(0) einzuschliessen und dann die [break](#) Anweisung zu benutzen. Der folgende Programm-Ausschnitt zeigt dies:

```
do {
    if ($i < 5) {
        print "i ist nicht gross genug";
        break;
    }
    $i *= $factor;
    if ($i < $minimum_limit) {
        break;
    }
    print "i ist ok";

    ...process i...
} while(0);
```

Es ist nicht weiter tragisch, wenn sie dieses Beispiel nicht oder nur zum Teil verstehen. Sie können auch ohne dieses Feature effektive PHP-Programme und Skripte schreiben.

[Zurück](#)
while

[Anfang](#)
[Hoch](#)

[Vor](#)
for

for

For-Schleifen sind die komplexesten Schleifen in PHP. Sie funktionieren wie ihr Gegenstück in C. Die Syntax einer `for`-Schleife sieht so aus:

```
for (expr1; expr2; expr3) Anweisung
```

Der erste Parameter (*expr1*) wird beim Schleifenbeginn geprüft bzw. ausgeführt (ohne jegliche Vorbedingung).

Zu Beginn jedes Durchlaufs wird nun *expr2* geprüft. Wenn dieser `TRUE` ist, fährt die Schleife weiter fort mit der Ausführung der nachfolgenden Befehle. Wenn das Ergebnis `FALSE` lautet, wird die Schleife beendet.

Am Ende jedes Durchlaufs wird nun auch noch *expr3* geprüft / ausgeführt.

Jeder der Parameter kann leer sein (optional). Ist *expr2* leer, wird die Schleife unbestimmt oft durchlaufen, da PHP ihn als `TRUE` wertet (wie in C). Das ist nicht so sinnlos, wie sie vielleicht glauben, weil man häufig eine Schleife erst durch eine bedingte [break](#)-Anweisung statt durch eine unwahr werdende `for`-Bedingung beenden möchte.

Beachten sie die folgenden Beispiele. Alle geben Zahlen von 1 bis 10 aus:

```
/* Beispiel 1 */
for ($i = 1; $i <= 10; $i++) {
    print $i;
}

/* Beispiel 2 */
for ($i = 1;;$i++) {
    if ($i > 10) {
        break;
    }
    print $i;
}

/* Beispiel 3 */
```

```
for
$i = 1;
for (;;) {
    if ($i > 10) {
        break;
    }
    print $i;
    $i++;
}

/* Beispiel 4 */

for ($i = 1; $i <= 10; print $i, $i++) ;
```

Selbstverständlich sieht das erste (oder das vierte) Beispiel am besten aus, aber sie werden noch feststellen, dass es oftmals ganz nützlich sein kann, leere Parameter in **for**-Schleifen zu verwenden.

PHP unterstützt auch bei **for**-Schleifen die alternative "Doppelpunkt-Syntax".

```
for (expr1; expr2; expr3): Anweisung; ...; endfor;
```

Andere Sprachen haben für das Durchlaufen eines Hash's oder Arrays eine **foreach**-Anweisung. PHP3 hat dies nicht; im Gegensatz zu PHP4 (vgl. [foreach](#)). In PHP3 kann man dafür eine Kombination von [while](#) mit der [list\(\)](#)- und [each\(\)](#)-Funktion einsetzen. Beispiele finden sie in der Dokumentation zu diesen Funktionen.

[Zurück](#)
do..while

[Anfang](#)
[Hoch](#)

[Vor](#)
foreach

foreach

PHP4 (nicht PHP3) enthält eine `foreach`-Funktion, genau wie Perl und andere Sprachen. Diese ermöglicht es, auf einfache Weise ein Array zu durchlaufen. Es gibt zwei Syntax-Formen; die zweite ist eine unbedeutende aber sinnvolle Erweiterung der ersten Syntax:

```
foreach(array_expression as $value) Anweisung  
foreach(array_expression as $key => $value) Anweisung
```

Die erste Form durchläuft das `array_expression`-Array. Bei jedem Durchgang wird der Wert des aktuellen Elements `$value` zugewiesen und der interne Array-Zeiger um 1 erhöht. Dadurch wird beim nächsten Durchgang automatisch das nächste Element ausgewertet.

Die zweite Form arbeitet genauso, ausser dass bei jedem Durchlauf auch der aktuelle Schlüssel der Variablen `$key` zugewiesen wird.

Anmerkung: Sobald `foreach` zum ersten Mal ausgeführt wird, wird der interne Array-Pointer automatisch auf das erste Element des Arrays zurück gesetzt. Das heisst, dass sie nicht [reset\(\)](#) vor einem Durchlauf von `foreach` aufrufen müssen.

Anmerkung: Beachten sie auch, dass `foreach` mit einer Kopie des angegebenen Arrays arbeitet. Deshalb wird auch der Array-Pointer nicht wie bei dem `each`-Konstrukt verändert.

Beachten sie, dass die folgenden Beispiele in ihrer Funktionalität identisch sind:

```
reset ($arr);  
while (list(, $value) = each ($arr)) {  
    echo "Wert: $value<br>\n";  
}  
  
foreach ($arr as $value) {  
    echo "Wert: $value<br>\n";  
}
```

Auch hier funktioniert alles gleich:

```

reset ($arr);
while (list($key, $value) = each ($arr)) {
    echo "Schlüssel: $key; Wert: $value<br>\n";
}

foreach ($arr as $key => $value) {
    echo "Schlüssel: $key; Wert: $value<br>\n";
}

```

Noch einige Beispiele, die die Anwendung verdeutlichen:

```

/* foreach Beispiel 1: Nur der Wert */

$a = array (1, 2, 3, 17);

foreach ($a as $v) {
    print "Aktueller Wert von \$a: $v.\n";
}

/* foreach Beispiel 2: Wert (mit Ausgabe des Array-Schlüssels) */

$a = array (1, 2, 3, 17);

$i = 0; /* nur für Anschauungs-Zweck */

foreach($a as $v) {
    print "\$a[$i] => $k.\n";
}

/* foreach Beispiel 3: Schlüssel und Wert */

$a = array (
    "one" => 1,
    "two" => 2,
    "three" => 3,
    "seventeen" => 17
);

foreach($a as $k => $v) {
    print "\$a[$k] => $v.\n";
}

```

foreach

[Zurück](#)

for

[Anfang](#)

[Hoch](#)

[Vor](#)

break

break

Break bricht die Ausführung der aktuellen `for`, `while` oder `switch` Anweisungs-Sequenz ab.

Einem `break` kann optional ein numerisches Argument angehängt werden, das die Anzahl der abzubrechenden Befehls-Sequenzen enthält.

```
$arr = array('eins', 'zwei', 'drei', 'stop', 'vier');
while (list( , $val) = each($arr)) {
    if ($val == 'stop') {
        break; /* Man kann hier auch 'break 1;' schreiben. */
    }
    echo "$val<br>\n";
}

/* Benutzung des optionalen Argumentes. */
$i = 0;
while ( ++$i ) {
    switch ( $i ) {
        case 5:
            echo "Bei 5<br>\n";
            break 1; /* Beendet nur switch. */
        case 10:
            echo "Bei 10; quitting<br>\n";
            break 2; /* Beendet switch und while. */
        default:
            break;
    }
}
```


continue

Continue wird innerhalb von Schleifen gebraucht. Die Schleife wird an der aktuellen Stelle abgebrochen und es wird der nächste Durchlauf begonnen.

Continue kann optional ein numerisches Argument erhalten, das die Anzahl der zu überspringenden Schleifendurchläufe beinhaltet.

```
while (list ($key, $value) = each ($arr)) {  
    if (!($key % 2)) { // überspringe ungerade Werte  
        continue;  
    }  
    tue_was_mit_ungerade ($value);  
}  
  
$i = 0;  
while ($i++ < 5) {  
    echo "Aussen<br>\n";  
    while (1) {  
        echo "  Mitte<br>\n";  
        while (1) {  
            echo "    Innen<br>\n";  
            continue 3;  
        }  
        echo "Das wird nie ausgegeben.<br>\n";  
    }  
    echo "Dies auch nicht.<br>\n";  
}
```

switch

Die `switch`-Anweisung ist gleichbedeutend einer Reihe von `if`-Anweisungen mit dem gleichen Parameter. Häufig wollen sie ein und die selbe Variable (bzw. den selben Ausdruck) mit verschiedensten Werten vergleichen und in Abhängigkeit vom Auswertungsergebnis verschiedene Programmteile ausführen. Genau das ermöglicht die `switch`-Anweisung.

Die folgenden 2 Beispiele zeigen 2 verschiedene Wege, das gleiche zu bewirken; eins gebraucht mehrere `if`-Befehle, das andere eine `switch`-Anweisung:

```
if ($i == 0) {
    print "i ist gleich 0";
}
if ($i == 1) {
    print "i ist gleich 1";
}
if ($i == 2) {
    print "i ist gleich 2";
}

switch ($i) {
    case 0:
        print "i ist gleich 0";
        break;
    case 1:
        print "i ist gleich 1";
        break;
    case 2:
        print "i ist gleich 2";
        break;
}
```

Es ist wichtig, die Ausführung einer `switch`-Anweisung zu verstehen, um Fehler zu vermeiden. Die `switch`-Anweisung wird Zeile für Zeile (also Anweisung für Anweisung) abgearbeitet. Zu Beginn wird nichts ausgeführt. Erst wenn bei einem `case`-Teil eine Entsprechung zum `switch`-Ausdruck vorliegt, werden die darin enthaltenen Befehle ausgeführt. PHP fährt dann mit der Abarbeitung des restlichen Codes innerhalb des `switch`-Blocks fort (oder bis zum ersten Auftreten einer `break`-Anweisung). Ohne `break` am Ende eines `case`-Teils werden also noch die folgenden `case`-Blöcke ausgeführt. Z.B.:

```
switch ($i) {  
    case 0:  
        print "i ist gleich 0";  
    case 1:  
        print "i ist gleich 1";  
    case 2:  
        print "i ist gleich 2";  
}
```

Wenn hier \$i gleich 0 ist, würde PHP alle print-Anweisungen ausführen. Ist \$i gleich 1, würden die letzten beiden print-Befehle ausgeführt und wenn \$i = 2 ist, würde nur der letzte print-Befehl ausgeführt. Deshalb ist es wichtig, bei der ersten gefundenen Übereinstimmung eine **break**-Anweisung zu setzen (abgesehen von bestimmten Fällen, wo genau dieses Verhalten gefordert wird).

Bei einer **switch**-Anweisung wird die Bedingung also nur einmal überprüft und das Ergebnis mit jeder **case**-Anweisung verglichen. Bei einem **elseif**-Befehl wird die Bedingung neu geprüft. Eine **switch**-Anweisung kann schneller als mehrere **if**-Befehle sein, z.B. bei komplizierteren Bedingungen als einem einfachen Vergleich.

Der Anweisungsteil von **case** kann auch leer sein. Dann wird die Kontrolle einfach an den nächsten **case**-Teil übergeben.

```
switch ($i) {  
    case 0:  
    case 1:  
    case 2:  
        print "i ist kleiner als 3 aber nicht negativ";  
        break;  
    case 3:  
        print "i ist gleich 3";  
}
```

Ein Spezialfall ist die Anweisung **default**. Diese trifft auf alles zu, was nicht von den voranstehenden **case**-Ausdrücken erfasst wurde, wie z.B.:

```
switch ($i) {  
    case 0:  
        print "i ist gleich 0";  
        break;  
    case 1:  
        print "i ist gleich 1";  
        break;  
    case 2:  
        print "i ist gleich 2";  
        break;  
    default:  
        print "i ist weder 0, 1 noch 2";  
}
```

Der **case**-Ausdruck kann eine Prüfung einfacher Typen sein, also von Integer- oder Fließkomma-Zahlen oder von Strings / Zeichenketten. Arrays oder Objekte können nicht benutzt werden, es sei denn, sie werden auf einfache Typen herunter gebrochen.

Die alternative Syntax gilt auch für switch-Sequenzen. Mehr Informationen dazu unter [Alternative Syntax für Kontroll-Strukturen](#) .

```
switch ($i):  
    case 0:  
        print "i ist gleich 0";  
        break;  
    case 1:  
        print "i ist gleich 1";  
        break;  
    case 2:  
        print "i ist gleich 2";  
        break;  
    default:  
        print "i ist weder 0, 1 noch 2";  
endswitch;
```

require()

Der [require\(\)](#)-Befehl setzt an seine Stelle den Inhalt der angegebenen Datei (ähnlich dem #include von C).

Wichtig: PHP kehrt zu Beginn der per [include\(\)](#) oder [require\(\)](#) eingebundenen Dateien vom PHP- in den HTML-Modus und am Schluß der Datei wieder vom HTML- in den PHP-Modus zurück. Falls innerhalb dieser Dateien also PHP-Code ausgeführt werden soll, muss dieser eingeschlossen werden von gültigen PHP-Start- und PHP-Ende-Marken (siehe [gültige PHP-Start- und Ende-Tags](#)).

Require() ist keine PHP-Anweisung, sondern eine spezielle Sprachanweisung. Sie ist anderen Regeln unterworfen als Funktionen. Einerseits unterliegt sie keinen Kontroll-Strukturen, andererseits gibt sie keinen Wert zurück. Der Versuch, von einem [require\(\)](#)-Aufruf einen Rückgabewert zu erhalten, führt zu einem Parse-Fehler.

Anders als [include\(\)](#) wird [require\(\)](#) *immer* die angegebene Datei einlesen, *auch dann, wenn die Programmzeile, in der sie steht, nicht ausgeführt wird*. Wenn sie eine Datei nur bedingt einlesen wollen, benutzen sie [include\(\)](#). Die bedingte Anweisung würde [require\(\)](#) nicht davon abhalten, die Datei zu laden.

Innerhalb von Schleifen tritt der Effekt auf, dass, obwohl die den [require\(\)](#)-Befehl enthaltende Zeile mehrfach angesprungen wird, die entsprechende Datei trotzdem nur genau einmal eingelesen (ausgeführt) wird.

Sie können also [require\(\)](#) nicht innerhalb von Programm-Schleifen einsetzen. Deshalb, und wenn sie verschiedene Dateien einlesen wollen, müssen sie in Schleifen den Befehl [include\(\)](#) benutzen.

```
require ('header.inc');
```

Beachten sie, dass sowohl [include\(\)](#) als auch [require\(\)](#) den Inhalt der angegebenen Datei in das Skript nur 1:1 einlesen. Sie kann weder über HTTP noch sonstwie eingelesen werden. Jede vor dem Einbinden der Datei definierte Variable ist innerhalb dieser Datei verfügbar, da sie Teil des umgebenden Skripts wird.

```
require ("file.inc?vareins=1&varzwei=2"); /* Funkt. nicht. */  
  
$vareins = 1;  
$varzwei = 2;  
/* $vareins und $varzwei sind in file.inc verfügbar */  
require ("file.inc");
```

Lassen sie sich nicht durch die Tatsache irre führen, dass sie Dateien über HTTP anfordern oder einbinden können (siehe [Remote files](#)-Funktionalität). Die obigen Angaben gelten dennoch.

In PHP3 ist es möglich, eine `return`-Anweisung innerhalb einer per [require\(\)](#) eingebundenen Datei auszuführen, solange diese Anweisung im Hauptteil dieser Datei steht. Die `return`-Anweisung darf nicht in Blöcken (also innerhalb von geschwungenen Klammern "{}" stehen. In PHP4 existiert diese Möglichkeit nicht mehr. Wenn sie diese dennoch benötigen, sehen sie unter [include\(\)](#) nach.

[Zurück](#)
switch

[Anfang](#)
[Hoch](#)

[Vor](#)
[include\(\)](#)

include()

Die [include\(\)](#)-Anweisung liest die angegebene Datei ein und wertet sie aus.

Wichtig ist, dass beim [include\(\)](#)- oder [require\(\)](#)-Befehl vom PHP-Parsing-Modus in den HTML-Modus geschaltet wird und bei Rückkehr in das aufrufende Skript wieder zurück vom HTML- in den PHP-Modus. Deshalb muss jeder PHP-Code innerhalb der eingebundenen Dateien umschlossen werden von [gültigen PHP-Start- und Ende-Tags](#).

Das passiert jedes Mal, sobald die [include\(\)](#)-Anweisung auftritt. Deshalb können sie [include\(\)](#) auch sehr gut innerhalb von Schleifen verwenden, um eine Anzahl unterschiedlicher Dateien einzubinden.

```
$files = array ('first.inc', 'second.inc', 'third.inc');
for ($i = 0; $i < count($files); $i++) {
    include $files[$i];
}
```

Include() unterscheidet sich von [require\(\)](#) dadurch, dass die include-Anweisung jedesmal neu ausgewertet wird, sobald sie auftritt (und nur zur Ausführungszeit). Dagegen wird die [require\(\)](#)-Anweisung beim ersten Auftreten mit der angegebenen Datei ersetzt, egal ob sie einzubinden ist oder nicht (innerhalb von bedingten Anweisungen, z.B. bei [if](#) auch dann, wenn das ausgewertete Argument false ergeben hat).

Da [include\(\)](#) ein spezieller Sprach-Konstrukt ist, müssen sie ihn innerhalb einer bedingten Anweisung in einen Anweisungsblock setzen.

```
/* Das ist falsch und führt nicht zum gewünschten Ergebnis. */

if ($Bedingung)
    include($diesedatei);
else
    include($anderedatei);

/* Diese ist korrekt. */

if ($Bedingung) {
    include($diesedatei);
} else {
    include($anderedatei);
}
```

Sowohl in PHP3 als auch in PHP4 ist es möglich, eine `return`-Anweisung innerhalb einer [include\(\)](#)eten Datei anzugeben, um die Ausführung dieser Datei abubrechen und zum aufrufenden-Skript zurück zu kehren. Es gibt aber ein paar Unterschiede in der Arbeitsweise. Der erste ist, dass bei PHP3 die `return`-Anweisung nicht innerhalb eines Blocks auftreten darf, es sei denn, es ist ein Funktions-Block. In diesem Fall gilt `return` für diese Funktion und nicht für die ganze Datei. In PHP4 gibt es diese Beschränkung nicht. PHP4 erlaubt ihnen auch die Rückgabe von Werten bei

include

[include\(\)](#)eten Dateien. Sie können den Wert des [include\(\)](#)-Aufrufs nutzen, als wenn sie eine Funktion aufgerufen hätten. Das wird in PHP3 eine Parse-Error ergeben.

Beispiel 11-1 Include() in PHP3 und PHP4

Beachten sie, dass die folgende Datei, genannt `test.inc`) im gleichen Verzeichnis wie die Hauptdatei stehen muss:

```
<?php
echo "Vor dem Return <br>\n";
if (1) {
    return 27;
}
echo "Nach dem Return <br>\n";
?>
```

Die Datei `main.html` enthält folgendes:

```
<?php
$retval = include ('test.inc');
echo "Datei gibt zurück: '$retval'<br>\n";
?>
```

Sobald `main.html` in PHP3 aufgerufen wird, wird ein Parse-Error in Zeile 2 angezeigt; sie können in PHP3 also keinen Rückgabewert von [include\(\)](#) erhalten. In PHP4 wird das Ergebnis sein:

```
Vor dem Return
Datei gibt zurück: '27'
```

Die Datei `main.html` soll nun folgendes enthalten:

```
<?php
include ('test.inc');
echo "Zurück in main.html<br>\n";
?>
```

In PHP4 wird die Ausgabe sein:

```
Vor dem Return
Zurück in main.html
```

PHP3 wird dagegen folgendes ausgeben:

```
Vor dem Return
27Zurück in main.html
```

Parse error: parse error in /home/torben/public_html/phptest/main.html on line 5

Der obige Parse-Error ist das Ergebnis der Tatsache, dass die `return`-Anweisung innerhalb eines Nicht-Funktions-Blocks von `test.inc` steht. Wenn das `return` ausserhalb dieser Blocks zu stehen kommt, wird die Ausgabe wie folgt aussehen:

```
Before the return
27Back in main.html
```

Die '27' entsteht aus der Tatsache, dass PHP3 keine solchen Rückgabewerte unterstützt.

Beachten sie, dass sowohl [include\(\)](#) als auch [require\(\)](#) den Inhalt der einzufügenden Datei an ihre eigene Stelle des Skripts setzen. Sie lesen diese Datei nicht über HTTP oder sonstiges ein. Deshalb ist jede in der Hauptdatei gesetzte Variable auch innerhalb der eingebundenen Datei vorhanden; sie ist ja Bestandteil dieser Datei geworden.

```
include ("file.inc?vareins=1&varzwei=2"); /* Funkt. nicht. */

$vareins = 1;
$varzwei = 2;
/* $vareins und $varzwei sind in file.inc verfügbar */
include ("file.inc");
```

Lassen sie sich nicht durch die Tatsache irritieren, dass sie Dateien auch über HTTP per `require` oder `include` einbinden können (vgl. [Remote files](#)). Das oben Gesagte behält seine Gültigkeit.

Siehe auch [readfile\(\)](#), [require\(\)](#) und [virtual\(\)](#).

[Zurück](#)
[require\(\)](#)

[Anfang](#)
[Hoch](#)

[Vor](#)
[require_once\(\)](#)

require_once()

Wird noch erweitert: Die [require_once\(\)](#)-Anweisung ersetzt sich selbst durch die angegebene Datei (ähnlich der C-Preprozessor-Anweisung `#include`).

Siehe auch: [require\(\)](#), [include\(\)](#), [include_once\(\)](#), [get_required_files\(\)](#) und [get_included_files\(\)](#).

include_once()

Wird noch erweitert: Die [include_once\(\)](#)-Anweisung ersetzt sich selbst durch die angegebene Datei (ähnlich der C-Preprozessor-Anweisung `#include`).

Siehe auch: [require\(\)](#), [include\(\)](#), [require_once\(\)](#), [get_required_files\(\)](#) und [get_included_files\(\)](#).

Kapitel 12 Funktionen

Inhaltsverzeichnis

[Vom Nutzer definierte Funktionen](#)

[Funktionsparameter](#)

[Rückgabewerte](#)

[old function](#)

[Variablenfunktionen](#)

Vom Nutzer definierte Funktionen

Eine Funktion kann wie folgt definiert werden:

```
function foo ($arg_1, $arg_2, ..., $arg_n) {  
    echo "Beispielfunktion.\n";  
    return $retval;  
}
```

Jeder beliebige korrekte PHP-Code kann in einer Funktion vorkommen, sogar andere Funktionen und [Klassen-](#) Definitionen.

In PHP3 müssen Funktionen definiert sein, bevor man auf sie verweist. In PHP4 ist das nicht mehr erforderlich.

Funktionsparameter

Mit einer Parameterliste kann man Informationen an eine Funktion übergeben. Die Parameterliste ist eine kommasetrennte Liste von Variablen und/oder Konstanten.

PHP unterstützt die Weitergabe von Parametern als Werte (das ist der Standard), als [Verweise](#), und als [Vorgabewerte](#). Die Übergabe einer beliebigen Parameterzahl ist nicht möglich, einen ähnlichen Effekt kann man aber durch die Übergabe von Arrays erreichen.

```
function rechne_array($eingabe) {  
    echo "$eingabe[0] + $eingabe[1] = ", $eingabe[0]+$eingabe[1];  
}
```

Verweise als Parameter übergeben

Normalerweise werden den Funktionen Werte als Parameter übermittelt. Wenn man den Wert dieses Parameters innerhalb der Funktion ändert, bleibt der Parameter ausserhalb der Funktion unverändert. Wollen Sie aber genau das erreichen, dann müssen Sie die Parameter als Verweise übergeben.

Wenn eine Funktion einen Parameter generell als Verweis behandeln soll, setzt man ein kaufmännisches Und (&) in der Funktionsdefinition vor den Parameternamen:

```
function fuege_etwas_anderes_an (&$string) {  
    $string .= 'und nun zu etwas völlig anderem.';  
}  
$str = 'Dies ist ein String, ';  
fuege_etwas_anderes_an ($str);  
echo $str; // Ausgabe 'Dies ist ein String, und nun zu etwas völlig anderem.'
```

Wenn Sie eine Variable andererseits als Verweis an eine Funktion übergeben wollen, die dies nicht tut, können Sie das kaufmännische Und (&) auch beim Aufruf der Funktion verwenden:

```
function bla ($fasel) {  
    $fasel .= ' und nun zu etwas völlig anderem.';  
}  
$str = 'Dies ist ein String, , ';  
bla ($str);  
echo $str; // outputs 'Dies ist ein String, '  
bla (&$str);  
echo $str; // outputs 'Dies ist ein String, und nun zu etwas völlig anderem.'
```

Vorgabewerte für Parameter

Eine Funktion kann C++-artige Vorgabewerte für skalare Parameter wie folgt definieren:

```
function machkaffee ($typ = "Cappucino") {
    return "Ich mache eine Tasse $typ.\n";
}
echo machkaffee ();
echo makecoffee ("Espresso");
```

Die Ausgabe von diesem kleinen Skript ist:

```
Ich mache eine Tasse Cappucino.
Ich mache eine Tasse Espresso.
```

Der Vorgabewert muß eine Konstante sein, darf also (zum Beispiel) keine Variable oder Element einer Klasse sein.

Bitte beachten Sie, dass alle Vorgabewerte rechts von den Nicht-Vorgabeparametern stehen sollten; - sonst wird es nicht funktionieren. Betrachten Sie folgendes Beispiel:

```
function mach_joghurt ($typ = "rechtsdrehendes", $geschmack) {
    return "Mache einen Becher $typ $geschmack-joghurt.\n";
}

echo mach_joghurt ("Brombeer");    // arbeitet nicht wie erwartet
```

Die Ausgabe dieses Beispiels ist::

```
Warning: Missing argument 2 in call to makeyogurt() in
/usr/local/etc/httpd/htdocs/php3test/functest.html on line 41
Mache einen Becher Brombeer-joghurt.
```

Nun vergleichen Sie bitte oberes Beispiel mit folgendem:

```
function mach_joghurt ($geschmack, $typ = "rechtsdrehendes") {
    return "Mache einen Becher $typ $geschmack-joghurt.\n";
}

echo mach_jogurt ("Brombeer");    // arbeitet wie erwartet.
```

... und jetzt ist die Ausgabe:

```
Mache einen Becher rechtsdrehendes Brombeer-Joghurt.
```

[Zurück](#)

Funktionen

[Anfang](#)

[Hoch](#)

[Vor](#)

Rückgabewerte

Rückgabewerte

Sie können Werte mit dem optionalen Befehl "return" zurückgeben. Es können Variablen jedes Typs zurückgegeben werden, auch Listen oder Objekte.

```
function quadrat ($zahl) {  
    return $zahl * $zahl;  
}  
echo quadrat (4);    // gibt '16' aus.
```

Es ist leider nicht möglich, mehrere Werte von einer Funktion zurückgeben zu lassen. Ein ähnliches Resultat kann man aber durch die Rückgabe von Listen bekommen.

```
function kleine_zahlen() {  
    return array (0, 1, 2);  
}  
list ($null, $eins, $zwei) = small_numbers();
```


old_function

Der Befehl `old_function` erlaubt es, ein Funktion genauso wie bei PHP/FI2 zu definieren (nur "function" wird durch "old_function" ersetzt).

Dieser Befehl sollte nur vom PHP/FI2->PHP3-Konverter genutzt werden.

Warnung

Funktionen, die durch `old_function` definiert werden, können nicht von PHP selbst aufgerufen werden. Das heißt unter anderem, dass sie nicht in Funktionen wie **`usort ()`**, [array_walk\(\)](#) und **`register_shutdown_function()`** benutzt werden können. Dieses Manko können Sie umgehen, indem Sie eine Adapterfunktion in normalem PHP3 schreiben, die die `old_function` aufruft.

Variablenfunktionen

PHP unterstützt das Konzept der Variablenfunktionen. Wenn Sie an das Ende einer Variablen Klammern hängen, versucht PHP eine Funktion aufzurufen, deren Name der aktuelle Wert der Variable ist. Diese Möglichkeit kann für Callbacks, Funktionstabellen und

Beispiel 12-1 Beispiel für Variablenfunktionen

```
<?php
function blah() {
    echo "In blah()<br>\n";
}

function fasel( $arg = ' ' ) {
    echo "In fasel(); ist der Parameter '$arg'.<br>\n";
}

$func = 'blah';
$func();
$func = 'fasel';
$func( 'test' );
?>
```

Kapitel 13 Klassen und Objekte

Inhaltsverzeichnis

[Klassen](#)

Klassen

Eine Klasse ist eine Sammlung von Variablen und von Funktionen, die mit diesen Variablen arbeiten. Eine Klasse wird folgendermaßen definiert:

```
<?php
class Einkaufswagen {
    var $produkte; // Produkte in Ihrem Einkaufswagen

    // Füge dem Einkaufswagen $anzahl Artikel der Sorte $artnr zu

    function fuege_hinzu ($artnr, $anzahl) {
        $this->produkte[$artnr] += $anzahl;
    }

    // Nimm $anzahl von Artikel wieder aus dem Einkaufswagen

    function nimm_heraus ($artnr, $anzahl) {
        if ($this->produkte[$artnr] > $anzahl) {
            $this->produkte[$artnr] -= $anzahl;
            return true;
        } else {
            return false;
        }
    }
}
?>
```

In diesem Beispiel wird eine Klasse "Einkaufswagen" definiert. Sie besteht aus einem assoziativen Array von Produkten im Einkaufswagen und zwei Funktionen zum Zufügen und Entfernen von Einkäufen.

Klassen sind Typen, das heisst die Blaupausen für reale Variablen. Um sie zu nutzen, muß zunächst eine Variable mit dem Operator `new` angelegt werden.

```
$einkaufswagen = new Einkaufswagen;
$einkaufswagen->fuege_hinzu("10", 1);
```

Hier wird das Objekt \$einkaufswagen aus der Klasse Einkaufswagen geschaffen. Dann wird die enthaltene Funktion fuege_hinzu() aufgerufen, um ein Produkt mit der Artikelnummer 10 in den Einkaufswagen zu tun.

Klassen können ebenfalls Erweiterungen von anderen Klassen sein. Die erweiterte, oder auch abgeleitete Klasse enthält alle Funktionen der ursprünglichen Klasse, und dazu die eigenen Ergänzungen. Das geschieht mit dem Schlüsselwort "extends". Mehrfachvererbung wird von PHP nicht unterstützt.

```
class Mein_Einkaufswagen extends Einkaufswagen {
    var $besitzer;

    function setze_besitzer ($name) {
        $this->besitzer = $name;
    }
}
```

In diesem Beispiel wird eine Klasse Mein_Einkaufswagen definiert. Sie enthält alle Funktionen der Klasse Einkaufswagen, eine zusätzliche Variable \$besitzer und die zusätzliche Funktion setze_besitzer(). Man kann den Einkaufswagen auch weiterhin wie oben erzeugen, nur kann man jetzt auch den Besitzer setzen oder herausfinden. Die alten Funktionen der Klasse Einkaufswagen können ebenfalls weiterverwendet werden.

```
$meinkaufswagen = new Mein_Einkaufswagen; // Kreiere einen Einkaufswagen
$meinkaufswagen->setze_besitzer ("kris"); // Name dieser Klasse
print $meinkaufswagen->besitzer; // schreibe den Namen des Besitzers
$meinkaufswagen->fuege_hinzu ("10", 1); // (Siehe oben, vererbt
// von Einkaufswagen)
```

Innerhalb der Funktionen einer Klasse bezeichnet die Variable \$this das aktuelle Objekt. Sie können mit \$this->irgendwas auf dessen Variablen und Funktionen zugreifen.

Konstruktoren sind Funktionen einer Klasse, die beim Erschaffen eines neuen Objektes automatisch aufgerufen werden. Eine Funktion wird zu einem Konstruktor, wenn Sie den gleichen Namen wie die Klasse trägt.

```
class Auto_Einkaufswagen extends Einkaufswagen {
    function Auto_Einkaufswagen () {
        $this->fuege_hinzu ("10", 1);
    }
}
```

Die Klasse Auto_Einkaufswagen entspricht der Klasse Einkaufswagen plus einen Konstruktor, der bereits für eine erste Füllung (1 Artikel der Nummer 10) gesorgt hat. Jeder neu erzeugte Auto_Einkaufswagen enthält so von vorneherein diesen Artikel. Konstruktoren können auch Parameter enthalten. Aber diese Parameter sind optional, und können so nützlicher eingesetzt werden.

```
class Konstruktor_Einkaufswagen extends Einkaufswagen {  
    function Konstruktor_Einkaufswagen ($produkt = "10", $anzahl = 1) {  
        $this->fuege_hinzu ($produkt, $anzahl);  
    }  
}  
  
// Kaufe wieder den gleichen alten Kram ein.  
  
$standard_einkaufswagen = new Konstruktor_Einkaufswagen;  
  
// Kaufe etwas bestimmtes ein ...  
  
$anderer_Einkaufswagen = new Konstruktor_Einkaufswagen ("20", 17);
```

Achtung

Bei abgeleiteten Klassen wird der Konstruktor der Ursprungsklasse nicht automatisch aufgerufen, wenn der Konstruktor der abgeleiteten Klasse aufgerufen wird.

[Zurück](#)

Variablenfunktionen

[Anfang](#)[Hoch](#)[Vor](#)

Features

III Features

Inhaltsverzeichnis

- 14 [Fehlerbehandlung](#)
 - 15 [Creating GIF images](#)
 - 16 [HTTP-Authentifizierung mit PHP](#)
 - 17 [Cookies](#)
 - 18 [Steuerung von Dateiuploads](#)
 - 19 [Using remote files](#)
 - 20 [Verbindungssteuerung](#)
 - 21 [Persistente Datenbankverbindungen](#)
-

Kapitel 14 Fehlerbehandlung

Es gibt 4 verschiedene Arten von Fehlermeldungen und Warnungen in PHP. Das sind:

- 1 - Normale Funktionsfehler
- 2 - Normale Warnungen
- 4 - Parser-Fehler
- 8 - Mitteilungen (Warnungen, die ignoriert werden können, die aber eventuell auf einen Fehler im Code schließen lassen.)

Die oben aufgeführten Fehlernummern werden addiert, um den Fehlerlevel festzulegen. Der standardmäßige Wert ist 7, d.h. $1 + 2 + 4$ oder, anders ausgedrückt, es werden alle Fehlermeldungen außer den Mitteilungen ausgegeben. Der Fehlerlevel kann in der Konfigurationsdatei `php3.ini` durch die `error_reporting` Option gesetzt werden. Es ist auch möglich, den Fehlerlevel in der Apache-Konfigurationsdatei `httpd.conf` durch die Option `php3_error_reporting` oder zur Laufzeit durch ein Skript, das die Funktion [error_reporting\(\)](#) nutzt, zu setzen.

Alle [PHP Funktionen](#) können auch mit vorangestelltem "@" aufgerufen werden. Das bewirkt das Unterdrücken der Fehlermeldung für die Funktion. Bei auftretendem Fehler und eingeschalteter [track_errors](#) Option wird die entsprechende Fehlermeldung in der globalen Variablen `$php_errormsg` gespeichert.

Kapitel 15 Creating GIF images

PHP ist nicht darauf beschränkt, nur HTML auszugeben. Es kann auch dazu genutzt werden, einzelne GIF-Bilddateien oder auch auf bequeme Art und Weise eine ganze Abfolge von GIF's zu erzeugen. Um all das zu tun, müssen sie nur die GD-Library mit Bildfunktionen eincompilieren.

Beispiel 15-1 GIF-Erzeugung mit PHP

```
<?php
    Header("Content-type: image/gif");
    $string=implode($argv, " ");
    $im = imagecreatefromgif("images/taste1.gif");
    $orange = ImageColorAllocate($im, 220, 210, 60);
    $px = (imagesx($im)-7.5*strlen($string))/2;
    ImageString($im, 3, $px, 9, $string, $orange);
    ImageGif($im);
    ImageDestroy($im);
?>
```

Dieses Beispiel würde von einer Seite mit einem Tag wie diesem aufgerufen: ``. Der obige taste.php3-Skript nimmt dann den "text"-String und legt ihn über das Grundbild, was in diesem Fall "images/taste1.gif" ist, und gibt das endgültige Bild aus. Das ist ein wirklich komfortabler Weg zu vermeiden, dass man jedesmal, wenn man den Text auf Tasten ändert, diese von Hand neu zeichnen muss, denn werden sie dynamisch generiert.

Kapitel 16 HTTP-Authentifizierung mit PHP

Die HTTP-Authentifizierung durch PHP ist nur verfügbar, wenn PHP als Apache-Modul läuft und funktioniert daher nicht mit der CGI-Version. In einem PHP-Skript für ein Apache-Modul kann man die **Header()** benutzen, um eine "Authentifizierung notwendig"-Nachricht an den Client-Browser zu senden, damit dieser ein Fenster zur Eingabe von Benutzernamen/Passwort öffnet. Hat der Benutzer diese eingegeben, wird die URL des PHP-Skripts mit den Variablen `$PHP_AUTH_USER`, `$PHP_AUTH_PW` und `$PHP_AUTH_TYPE`, die den jeweiligen Benutzernamen, das Passwort und den Typ der Identifizierung enthalten, erneut aufgerufen. Momentan wird nur die grundlegende Authentifizierung unterstützt. Näheres hierzu bei der **Header()** Funktion.

Ein Auszug aus einem Skript, das die Clientauthentifizierung auf einer Seite erzwingt, würde so aussehen:

Beispiel 16-1 HTTP Authentifizierung

```
<?php
if(!isset($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\n";
    exit;
} else {
    echo "Hello $PHP_AUTH_USER.<P>";
    echo "You entered $PHP_AUTH_PW as your password.<P>";
}
?>
```

Anstatt `$PHP_AUTH_USER` und `$PHP_AUTH_PW` einfach nur auszugeben, wird man den Benutzernamen und das Passwort auf Gültigkeit überprüfen wollen. Dies kann durch die Abfrage einer Datenbank oder das Einlesen einer Textdatei geschehen.

Vorsicht bei einigen Internet Explorer Versionen - sie scheinen sehr wählerisch zu sein, was die Reihenfolge der Header angeht. Abhilfe schafft hier das Senden des *WWW-Authenticate* Headers vor dem HTTP/1.0 401 zu schaffen.

Um zu unterbinden, daß ein Skript das Passwort einer durch einen traditionellen externen Mechanismus geschützten Seite ausliest, werden die `PHP_AUTH` Variablen nicht gesetzt, wenn eine externe Authentifizierung für diese bestimmte Seite aktiviert ist. In diesem Fall kann die `$REMOTE_USER` Variable benutzt werden, um den Benutzer durch die externe Zugriffskontrolle zu identifizieren.

Zu beachten ist, daß Obenstehendes keinesfalls jemanden, der die Kontrolle über eine nichtgeschützte URL hat, davon abhalten kann, Passwörter von geschützten URLs auf dem gleichen Rechner auszulesen.

Sowohl Netscape als auch der Internet Explorer löschen den lokalen Cache des Authentifizierungsfensters, wenn der Server eine 401-Meldung zurückgibt. Dies kann benutzt werden, um einen Benutzer "auszuloggen" und eine erneute Eingabe des Benutzernamens/Passworts zu erzwingen. Einige nutzen dies, um Logins nach Ablauf einer bestimmten Zeitspanne auslaufen zu lassen oder einen Logout-Button zur Verfügung zu stellen.

Beispiel 16-2 HTTP Authentifizierung, mit erneuter Anforderung von Name/Passwort

```

<?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($PHP_AUTH_USER) || ($SeenBefore == 1 && !strcmp($OldAuth,
$PHP_AUTH_USER)) ) {
    authenticate();
}
else {
    echo "Welcome: $PHP_AUTH_USER<BR>";
    echo "Old: $OldAuth";
    echo "<FORM ACTION=\"\$PHP_SELF\" METHOD=POST>\n";
    echo "<INPUT TYPE=HIDDEN NAME=\"SeenBefore\" VALUE=\"1\">\n";
    echo "<INPUT TYPE=HIDDEN NAME=\"OldAuth\" VALUE=\"\$PHP_AUTH_USER\">\n";
    echo "<INPUT TYPE=Submit VALUE=\"Re Authenticate\">\n";
    echo "</FORM>\n";
}
?>

```

Dieses Verhalten wird vom HTTP Basic A-uthentication Standard nicht gefordert, daher sollte man sich nie darauf verlassen. Tests mit Lynx haben gezeigt, daß Lynx die Authentifizierungsinformationen nicht bei Erhalt einer 401-Meldung löscht. Ein Klick auf den Zurück- Button und danach auf Vorwärts wird die angeforderte Adresse öffnen (und zwar so lange, bis die Identifizierung der Benutzer geändert wird).

Weiterhin muss beachtet werden, daß dies unter dem Microsoft IIS Server mit der CGI-Version von PHP aufgrund einer Einschränkung des IIS nicht funktioniert.

[Zurück](#)
[Creating GIF images](#)
[Anfang](#)
[Hoch](#)
[Vor](#)
[Cookies](#)

Kapitel 17 Cookies

PHP unterstützt HTTP Cookies. Hierbei handelt es sich um einen Mechanismus, um Informationen beim Client zu speichern und somit wiederkehrende Besucher zu identifizieren oder ihren Weg innerhalb des Angebotes nachzuvollziehen. Cookies können durch die Funktion [setcookie\(\)](#) gesetzt werden. Sie sind Bestandteil eines HTTP-Headers, was bedeutet, daß die Funktion [setcookie\(\)](#) aufgerufen werden muß, bevor irgendeine Ausgabe an den Browser erfolgt. Dies ist die gleiche Einschränkung, der auch die Funktion [header\(\)](#) unterliegt.

Vom Client gesendete Cookies werden automatisch in eine Variable geschrieben, wie auch bei GET oder POST. Sollen einem Cookie mehrere Werte zugewiesen werden, muss dem Cookienamen lediglich `[]` angefügt werden. Einzelheiten siehe [setcookie\(\)](#) Funktion.

Kapitel 18 Steuerung von Dateiuploads

Inhaltsverzeichnis

[Dateiuploads mit POST](#)

[Häufige Probleme](#)

[Mehrere Dateien uploaden](#)

[PUT-Unterstützung](#)

Dateiuploads mit POST

PHP kann Dateiuploads mit jedem RFC-1867 konformen Browser (dazu gehören der Netscape Navigator 3 oder höher, Microsoft Internet Explorer 3 mit Patch von Microsoft oder höher ohne Patch) durchführen. Es können sowohl Text- als auch Binärdaten hochgeladen werden. Mit PHP's Authentifizierungs- und Dateifunktionen besteht volle Kontrolle darüber, wer Dateien hochladen darf und was mit den Dateien geschehen soll, wenn der Upload beendet ist.

PHP unterstützt auch Dateiuploads nach der PUT-Methode, die beispielsweise vom Netscape Composer und den W3C Amaya Clients benutzt wird. Siehe dazu [PUT-Unterstützung](#) für nähere Informationen.

Eine Maske für den Dateiuupload kann erstellt werden, indem man ein Formular entwirft, was ungefähr so aussieht:

Beispiel 18-1 Formular für den Dateiuupload

```
<FORM ENCTYPE="multipart/form-data" ACTION="_URL_" METHOD=POST>
<INPUT TYPE="hidden" name="MAX_FILE_SIZE" value="1000">
Send this file: <INPUT NAME="userfile" TYPE="file">
<INPUT TYPE="submit" VALUE="Send File">
</FORM>
```

Die URL sollte auf eine PHP-Datei verweisen. Das versteckte Feld MAX_FILE_SIZE muß dem Dateieingabefeld vorausgehen und den Wert der maximal akzeptierten Dateigröße in Bytes enthalten. Bei erfolgreichem Upload des Files werden folgende Variablen definiert:

- \$userfile - Der temporäre Name, unter dem die hochgeladene Datei auf dem Server gespeichert wurde.
- \$userfile_name - Der ursprüngliche Dateiname auf dem System des Absenders.

- \$userfile_size - Größe der hochgeladenen Datei in Bytes.
- \$userfile_type - Der Mime-Typ der Datei, wenn dies Information zur Verfügung gestellt wird. Ein Beispiel wäre "image/gif".

Die Variable "\$userfile" trägt den Namen des Dateifeldes im Formular; im obigen Beispiel haben wir die Bezeichnung "userfile" gewählt.

Standardmäßig werden Dateien in dem vorgegebenen temporären Verzeichnis des Servers gespeichert. Dies kann durch das Setzen der Umgebungsvariablen TMPDIR direkt in der Umgebung, in der PHP ausgeführt wird, geändert werden. Ein Setzen des temporären Verzeichnisses durch die Funktion [putenv\(\)](#) innerhalb eines Skriptes ist nicht möglich.

Das PHP-Skript, dem die hochzuladende Datei übergeben wird, sollte alle nötigen Anweisungen enthalten, die festlegen, wie mit der hochgeladenen Datei verfahren werden soll. Beispielsweise kann die \$file_size - Variable verwendet werden, um Dateien auszusortieren, die zu gross oder zu klein sind. Oder man benutzt die \$file_type - Variable, um sich aller Dateien zu entledigen, die nicht bestimmten Typen entsprechen. Auf jeden Fall sollte die hochgeladene Datei aus dem temporären Verzeichnis gelöscht oder an andere Stelle verschoben werden.

Die Datei wird aus dem temporären Verzeichnis gelöscht, sobald das Skript abgearbeitet ist, wenn sie nicht verschoben oder umbenannt wurde.

[Zurück](#)

Cookies

[Anfang](#)

[Hoch](#)

[Vor](#)

Häufige Probleme

Häufige Probleme

Der mit `MAX_FILE_SIZE` eingestellte Wert kann nicht grösser sein als der durch die Zeile `upload_max_filesize` der `PHP3.ini` (oder der entsprechenden Zeile `php3_upload_max_filesize` in der Apache Konfigurationsdatei) festgelegte. Standardmäßig sind dies 2 MB.

Bitte beachten, daß der CERN httpd alles abschneidet, was auf das erste Leerzeichen im content-type Mime-Header folgt, den er vom Client erhält. Solange dies der Fall ist, unterstützt der CERN httpd keine Dateiuploads.

Mehrere Dateien uploaden

Es ist möglich, gleichzeitig mehrere Dateien hochzuladen und die Upload-Informationen automatisch in Arrays aufgeführt zu erhalten. Dabei muß im HTML-Formular die gleiche Syntax zum Übermitteln des Arrays benutzt werden, was man durch verschiedene Select- und Checkboxes erreicht.

Anmerkung: Die Unterstützung für den Upload mehrerer Dateien wurde in Version 3.0.10 hinzugefügt.

Beispiel 18-2 Formular zum Upload mehrerer Dateien

```
<form action="file-upload.html" 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>
```

Wenn das obige Formular abgeschickt wird, werden die Arrays `$userfile`, `$userfile_name` und `$userfile_size` für den globalen Bereich, wie in `$HTTP_POST_VARS`, angeordnet. Jede dieser Variablen ist ein numerisch indizierte Array, das aus den entsprechenden Werten der übertragenen Dateien besteht.

Nehmen wir an, die Dateinamen `/home/test/review.html` und `/home/test/xwp.out` werden übertragen. In diesem Fall beinhaltet `$userfile_name[0]` den Wert `review.html` und `$userfile_name[1]` enthält den Wert `xwp.out`. Genauso enthält `$userfile_size[0]` die Dateigröße von `review.html`, und so weiter.

PUT-Unterstützung

PHP unterstützt die HTTP PUT-Methode, wie sie von Clients wie Netscape Composer oder W3C Amaya benutzt wird. PUT-Anfragen sind weitaus unkomplizierter als ein Dateiupload und sehen etwa so aus:

```
PUT /path/filename.html HTTP/1.1
```

Das würde normalerweise bedeuten, daß der entfernte Client den folgenden Inhalt als /path/filename.html im Webverzeichnisbaum speichert. Natürlich ist es keine gute Idee, dass PHP oder Apache jeden Benutzer beliebige Dateien überschreiben lassen. Um eine solche Anfrage bearbeiten zu können, muss der Webserver erst angewiesen werden, ein bestimmtes PHP-Skript für die Abarbeitung aufzurufen. Im Apache wird dies durch die *Script* - Direktive festgelegt. Sie kann irgendwo in der Apache-Konfigurationsdatei plziert werden, gebräuchlich ist die Plazierung innerhalb einer <Directory>- oder <Virtualhost>- Sektion. Eine Zeile wie die folgende erledigt dies:

```
Script PUT /put.php3
```

Diese Zeile legt fest, daß Apache alle PUT-Anfragen für URIs, die dem Kontext entsprechen, in dem diese Zeile steht, an das put.php3 Skript weiterleitet. Dies setzt voraus, daß PHP aktiv und für die .php3-Dateierweiterung registriert ist.

Innerhalb der put.php3-Datei könnte folgendes stehen:

```
<? copy($PHP_UPLOADED_FILE_NAME,$DOCUMENT_ROOT.$REQUEST_URI); ?>
```

Dies kopiert die Datei an den vom Client angegebenen Ort. Möglicherweise sollen vor dem Kopieren der Datei noch Überprüfungen und/oder Benutzerauthentifizierung stattfinden. Wenn PHP eine PUT-Anfrage erhält, wird die Datei in einem temporären Verzeichnis gespeichert, genau wie bei [POST-Methode](#). Nach Abarbeiten der Anfrage wird die temporäre Datei gelöscht. Also muß das PHP-Skript, das die PUT-Anfrage bearbeitet, die Datei irgendwohin kopieren. Der Name der temporären Datei findet sich in der Variablen \$PHP_PUT_FILENAME; der Name der zu speichernden Datei steht in \$REQUEST_URI (kann bei Nicht-Apache Webservern variieren). Dieser Zielfeldname wird vom Client festgelegt. Man kann den Client auch umgehen; beispielsweise könnten alle hochgeladenen Dateien in ein spezielles Upload-Directory kopiert werden.

Kapitel 19 Using remote files

As long as support for the "URL fopen wrapper" is enabled when you configure PHP (which it is unless you explicitly pass the `--disable-url-fopen-wrapper` flag to configure), you can use HTTP and FTP URLs with most functions that take a filename as a parameter, including the [require\(\)](#) and [include\(\)](#) statements.

Anmerkung: You can't use remote files in [include\(\)](#) and [require\(\)](#) statements on Windows.

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.

Beispiel 19-1 Getting the title of a remote page

```
<?php
$file = fopen("http://www.php.net/", "r");
if (!$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 as long you connect as a user with the correct access rights, and the file doesn't exist already. 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.)

Beispiel 19-2 Storing data on a remote server

```
<?php
$file = fopen("ftp://ftp.php.net/incoming/outputfile", "w");
if (!$file) {
    echo "<p>Unable to open remote file for writing.\n";
    exit;
}
/* Write the data here. */
fputs($file, "$HTTP_USER_AGENT\n");
fclose($file);
?>
```

Anmerkung: You might get the idea from the example above to use this technique to write to a remote log, but as mentioned above, you can only write to a new file using the URL fopen() wrappers. To do distributed logging like that, you should take a look at [syslog\(\)](#).

[Zurück](#)

PUT-Unterstützung

[Anfang](#)[Hoch](#)[Vor](#)

Verbindungssteuerung

Kapitel 20 Verbindungssteuerung

Anmerkung: Folgendes trifft auf die Versionen 3.0.7 und später zu.

PHP erhält intern einen Verbindungsstatus. Dieser kann drei Zustände annehmen:

- 0 - NORMAL
- 1 - ABORTED
- 2 - TIMEOUT

Wenn ein PHP-Skript aktiv ist, ist der Status üblicherweise NORMAL. Sollte der Client-Rechner die Verbindung beenden, wird der Status auf ABORTED gesetzt (Ein clientseitiges Beenden der Verbindung wird für gewöhnlich veranlaßt, wenn der Benutzer den STOP-Button seines Browsers drückt). Wenn das eingestellte Zeitlimit (siehe [set_time_limit\(\)](#)) überschritten wird, wird der Status TIMEOUT gesetzt.

Man kann entscheiden, ob der Verbindungsabbruch seitens eines Clients den Abbruch des Skriptes zur Folge haben soll. Manchmal ist es sinnvoll, Skripte sauber zu beenden, auch wenn kein Browser mehr zur Verfügung steht, der die Ausgabe empfängt. Die Abarbeitung eines Skriptes wird standardmäßig abgebrochen, wenn der Client die Verbindung beendet. Dieses Verhalten kann sowohl durch die Option `ignore_user_abort` in der Konfigurationsdatei `php3.ini`, durch die entsprechende Option `php3_ignore_user_abort` in der Apache-Konfigurationsdatei als auch durch [ignore_user_abort\(\)](#) beeinflusst werden. Wenn PHP nicht angewiesen wird, einen Verbindungsabbruch durch den Benutzer zu ignorieren und die Verbindung dann durch den Benutzer beendet wird, wird die Abarbeitung des Skriptes abgebrochen. Die einzige Ausnahme ist, wenn durch die Funktion [register_shutdown_function\(\)](#) eine Shutdown-Funktion angegeben wird, die bei clientseitigem Abbruch ausgeführt wird. Wenn dann der Benutzer den STOP-Button seines Browsers drückt, wird PHP bei der nächsten Ausgabe des Skriptes feststellen, daß die Verbindung abgebrochen wurde und die Shutdown-Funktion aufrufen. Diese Shutdown-Funktion wird auch aufgerufen, wenn das Skript auf normalem Wege beendet wird, daher sollte man, wenn man für den Fall eines Benutzerabbruchs etwas anderes vorgesehen hat, die Funktion **connection_aborted()** verwenden. Sie gibt true zurück, wenn die Verbindung abgebrochen wurde.

Ein Skript kann ebenfalls durch den eingebauten Script-Timer beendet werden. Der Standard-Timeout beträgt 30 Sekunden. Er kann durch die Option `max_execution_time` in der `php3.ini`, durch den entsprechenden Eintrag `php3_max_execution_time` in der Apache-Konfigurationsdatei oder durch die Funktion [set_time_limit\(\)](#) beeinflusst werden. Bei Zeitüberschreitung wird das Skript beendet und, genau wie im obigen Fall des Verbindungsabbruchs, eine registrierte Shutdown-Funktion ausgeführt. Um zu überprüfen, ob es sich um einen Abbruch aufgrund von Zeitüberschreitung handelt, kann die Funktion [connection_timeout\(\)](#) benutzt werden. Sie gibt true zurück, wenn es sich um eine Zeitüberschreitung handelt.

Zu bemerken ist, daß der ABORTED und der TIMEOUT-Status gleichzeitig auftreten können. Dies ist

möglich, wenn PHP angewiesen wird, Benutzerabbrüche zu ignorieren. PHP wird feststellen, daß der Benutzer die Verbindung abgebrochen hat, das Skript allerdings läuft weiter. Sollte es dann das Zeitlimit erreichen, wird es abgebrochen und eine Shutdown-Funktion, wenn definiert, wird aufgerufen. Zu diesem Zeitpunkt kann man feststellen, dass [connection_timeout\(\)](#) und [connection_aborted\(\)](#) true zurückgeben. Diese beiden Statusmöglichkeiten können auch durch einen Aufruf der Funktion [connection_status\(\)](#) abgefragt werden. Sie liefert ein Bitfeld des aktiven Status. Wenn beispielsweise TIMEOUT und ABORTED aktiv sind, wird 3 zurückgegeben.

[Zurück](#)

Using remote files

[Anfang](#)

[Hoch](#)

[Vor](#)

Persistente
Datenbankverbindungen

Kapitel 21 Persistente Datenbankverbindungen

Persistente Verbindungen sind SQL-Verbindungen, die nach Abarbeitung des Skriptes nicht geschlossen werden. Wenn eine persistente Verbindung angefordert wird, prüft PHP zuerst, ob bereits eine identische persistente Verbindung (die vielleicht vorher offen geblieben ist) existiert und benutzt sie in diesem Fall. Sollte keine Verbindung existieren, wird eine hergestellt. Eine 'identische' Verbindung ist eine Verbindung, die zu dem gleichen Host mit dem gleichen Usernamen und Passwort hergestellt wurde.

Wer nicht durchgängig mit der Art und Weise, wie Webserver arbeiten und die Last verteilen, vertraut ist, könnte missverstehen, wofür persistente Verbindungen gedacht sind. Im Besonderen bieten sie *keine* Möglichkeit, 'Benutzersitzungen' auf der gleichen SQL-Verbindung zu öffnen und *keine* Möglichkeit, eine leistungsstarke Transaktion aufzubauen, und sie können viele andere Sachen nicht. Um absolute Klarheit zu schaffen: Persistente Verbindungen bieten *keine* Funktionalität, die nicht auch von nicht-persistenten Verbindungen bereitgestellt wird.

Warum?

Das hat mit der Arbeitsweise von Webservern zu tun. Es gibt drei Möglichkeiten, wie ein Webserver PHP zur Generierung von Webseiten einsetzen kann.

Die erste Methode ist, PHP als CGI-'Wrapper' zu benutzen. Wenn diese Methode eingesetzt wird, wird für jede Anfrage nach einer PHP-Seite vom Webserver eine Instanz des PHP- Interpreters gestartet und anschliessend wieder beendet. Durch die Beendigung des Interpreters nach abgeschlossener Anfrage werden alle Ressourcen, auf die zugegriffen wurde (wie beispielsweise eine Verbindung zu einem SQL-Datenbankserver) wieder geschlossen. In diesem Fall erreicht man nichts, wenn man persistente Verbindungen benutzt - sie sind eben nicht beständig.

Die zweite und populärere Methode ist der Einsatz von PHP als Modul in einem Multiprozess-Webserver, was momentan nur auf den Apache zutrifft. Typischerweise hat ein Multiprozess- Webserver einen Prozess (den 'Eltern' Prozess), der einen Satz weiterer Prozesse (die 'Kinder') koordiniert, die die eigentliche Arbeit des Bereitstellens der Seiten übernehmen. Jede Anfrage, die von einem Client erfolgt, wird an einen untergeordneten Prozess, der noch keine andere Anfrage bearbeitet, weitergereicht. Das bedeutet, dass eine zweite Anfrage des Clients an den Server unter Umständen von einem anderen untergeordneten Prozess als die erste Anfrage bearbeitet wird. In diesem Fall sorgt eine persistente Verbindung dafür, dass jeder untergeordnete Prozess sich nur einmal mit dem SQL-Server verbinden muss, wenn eine solche benötigt wird. Benötigt dann eine weitere Seite die Verbindung mit dem SQL-Server, kann auf die zurückgegriffen werden, die der untergeordnete Prozess vorher hergestellt hat.

Die letzte Methode ist, PHP als Plug-in für einen Multithread- Webserver zu benutzen. Momentan ist

dies noch Theorie - PHP arbeitet noch nicht als Plug-in für irgendeinen dieser Server. An der Unterstützung für ISAPI, WSAPI und NSAPI wird gearbeitet, so dass die Nutzung von PHP mit Multithread-Serven wie Netscape Fast Track, Microsoft Internet Information Server (IIS) und O'Reilly's WebSite Pro möglich wird. Wenn es soweit ist, wird das Verhalten im wesentlichen dem oben beschriebenen Multiprozess-Modell entsprechen.

Wozu dienen persistente Verbindungen, wenn sie keine zusätzliche Funktionalität bieten?

Die Antwort ist ausserordentlich einfach: Effizienz. Persistente Verbindungen sind nützlich, wenn die Notwendigkeit, eine Verbindung zu einem SQL-Server herzustellen, hoch ist. Ob dies der Fall ist oder nicht, hängt von vielen Faktoren ab - zum Beispiel, um was für eine Datenbank es sich handelt, ob sie auf dem gleichen Rechner wie der Webserver läuft oder welche Last die SQL-Maschine zu bewältigen hat usw. Grundsätzlich gilt, dass, wenn viele Verbindungen hergestellt werden müssen, persistente Verbindungen ausserordentlich hilfreich sind. Sie veranlassen den untergeordneten Prozess, sich während seiner gesamten Lebensdauer lediglich einmal mit dem SQL-Server zu verbinden, anstatt jedesmal beim Aufruf einer Seite, die eine Verbindung benötigt. Das heisst, dass jeder untergeordnete Prozess, der eine persistente Verbindung öffnet, eine eigene dauerhafte Verbindung zum Server hat. Bei 20 untergeordnete Prozessen, die ein Skript ausführen, das eine persistente Verbindung zum SQL-Server herstellt, hat man beispielsweise 20 verschiedene Verbindungen zum SQL-Server - eine für jeden untergeordneten Prozess.

Eine wichtige Zusammenfassung. Persistente Verbindungen wurden entwickelt, um eins-zu-eins Abbildungen auf reguläre Verbindungen zu haben. Das heisst, dass man *immer* in der Lage sein sollte, die persistenten Verbindungen durch nicht-persistente zu ersetzen, ohne dass dies den Skriptablauf verändert. Es *kann* (und wird vermutlich auch) die Effizienz des Skriptes beeinflussen, aber nicht dessen Verhalten.

[Zurück](#)

Verbindungssteuerung

[Anfang](#)

[Hoch](#)

[Vor](#)

Funktionsreferenz

IV Funktionsreferenz

Inhaltsverzeichnis

- I. [Apache-spezifische Funktionen](#)
- II. [Array Funktionen](#)
- III. [Aspell Funktionen](#)
- IV. [Mathematische Funktionen mit beliebiger Genauigkeit](#)
- V. [Kalender-Funktionen](#)
- VI. [COM support functions for Windows](#)
- VII. [Klassen- und Objekt Funktionen](#)
- VIII. [ClibPDF functions](#)
- IX. [Cybercash payment functions](#)
- X. [Database \(dbm-style\) abstraction layer functions](#)
- XI. [Datums- und Zeit-Funktionen](#)
- XII. [dBase Funktionen](#)
- XIII. [dbm functions](#)
- XIV. [Verzeichnis-Funktionen](#)
- XV. [Dynamisch geladene Bibliothek](#)
- XVI. [DOM XML functions](#)
- XVII. [filePro functions](#)
- XVIII. [Funktionen des Dateisystems](#)
- XIX. [Forms Data Format functions](#)
- XX. [FTP-Funktionen](#)
- XXI. [GNU Gettext](#)
- XXII. [HTTP-Funktionen](#)
- XXIII. [Hyperwave functions](#)
- XXIV. [ICAP Functions](#)
- XXV. [Grafik-Funktionen](#)
- XXVI. [IMAP, POP3 und NNTP Funktionen](#)
- XXVII. [Informix functions](#)

- XXVIII. [InterBase-Funktionen](#)
- XXIX. [LDAP functions](#)
- XXX. [Mail Funktionen](#)
- XXXI. [Mathematische Funktionen](#)
- XXXII. [MCAL functions](#)
- XXXIII. [Encryption functions](#)
- XXXIV. [Hash functions](#)
- XXXV. [Microsoft SQL Server functions](#)
- XXXVI. [Sonstige Funktionen](#)
- XXXVII. [mSQL functions](#)
- XXXVIII. [MySQL Funktionen](#)
- XXXIX. [Netzwerk Funktionen](#)
- XL. [ODBC Funktionen](#)
- XLI. [Oracle functions](#)
- XLII. [Oracle 8 functions](#)
- XLIII. [PDF Funktionen](#)
- XLIV. [Payflow Pro functions](#)
- XLV. [PHP Optionen und Informationen](#)
- XLVI. [POSIX functions](#)
- XLVII. [PostgreSQL Funktionen](#)
- XLVIII. [Program Execution functions](#)
- XLIX. [Pspell Functions](#)
- L. [GNU Recode functions](#)
- LI. [Perl-compatible Regular Expression functions](#)
- LII. [Regular expression functions](#)
- LIII. [Semaphor und Shared Memory Funktionen](#)
- LIV. [Session Funktionen](#)
- LV. [Shockwave Flash functions](#)
- LVI. [SNMP functions](#)
- LVII. [String-Funktionen](#)
- LVIII. [Sybase functions](#)
- LIX. [URL functions](#)
- LX. [Variablen-Functions](#)

LXI. [Vmailmgr functions](#)

LXII. [WDDX functions](#)

LXIII. [XML parser functions](#)

LXIV. [NIS functions](#)

LXV. [Compression functions](#)

[Zurück](#)

[Anfang](#)

[Vor](#)

Persistente

Apache-spezifische Funktionen

Datenbankverbindungen

I. Apache-spezifische Funktionen

Inhaltsverzeichnis

[apache_lookup_uri](#) Führt eine Teilanfrage für eine URI durch und liefert alle Informationen darüber zurück

[apache_note](#) Setzt und liest Apache Notes

[getallheaders](#) Liefert alle HTTP-Header der aktuellen Anfrage aus

[virtual](#) Führt eine Apache-Unteranfrage durch

apache_lookup_uri

(PHP3 >= 3.0.4, PHP4)

apache_lookup_uri -- Führt eine Teilanfrage für eine URI durch und liefert alle Informationen darüber zurück

Description

`class apache_lookup_uri (string filename)`

Apache_lookup_uri() führt eine Teilanfrage über eine bestimmte URI durch. Die Abfrage geht gerade soweit, dass alle wichtigen Informationen über die gegebene Quelle gesammelt werden können. Als Resultat wird eine Klasse zurückgeliefert, die folgende Elemente enthält:

- 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

Anmerkung: **Apache_lookup_uri()** existiert nur, wenn PHP als Apache-Modul

eingrichtet wurde.

[Zurück](#)

Apache-spezifische Funktionen

[Anfang](#)

[Hoch](#)

[Vor](#)

apache_note

apache_note

(PHP3 >= 3.0.2, PHP4)

apache_note -- Setzt und liest Apache Notes

Description

string **apache_note** (string note_name [, string note_value])

Apache_note() ist eine Apache-spezifische Funktion, die Einträge in Apaches `notes`-Tabelle setzt und ausliest. Wenn sie mit einem einzelnen Parameter aufgerufen wird, gibt sie den aktuellen Wert der Note *note_name* zurück. Bei zwei Parametern wird der Eintrag *note_name* auf den zweiten Parameter *note_value* gesetzt und liefert den bisherigen Wert zurück.

getallheaders

(PHP3 , PHP4)

getallheaders -- Liefert alle HTTP-Header der aktuellen Anfrage aus

Description

array **getallheaders**(void);

Diese Funktion liefert ein assoziatives Array zurück, das alle HTTP-Header-Informationen der aktuellen HTTP-Anfrage enthält.

Anmerkung: Die Werte der normalen CGI-Variablen können Sie auch durch das Auslesen der Umgebungsvariablen erhalten. Dieser Weg funktioniert unabhängig vom Apache. Probieren Sie [phpinfo\(\)](#) aus, um alle Umgebungsvariablen, die auf diese Weise erreichbar sind, zu erhalten.

Beispiel 1 Getallheaders() Example

```
$headers= getallheaders();
while (list ($header, $value) = each ($headers)) {
    echo "$header: $value<br>\n";
}
```

Dieses Beispiel gibt alle HTTP-Header der aktuellen Anfrage aus.

Anmerkung: **Getallheaders()** wird zur Zeit nur durch PHP als Apachemodul unterstützt.

virtual

(PHP3 , PHP4)

virtual -- Führt eine Apache-Unteranfrage durch

Description

int **virtual** (string filename)

Virtual() ist eine Apache-spezifische Funktion, die `<!--#include virtual... -->` aus `mod_include` entspricht. Sie führt eine Unteranfrage durch, die für das Einbinden von CGI-Skripten, `.shtml`-Dateien oder anderes, das vom Apache verarbeitet werden soll, nützlich ist. Beachten Sie bitte, dass ein CGI-Skript mindestens einen gültigen CGI-Header generieren muß, mindestens einen Content-type-Header. Für PHP-Dateien müssen Sie [include\(\)](#) oder [require\(\)](#) benutzen, **virtual()** kann selbst keine PHP-Dokumente einbinden.

II. Array Funktionen

Inhaltsverzeichnis

- [array](#) Einen Array erstellen
- [array_count_values](#) Counts all the values of an array
- [array_flip](#) Flip all the values of an array
- [array_keys](#) Return all the keys of an array
- [array_merge](#) Merge two or more arrays
- [array_merge_recursive](#) Merge two or more arrays recursively
- [array_multisort](#) Sort multiple or multi-dimensional arrays
- [array_pad](#) Pad array to the specified length with a value
- [array_pop](#) Pop the element off the end of array
- [array_push](#) Push one or more elements onto the end of array
- [array_rand](#) Pick one or more random entries out of an array
- [array_reverse](#) Return an array with elements in reverse order
- [array_shift](#) Pop an element off the beginning of array
- [array_slice](#) Extract a slice of the array
- [array_splice](#) Remove a portion of the array and replace it with something else
- [array_unshift](#) Push one or more elements onto the beginning of array
- [array_values](#) Return all the values of an array
- [array_walk](#) Apply a user function to every member of an array.
- [arsort](#) Sort an array in reverse order and maintain index association
- [asort](#) Sort an array and maintain index association
- [compact](#) Create array containing variables and their values
- [count](#) count elements in a variable
- [current](#) Return the current element in an array
- [each](#) Return the next key and value pair from an array
- [end](#) Set the internal pointer of an array to its last element
- [extract](#) Import variables into the symbol table from an array
- [in_array](#) Return true if a value exists in an array

- [key](#) Fetch a key from an associative array
- [krsort](#) Sort an array by key in reverse order
- [ksort](#) Sort an array by key
- [list](#) Assign variables as if they were an array
- [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 integers
- [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 an array
- [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

[Zurück](#)

virtual

[Anfang](#)

[Hoch](#)

[Vor](#)

array

array

(unknown)

array -- Einen Array erstellen

Beschreibung

array **array**(...);

Gibt einen Array von Parametern zurück. Mit dem => Operator können diese Parameter indeziert werden.

Anmerkung: **array()** ist ein Sprachkonstrukt, das benutzt wird um ein buchstäbliches Wertfeld zu erzeugen, und nicht eine reguläre Funktion.

Das folgende Beispiel zeigt, wie man einen zwei-dimensionalen Array erzeugt, wie man Schlüssel für assoziative Arrays festlegt und wie man in normalen Arrays numerische Indizes überspringt und fortsetzt.

Beispiel 1 array()-Beispiel

```
$fruits = array(
    "fruechte" => array("a"=>"orange", "b"=>"banane", "c"=>"apfel"),
    "zahlen" => array(1, 2, 3, 4, 5, 6),
    "luecken" => array("erstes", 5 => "zweites", "drittes")
);
```

See also: [list\(\)](#).

array_count_values

(PHP4 >= 4.0b4)

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.

Beispiel 1 array_count_values() example

```
$array = array(1,"hello",1,"world","hello");  
array_count_values($array);           // returns array(1=>2, "hello"=>2, "world"=>1)
```

Anmerkung: This function was added in PHP 4.0.

array_flip

(PHP4 >= 4.0b4)

array_flip -- Flip all the values of an array

Description

array **array_flip** (array trans)

array_flip() returns an array in flip order.

Beispiel 1 array_flip() example

```
$trans = array_flip ($trans);  
$original = strtr ($str, $trans);
```

Anmerkung: This function was added in PHP 4.0.

array_keys

(PHP4)

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.

Beispiel 1 array_keys() example

```
$array = array(0 => 100, "color" => "red");  
array_keys($array);           // returns array(0, "color")  
  
$array = array(1, 100, 2, 100);  
array_keys($array, 100); // returns array(0, 2)
```

See also [array_values\(\)](#).

Anmerkung: This function was added in PHP 4.0.

array_merge

(PHP4)

array_merge -- Merge two or more arrays

Description

array **array_merge** (array array1, array array2 [, ...])

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 had the same string keys, then the later value for that key will overwrite previous one. If, however, the arrays have the same numeric key, this does not happen since the values are appended.

Beispiel 1 array_merge() example

```
$array1 = array("color" => "red", 2, 4);  
$array2 = array("a", "b", "color" => "green", "shape" => "trapezoid");  
array_merge($array1, $array2);
```

Resulting array will be array("color" => "green", 2, 4, "a", "b", "shape" => "trapezoid").

Anmerkung: This function was added in PHP 4.0.

array_merge_recursive

(PHP4 >= 4.0.1)

array_merge_recursive -- Merge two or more arrays recursively

Description

array **array_merge_recursive** (array array1, array array2 [, ...])

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.

Beispiel 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);
```

Resulting array will be array("color" => array("favorite" => array("red", "green"), "blue"), 5, 10).

See also [array_merge\(\)](#).

array_multisort

(PHP4 >= 4.0b4)

array_multisort -- Sort multiple or multi-dimensional arrays

Description

bool **array_multisort** (array ar1 [, mixed arg [, ...]])

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 sortings flags specified after an array argument apply only to that array - they are reset to default SORT_ASC and SORT_REGULAR after before each new array argument.

Returns true on success, false on failure.

Beispiel 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.

Beispiel 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).

[Zurück](#)[array_merge_recursive](#)[Anfang](#)[Hoch](#)[Vor](#)[array_pad](#)

array_pad

(PHP4 >= 4.0b4)

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.

Beispiel 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

(PHP4)

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.

Beispiel 1 array_pop() example

```
$stack = array("orange", "apple", "raspberry");  
$fruit = array_pop($stack);
```

After this, \$stack has only 2 elements: "orange" and "apple", and \$fruit has "raspberry".

See also [array_push\(\)](#), [array_shift\(\)](#), and [array_unshift\(\)](#).

Anmerkung: This function was added in PHP 4.0.

array_push

(PHP4)

`array_push` -- Push one or more elements onto the end of array

Description

`int array_push (array array, mixed var [, ...])`

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.

Beispiel 1 array_push() example

```
$stack = array(1, 2);  
array_push($stack, "+", 3);
```

This example would result in `$stack` having 4 elements: 1, 2, "+", and 3.

See also: [array_pop\(\)](#), [array_shift\(\)](#), and [array_unshift\(\)](#).

Anmerkung: This function was added in PHP 4.0.

array_rand

(PHP4 >= 4.0.0)

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.

Beispiel 1 array_rand() example

```
srand((double)microtime() * 100000000);
$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_reverse

(PHP4 >= 4.0b4)

array_reverse -- Return an array with elements in reverse order

Description

array **array_reverse** (array array)

array_reverse() takes input *array* and returns a new array with the order of the elements reversed.

Beispiel 1 array_reverse() example

```
$input = array("php", 4.0, array("green", "red"));  
$result = array_reverse($input);
```

This makes \$result have array(array("green", "red"), 4.0, "php").

Anmerkung: This function was added in PHP 4.0 Beta 3.

array_shift

(PHP4)

array_shift -- Pop 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.

Beispiel 1 array_shift() example

```
$args = array("-v", "-f");  
$opt = array_shift($args);
```

This would result in \$args having one element "-f" left, and \$opt being "-v".

See also [array_unshift\(\)](#), [array_push\(\)](#), and [array_pop\(\)](#).

Anmerkung: This function was added in PHP 4.0.

array_slice

(PHP4)

array_slice -- Extract a slice of the array

Description

array **array_slice** (array *array*, int *offset* [, int *length*])

array_slice() returns a sequence of elements from the *array* 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*.

Beispiel 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\(\)](#).

Anmerkung: This function was added in PHP 4.0.

array_splice

(PHP4)

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() removed the elements designated by *offset* and *length* from the *input* array, and replaces them with the elements of the *replacement* array, if supplied.

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:

<code>array_push(\$input, \$x, \$y)</code>	<code>array_splice(\$input, count(\$input), 0, array(\$x, \$y))</code>
<code>array_pop(\$input)</code>	<code>array_splice(\$input, -1)</code>
<code>array_shift(\$input)</code>	<code>array_splice(\$input, 0, 1)</code>
<code>array_unshift(\$input, \$x, \$y)</code>	<code>array_splice(\$input, 0, 0, array(\$x, \$y))</code>
<code>\$a[\$x] = \$y</code>	<code>array_splice(\$input, \$x, 1, \$y)</code>

Returns the array consisting of removed elements.

Beispiel 1 array_splice() examples

```
$input = array("red", "green", "blue", "yellow");

array_splice($input, 2);           // $input is now array("red", "green")
array_splice($input, 1, -1);       // $input is now array("red", "yellow")
array_splice($input, 1, count($input), "orange");
                                   // $input is now array("red", "orange")
array_splice($input, -1, 1, array("black", "maroon"));
                                   // $input is now array("red", "green",
                                   //                      "blue", "black", "maroon")
```

See also [array_slice\(\)](#).

Anmerkung: This function was added in PHP 4.0.

[Zurück](#)

array_slice

[Anfang](#)

[Hoch](#)

[Vor](#)

array_unshift

array_unshift

(PHP4)

array_unshift -- Push one or more elements onto the beginning of array

Description

int **array_unshift** (array array, mixed var [, ...])

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.

Returns the new number of elements in the *array*.

Beispiel 1 array_unshift() example

```
$queue = array("p1", "p3");  
array_unshift($queue, "p4", "p5", "p6");
```

This would result in \$queue having 5 elements: "p4", "p5", "p6", "p1", and "p3".

See also [array_shift\(\)](#), [array_push\(\)](#), and [array_pop\(\)](#).

Anmerkung: This function was added in PHP 4.0.

array_values

(PHP4)

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.

Beispiel 1 array_values() example

```
$array = array("size" => "XL", "color" => "gold");  
array_values($array);    // returns array("XL", "gold")
```

Anmerkung: This function was added in PHP 4.0.

array_walk

(PHP3 >= 3.0.3, PHP4)

`array_walk` -- Apply a user function to every member of an array.

Description

`int array_walk (array arr, string func, mixed userdata)`

Applies the function named by *func* to each element of *arr*. *func* will be passed array value as the first parameter and array key as the second parameter. If *userdata* is supplied, it will be passed as the third parameter to the user function. If *func* requires more than two or three arguments, depending on *userdata*, a warning will be generated each time `array_walk()` calls *func*. These warnings may be suppressed by prepending the '@' sign to the `array_walk()` call, or by using [error_reporting\(\)](#).

Anmerkung: If *func* needs to be working with the actual values of the array, specify that the first parameter of *func* should be passed by reference. Then any changes made to those elements will be made in the array itself.

Anmerkung: Passing the key and userdata to *func* was added in 4.0.

Beispiel 1 array_walk() example

```
$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";
}

array_walk( $fruits, 'test_print' );
array_walk( $fruits, 'test_alter', 'fruit' );
array_walk( $fruits, 'test_print' );
```

See also [each\(\)](#) and [list\(\)](#).

array_walk

array_values

[Hoch](#)

arsort

arsort

(PHP3 , PHP4)

arsort -- Sort an array in reverse order and maintain index association

Description

`void arsort (array array)`

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.

Beispiel 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: `fruits[a] = orange fruits[d] = lemon fruits[b] = banana fruits[c] = apple` The fruits have been sorted in reverse alphabetical order, and the index associated with each element has been maintained.

See also: [asort\(\)](#), [rsort\(\)](#), [ksort\(\)](#), and [sort\(\)](#).

asort

(PHP3 , PHP4)

asort -- Sort an array and maintain index association

Description

void **asort** (array array)

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.

Beispiel 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: fruits[c] = apple fruits[b] = banana fruits[d] = lemon fruits[a] = orange The fruits have been sorted in alphabetical order, and the index associated with each element has been maintained.

See also [arsort\(\)](#), [rsort\(\)](#), [ksort\(\)](#), and [sort\(\)](#).

compact

(PHP4)

compact -- Create array containing variables and their values

Description

array **compact** (string varname | array varnames [, ...])

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.

Beispiel 1 compact() example

```
$city = "San Francisco";  
$state = "CA";  
$event = "SIGGRAPH";  
  
$location_vars = array("city", "state");  
  
$result = compact("event", $location_vars);
```

After this, \$result will be array("event" => "SIGGRAPH", "city" => "San Francisco", "state" => "CA").

See also [extract\(\)](#).

Anmerkung: This function was added in PHP 4.0.

count

(PHP3 , PHP4)

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).

Returns 1 if the variable is not an array.

Returns 0 if the variable is not set.

Warnung
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.

See also: [sizeof\(\)](#), [isset\(\)](#), and [is_array\(\)](#).

current

(PHP3 , PHP4)

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.

Warnung

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

(PHP3 , PHP4)

each -- Return the next key and value pair from an array

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.

Beispiel 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,

each

\$HTTP_POST_VARS:

Beispiel 2 Traversing \$HTTP_POST_VARS with each()

```
echo "Values submitted via POST method:<br>";
while (list($key, $val) = each($HTTP_POST_VARS)) {
    echo "$key => $val<br>";
}
```

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.

See also [key\(\)](#), [list\(\)](#), [current\(\)](#), [reset\(\)](#), [next\(\)](#), and [prev\(\)](#).

[Zurück](#)

current

[Anfang](#)

[Hoch](#)

[Vor](#)

end

end

(PHP3 , PHP4)

end -- Set the internal pointer of an array to its last element

Description

end (array array)

end() advances *array*'s internal pointer to the last element.

See also: [current\(\)](#), [each\(\)](#), **end()**, [next\(\)](#), and [reset\(\)](#).

extract

(PHP3 >= 3.0.7, PHP4)

extract -- Import variables into the symbol table from an array

Description

void **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 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.

extract() checks for collisions with existing variables. The way collisions are treated is determined by *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 new variable with *prefix*.

EXTR_PREFIX_ALL

Prefix all variables with *prefix*.

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 or EXTR_PREFIX_ALL.

extract() checks each key to see if it constitutes a valid variable name, and if it does only then does it proceed to import it.

A possible use for **extract** is to import into symbol table variables contained in an associative array returned by [wddx_deserialize\(\)](#).

Beispiel 1 extract example

```
<?
/* 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.

[Zurück](#)

end

[Anfang](#)

[Hoch](#)

[Vor](#)

in_array

in_array

(PHP4)

in_array -- Return true if a value exists in an array

Description

bool in_array (mixed needle, array haystack)

Searches *haystack* for *needle* and returns true if it is found in the array, false otherwise.

Beispiel 1 in_array() example

```
$os = array("Mac", "NT", "Irix", "Linux");  
if (in_array("Irix", $os))  
    print "Got Irix";
```

Anmerkung: This function was added in PHP 4.0.

key

(PHP3 , PHP4)

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\(\)](#), [next\(\)](#)

krsort

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

krsort -- Sort an array by key in reverse order

Description

int krsort (array array)

Sorts an array by key in reverse order, maintaining key to data correlations. This is useful mainly for associative arrays.

Beispiel 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: fruits[d] = lemon fruits[c] = apple fruits[b] = banana fruits[a] = orange

See also [asort\(\)](#), [arsort\(\)](#), [ksort\(\)](#) [sort\(\)](#), and [rsort\(\)](#).

ksort

(PHP3 , PHP4)

ksort -- Sort an array by key

Description

int **ksort** (array array)

Sorts an array by key, maintaining key to data correlations. This is useful mainly for associative arrays.

Beispiel 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: fruits[a] = orange fruits[b] = banana fruits[c] = apple fruits[d] = lemon

See also [asort\(\)](#), [arsort\(\)](#), [sort\(\)](#), and [rsort\(\)](#).

list

(unknown)

list -- Assign variables as if they were an array

Description

void **list**(...);

Like [array\(\)](#), this is not really a function, but a language construct. **list()** is used to assign a list of variables in one operation.

Beispiel 1 list() example

```
<table>
<tr>
  <th>Employee name</th>
  <th>Salary</th>
</tr>
<?php

$result = mysql($conn, "SELECT id, name, salary FROM employees");
while (list($id, $name, $salary) = mysql_fetch_row($result)) {
    print(" <tr>\n".
        "   <td><a href=\"info.php3?id=$id\">$name</a></td>\n".
        "   <td>$salary</td>\n".
        " </tr>\n");
}

?></table>
```

See also: [each\(\)](#), [array\(\)](#).

next

(PHP3 , PHP4)

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.

Warnung
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.

See also: [current\(\)](#), [end\(\)](#) [prev\(\)](#) and [reset\(\)](#)

pos

(PHP3 , PHP4)

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

(PHP3 , PHP4)

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.

Warnung
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

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

range -- Create an array containing a range of integers

Description

array **range** (int low, int high)

range() returns an array of integers from *low* to *high*, inclusive.

See [shuffle\(\)](#) for an example of its use.

reset

(PHP3 , PHP4)

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\(\)](#), [prev\(\)](#), and **reset()**.

rsort

(PHP3 , PHP4)

rsort -- Sort an array in reverse order

Description

void **rsort** (array array)

This function sorts an array in reverse order (highest to lowest).

Beispiel 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: fruits[0] = orange fruits[1] = lemon fruits[2] = banana fruits[3] = apple The fruits have been sorted in reverse alphabetical order.

See also: [arsort\(\)](#), [asort\(\)](#), [ksort\(\)](#), [sort\(\)](#), and [usort\(\)](#).

shuffle

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

shuffle -- Shuffle an array

Description

void **shuffle** (array array)

This function shuffles (randomizes the order of the elements in) an array.

Beispiel 1 shuffle() example

```
$numbers = range(1,20);
srand(time());
shuffle($numbers);
while (list(,$number) = each($numbers)) {
    echo "$number ";
}
```

See also [arsort\(\)](#), [asort\(\)](#), [ksort\(\)](#), [rsort\(\)](#), [sort\(\)](#) and [usort\(\)](#).

sizeof

(PHP3 , PHP4)

sizeof -- Get the number of elements in an array

Description

int **sizeof** (array array)

Returns the number of elements in the array.

See also: [count\(\)](#)

sort

(PHP3 , PHP4)

sort -- Sort an array

Description

void **sort** (array array)

This function sorts an array. Elements will be arranged from lowest to highest when this function has completed.

Beispiel 1 sort() example

```
$fruits = array("lemon", "orange", "banana", "apple");
sort($fruits);
reset($fruits);
while(list($key, $val)=each($fruits)) {
    echo "$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.

See also: [arsort\(\)](#), [asort\(\)](#), [ksort\(\)](#), [rsort\(\)](#), and [usort\(\)](#).

uasort

(PHP3 >= 3.0.4, PHP4)

uasort -- Sort an array with a user-defined comparison function and maintain index association

Description

```
void uasort (array array, function 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.

uksort

(PHP3 >= 3.0.4, PHP4)

uksort -- Sort an array by keys using a user-defined comparison function

Description

void **uksort** (array array, function 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.

Beispiel 1 uksort() example

```
function mycompare($a, $b) {  
    if ($a == $b) return 0;  
    return ($a > $b) ? -1 : 1;  
}  
$a = array(4 => "four", 3 => "three", 20 => "twenty", 10 => "ten");  
uksort($a, mycompare);  
while(list($key, $value) = each($a)) {  
    echo "$key: $value\n";  
}
```

This example would display: 20: twenty 10: ten 4: four 3: three

See also: [arsort\(\)](#), [asort\(\)](#), [uasort\(\)](#), [ksort\(\)](#), [rsort\(\)](#), and [sort\(\)](#).

usort

(PHP3 >= 3.0.3, PHP4)

usort -- Sort an array by values using a user-defined comparison function

Description

void **usort** (array array, function 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. If two members compare as equal, their order in the sorted array is undefined.

Beispiel 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

Anmerkung: Obviously in this trivial case the [rsort\(\)](#) function would be more appropriate.

Warnung

The underlying quicksort function in some C libraries (such as on Solaris systems) may cause PHP to crash if the comparison function does not return consistent values.

See also: [arsort\(\)](#), [asort\(\)](#), [ksort\(\)](#), [rsort\(\)](#) and [sort\(\)](#).

III. Aspell Funktionen

Die **aspell()** Funktionen erlauben es ein Wort auf korrekte Rechtschreibung zu prüfen und Alternativen anzubieten.

Sie benötigen die aspell-Bibliothek von: <http://aspell.sourceforge.net/>.

Inhaltsverzeichnis

[aspell_new](#) Läd ein neues Wörterbuch

[aspell_check](#) Überprüfe ein Wort

[aspell_check-raw](#) Überprüfe ein Wort genauso wie es übergeben wird

[aspell_suggest](#) Schlägt mögliche Schreibweisen vor

aspell_new

(PHP3 >= 3.0.7, PHP4)

aspell_new -- Läd ein neues Wörterbuch

Beschreibung

int **aspell_new** (string master, string personal)

Aspell_new() öffnet ein neues Wörterbuch und liefert eine Wörterbuchkennung zurück, die von anderen aspell-Funktionen gebraucht wird.

Beispiel 1 aspell_new

```
$aspell_link=aspell_new ("english");
```

aspell_check

(PHP3 >= 3.0.7, PHP4)

aspell_check -- Überprüfe ein Wort

Beschreibung

boolean **aspell_check** (int Wörterbuchkennung, string Wort)

Aspell_check() überprüft die Rechtschreibung eines Wortes und liefert true, wenn das Wort korrekt ist, ansonsten false.

Beispiel 1 aspell_check

```
$aspell_link=aspell_new ("english");
if (aspell_check ($aspell_link, "test")) {
    echo "This is a valid spelling";
} else {
    echo "Sorry, wrong spelling";
}
```

aspell_check-raw

(unknown)

aspell_check-raw -- Überprüfe ein Wort genauso wie es übergeben wird

Beschreibung

boolean **aspell_check_raw** (int Wörterbuchkennung, string Wort)

Aspell_check_raw() überprüft die Rechtschreibung eines Wortes ohne die Groß/Kleinschreibung zu variieren oder das Wort in irgendeiner Weise zu kürzen. Liefert true, wenn das Wort korrekt ist und sonst false.

Beispiel 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_suggest

(PHP3 >= 3.0.7, PHP4)

aspell_suggest -- Schlägt mögliche Schreibweisen vor

Beschreibung

array **aspell_suggest** (int Wörterbuchkennung, string Wort)

Aspell_suggest() liefert ein Array mit möglichen Schreibweisen eines übergebenen Wortes.

Beispiel 1 Aspell_suggest()

```
$aspell_link=aspell_new("english");

if (!aspell_check ($aspell_link, "test")) {
    $suggestions=aspell_suggest ($aspell_link, "test");

    for ($i=0; $i < count ($suggestions); $i++) {
        echo "Possible spelling: " . $suggestions[$i] . "<br>";
    }
}
```

IV. Mathematische Funktionen mit beliebiger Genauigkeit

Diese Funktionen sind nur verfügbar, wenn PHP mit `--enable-bcmath` konfiguriert wurde.

Inhaltsverzeichnis

[bcadd](#) Addiere zwei Zahlen beliebiger Genauigkeit.

[bccomp](#) Vergleich zweier Zahlen beliebiger Genauigkeit.

[bcdiv](#) Divide two arbitrary precision numbers.

[bcmmod](#) Modulo zweier Zahlen mit beliebiger Genauigkeit.

[bcmul](#) Multipliziert zwei Zahlen beliebiger Genauigkeit.

[bcpow](#) Potenziert zwei Zahlen beliebiger Genauigkeit.

[bcscale](#) Setzt die Genauigkeit aller bc math Funktionen.

[bcsqrt](#) Liefert die Quadratwurzel mit beliebiger Genauigkeit.

[bcsub](#) Subtrahiert zwei Zahlen mit beliebiger Genauigkeit.

bcadd

(PHP3 , PHP4)

bcadd -- Addiere zwei Zahlen beliebiger Genauigkeit.

Beschreibung

string **bcadd** (string linker Operant, string rechter Operant [, int Genauigkeit])

Addiert den *linken Operant* zum dem *rechten Operant* und liefert das Ergebnis als Zeichenkette. Der optionale Parameter *Genauigkeit* wird benutzt, um die Zahl der Nachkommastellen im Ergebnis zu bestimmen.

Siehe auch [bcsub\(\)](#).

bccomp

(PHP3 , PHP4)

bccomp -- Vergleich zweier Zahlen beliebiger Genauigkeit.

Beschreibung

int **bccomp** (string linker Operant, string rechter Operant [, int Genauigkeit])

Vergleicht den *linken Operant* mit dem *rechten Operant* und liefert das Ergebnis als integer. Der optionale Parameter *Genauigkeit* bestimmt die Anzahl der Nachkommastellen, die für den Vergleich ausgewertet werden. Der Rückgabewert ist 0, wenn die beiden Operanten gleich sind. Wenn der *linke Operant* größer als der *rechte Operant* ist, wird +1 zurück geliefert und wenn der *linke Operant* kleiner als der *rechte Operant* ist, wird -1 zurück geliefert.

bcdiv

(PHP3 , PHP4)

bcdiv -- Divide two arbitrary precision numbers.

Beschreibung

string **bcdiv** (string linker Operant, string rechter Operant [, int Genauigkeit])

Dividiert den *linken Operant* durch den *rechten Operant*. Der optionale Parameter *Genauigkeit* setzt die Anzahl der Nachkommastellen im Ergebnis.

Siehe auch [bcmul\(\)](#).

bcmod

(PHP3 , PHP4)

bcmod -- Modulo zweier Zahlen mit beliebiger Genauigkeit.

Beschreibung

string **bcmod** (string linker Operant, string Modulus)

Liefert den *linken Operant* modulo *Modulus*.

Siehe auch [bcddiv\(\)](#).

bcmul

(PHP3 , PHP4)

bcmul -- Multipliziert zwei Zahlen beliebiger Genauigkeit.

Beschreibung

string **bcmul** (string linker Operant, string rechter Operant [, int Genauigkeit])

Multipliziert den *linken Operant* mit dem *rechter Operant*. Der optionale Parameter *Genauigkeit* setzt die Anzahl der Nachkommastellen im Ergebnis.

Siehe auch [bcdiv\(\)](#).

bcpow

(PHP3 , PHP4)

bcpow -- Potenziert zwei Zahlen beliebiger Genauigkeit.

Beschreibung

string **bcpow** (string x, string y [, int Genauigkeit])

Potenziert *x* mit *y*. Der optionale Parameter *Genauigkeit* setzt die Anzahl der Nachkommastellen im Ergebnis.

Siehe auch [bcsqrt\(\)](#).

bcscale

(PHP3 , PHP4)

bcscale -- Setzt die Genauigkeit aller bc math Funktionen.

Beschreibung

string **bcscale** (int Genauigkeit)

Diese Funktionen setzt die Genauigkeit aller in der Folge aufgerufenen bc math Funktionen, die nicht explizit die Genauigkeit durch den optionalen Parameter setzen.

bcsqrt

(PHP3 , PHP4)

bcsqrt -- Liefert die Quadratwurzel mit beliebiger Genauigkeit.

Beschreibung

string **bcsqrt** (string Operant, int Genauigkeit)

Liefert die Quadratwurzel von *Operant*. Der optionale Parameter *Genauigkeit* setzt die Anzahl der Nachkommastellen im Ergebnis.

Siehe auch [bcpow\(\)](#).

bcsup

(PHP3 , PHP4)

bcsup -- Subtrahiert zwei Zahlen mit beliebiger Genauigkeit.

Beschreibung

string **bcsup** (string linker Operant, string rechter Operant [, int Genauigkeit])

Subtrahiert den *rechten Operanten* von dem *linken Operanten* und liefert das Ergebnis als Zeichenkette. Der optionale Parameter *Genauigkeit* setzt die Anzahl der Nachkommastellen im Ergebnis.

Siehe auch [bcadd\(\)](#).

V. Kalender-Funktionen

Dieses Modul bietet eine Reihe von Funktionen, die die Umwandlung von Daten zwischen verschiedenen Kalendern erleichtern. Die gemeinsame Basis für diese Umwandlung bilden das Julianische Datum J.D. (nicht zu verwechseln mit dem Julianischen Kalender). Das J.D. entspricht der Anzahl der Tage seit dem 1.1.4713 v.Chr. Jede Umrechnung zwischen zwei beliebigen Kalendern erfordert den Zwischenschritt über das J.D.

Weitere Informationen zum Thema liefert die Seite <http://genealogy.org/~scottlee/cal-overview.html>, aus der Teile der Erklärungen in den folgenden Funktionsbeschreibungen stammen, sowie die sehr ausführliche [Homepage zum Thema ZEIT](#).

Warnung

In PHP3 und älteren Versionen von PHP4 (bis einschließlich PHP4RC1) sind diese Funktionen nur verfügbar, wenn Sie die Kalender-Erweiterungen unter dl/calendar übersetzt haben. Weitere Informationen finden Sie in der Datei dl/README.

Inhaltsverzeichnis

[JDToGregorian](#) Konvertierung vom Julianischen Datum zum Gregorianischen Kalender

[GregorianToJD](#) Konvertierung vom Gregorianischen Kalender zum Julianischen Datum

[JDToJulian](#) Konvertierung vom Julianischen Datum zum Julianischen Kalender

[JulianToJD](#) Konvertierung vom Julianischen Kalender zum Julianischen Datum

[JDToJewish](#) Konvertierung vom Julianischen Datum zum Jüdischen Kalender

[JewishToJD](#) Konvertiert vom Jüdischen Kalender zum Julianischen Datum

[JDToFrench](#) Konvertiert ein Julianisches Datum zum Kalender der Französischen Revolution

[FrenchToJD](#) Konvertiert ein Datum der Französischen Revolution zu einem Julianischen Datum

[JDMonthName](#) Bestimmt den Monat aus dem Julianischen Datum

[JDDayOfWeek](#) Bestimmt den Wochentag aus einem Julianischen Datum

[easter_date](#) Zeitpunkt des Osterfestes (0 Uhr) als UNIX-Timestamp

[easter_days](#) Anzahl der Tage zwischen dem 21. März und Ostersonntag

[unixtojd](#) Konvertiert UNIX-Timestamp in Julianisches Datum

[jdtounix](#) Konvertiert Julianisches Datum in UNIX-Timestamp

JDToGregorian

(unknown)

JDToGregorian -- Konvertierung vom Julianischen Datum zum Gregorianischen Kalender

Beschreibung

string **jdtogregorian** (int julianday)

Diese Funktion konvertiert den in *julianday* angegebenen Tag im Julianischen Datum in einen String, der das Datum nach dem Gregorianische Kalender in der Form "MM/DD/YYYY" enthält.

[Zurück](#)

Kalender-Funktionen

[Anfang](#)

[Hoch](#)

[Vor](#)

GregorianToJD

GregorianToJD

(unknown)

GregorianToJD -- Konvertierung vom Gregorianischen Kalender zum Julianischen Datum

Beschreibung

int gregoriantojd (int month, int day, int year)

Diese Funktion konvertiert das in *month*, *day* und *year* übergebene Datum im Gregorianischen Kalender in einen Tag im Julianischen Datum.

Gültige Daten liegen im Bereich von 4714 v.Chr. bis 9999 n.Chr. Der Gregorianische Kalender wurde allerdings erst am 15. Oktober 1582 (bzw. am 5. Oktober nach dem bis dahin gültigen Julianischen Kalender) von Papst Gregor dem XIII. eingeführt, in einigen Ländern allerdings erst sehr viel später. So übernahmen die Briten den Gregorianischen Kalender 1752, die UDSSR 1918 und Griechenland erst im Jahre 1923. In den meisten Europäischen Ländern wurde vor dem Gregorianischen der Julianische Kalender benutzt.

Beispiel 1 Calendar functions

```
<?php
$jd = GregorianToJD (10,11,1970);
echo "$jd\n";
$gregorian = JDToGregorian ($jd);
echo "$gregorian\n";
?>
```

JDToJulian

(unknown)

JDToJulian -- Konvertierung vom Julianischen Datum zum Julianischen Kalender

Beschreibung

string **jdtojulian** (int julianday)

Diese Funktion konvertiert den in *julianday* angegebenen Tag im Julianischen Datum in einen String, der das entsprechende Datum im Julianischen Kalender in der Form "MM/DD/YYYY" enthält.

[Zurück](#)

GregorianToJD

[Anfang](#)[Hoch](#)[Vor](#)

JulianToJD

JulianToJD

(unknown)

JulianToJD -- Konvertierung vom Julianischen Kalender zum Julianischen Datum

Beschreibung

`int juliantojd (int month, int day, int year)`

Diese Funktion konvertiert das in *month*, *day* und *year* übergebene Datum im Gregorianischen Kalender in einen Tag im Julianischen Datum.

Gültige Daten liegen im Bereich von 4714 v.Chr. bis 9999 n.Chr. Der Julianische Kalender wurde allerdings erst 46 v.Chr. eingeführt, und die Details der Anwendung stabilisierten sich frühestens im Jahre 8 n.Chr., z.T. vielleicht sogar erst im vierten Jahrhundert nach Christus. Weiterhin betrachteten nicht alle Kulturen, die sich nach diesem Kalender richteten, den Januar als den ersten Monat des Jahres.

[Zurück](#)

JDToJulian

[Anfang](#)[Hoch](#)[Vor](#)

JDToJewish

JDToJewish

(unknown)

JDToJewish -- Konvertierung vom Julianischen Datum zum Jüdischen Kalender

Beschreibung

string **jdtojewish** (int julianday)

Diese Funktion konvertiert den in *julianday* angegebenen Tag im Julianischen Datum in einen String, der das entsprechende Datum im Jüdischen Kalender in der Form "MM/DD/YYYY" enthält.

[Zurück](#)[Anfang](#)[Vor](#)

JulianToJD

[Hoch](#)

JewishToJD

JewishToJD

(unknown)

JewishToJD -- Konvertiert vom Jüdischen Kalender zum Julianischen Datum

Beschreibung

`int jewishtojd (int month, int day, int year)`

Diese Funktion konvertiert das in *month*, *day* und *year* übergebene Datum im Jüdischen Kalender in einen Tag im Julianischen Datum.

Gültige Daten beginnen mit dem Jahr 1 (3761 v.Chr.). Der Jüdische Kalender ist bereits seit mehreren Jahrtausenden im Gebrauch, wobei ein neuer Monat jeweils mit der ersten Beobachtung des Neumonds beginnt.

JDToFrench

(unknown)

JDToFrench -- Konvertiert ein Julianisches Datum zum Kalender der Französischen Revolution

Beschreibung

string **jdtofrench** (int month, int day, int year)

Diese Funktion konvertiert den in *julianday* angegebenen Tag im Julianischen Datum in einen String, der das entsprechende Datum im Kalender der Französischen Revolution in der Form "MM/DD/YYYY" enthält.

[Zurück](#)[Anfang](#)[Vor](#)

JewishToJD

[Hoch](#)

FrenchToJD

FrenchToJD

(unknown)

FrenchToJD -- Konvertiert ein Datum der Französischen Revolution zu einem Julianischen Datum

Beschreibung

int **frenchtojd** (int month, int day, int year)

Diese Funktion konvertiert das in *month*, *day* und *year* übergebene Datum im Kalender der französischen Revolution in einen Tag im Julianischen Datum.

Diese Funktion verarbeitet nur Daten der Jahre 1 bis 14 (22. September 1792 bis 22 September 1806 im Gregorianischen Kalender), dieser Zeitraum geht aber immer noch über die Lebensdauer des Revolutionskalenders hinaus.

JDMonthName

(unknown)

JDMonthName -- Bestimmt den Monat aus dem Julianischen Datum

Beschreibung

string **jdmmonthname** (int julianday, int mode)

Diese Funktion bestimmt den Monatsnamen für den in *julianday* übergebenen Tag im Julianischen Datum. Die Ausgabe erfolgt als String in Abhängigkeit von *mode*.

Tabelle 1 Mögliche *mode* Werte

Mode	Bedeutung
0	Gregorianisch (Abk.)
1	Gregorianisch
2	Julianisch - (Abk.)
3	Julianisch
4	Jüdisch
5	Französisch revolutionär

JDDayOfWeek

(unknown)

JDDayOfWeek -- Bestimmt den Wochentag aus einem Julianischen Datum

Beschreibung

mixed **jddayofweek** (int julianday, int mode)

Diese Funktion bestimmt den Wochentag für den in *julianday* übergebenen Tag im Julianischen Datum. Die Ausgabe erfolgt in Abhängigkeit von *mode* als integer oder string.

Tabelle 1 Mögliche *mode* Werte

Mode	Bedeutung
0	Nummer des Tages als integer (0=Sonntag, 1=Montag, ...)
1	Tagesname als String (Englisch-Gregorianisch)
2	abgekürzter Tagesname als String (Englisch-Gregorianisch)

easter_date

(PHP3 >= 3.0.9, PHP4 >= 4.0RC2)

easter_date -- Zeitpunkt des Osterfestes (0 Uhr) als UNIX-Timestamp

Beschreibung

`int easter_date ([int year])`

Diese Funktion liefert den UNIX-Timestamp (in Sekunden seit 1970) zurück, der dem Beginn des Ostersonntag im durch *year* spezifizierten Jahr entspricht. Wird kein Jahr angegeben, so wird der Zeitpunkt für das laufende Jahr bestimmt.

Warnung

Liegt das angegebene Jahr vor 1970 oder nach 2037 (auf 32Bit-Systemen) so wird eine Warnung ausgegeben und das Ergebnis ist nicht gültig.

Beispiel 1 Easter_date() Beispiel

```
echo date ("M-d-Y", easter_date(1999));      /* " 4. April 1999" */
echo date ("M-d-Y", easter_date(2000));      /* "23. April 2000" */
echo date ("M-d-Y", easter_date(2001));      /* "15. April 2001" */
```

Das Datum des Osterfestes wurde im Jahre 325 auf dem Konzil von Nicaea auf den ersten Sonntag nach Frühlingsvollmond, also dem Vollmond während oder direkt nach der Tag- und Nachtgleiche am 21. März. Der hier genutzte Algorithmus wurde um das Jahr 532 von Dionysius Exiguus eingeführt. Bis zum Jahr 1752 wird nach dem Julianischen Kalender ein einfacher 19jähriger Zyklus für die Bestimmung der Mondphasen benutzt. Im September 1752 übernahmen dann auch Britannien und seine Kolonien den von Clavius und Lilius vorgeschlagenen und im Oktober 1582 von Papst Gregor dem XIII. eingeführten Gregorianischen Kalender. Hierdurch wurden zwei Korrekturfaktoren eingeführt, um den Zyklus genauer abzubilden.

(Der Code basiert auf einem C-Programm von Simon Kershaw, <webmaster@ely.anglican.org>)

Siehe auch [easter_days\(\)](#) für die Bestimmung von Osterfesten vor 1970 oder nach 2037.

easter_days

(PHP3 >= 3.0.9, PHP4 >= 4.0RC2)

easter_days -- Anzahl der Tage zwischen dem 21. März und Ostersonntag

Beschreibung

`int easter_days (int year)`

Diese Funktion ermittelt den Abstand in Tagen zwischen der Frühlings-Tag-und-Nachtgliche und dem Ostersonntag im durch *year* angegebenen Jahr. Wird kein Jahr angegeben, so wird der Wert für das laufende Jahr bestimmt.

Diese Funktion kann im Gegensatz zu [easter_date\(\)](#) das Osterfest auch in Jahren vor 1970 und nach 2037 bestimmen.

Beispiel 1 Easter_date() Beispiel

```
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   */
```

Siehe auch [easter_date\(\)](#).

unixtojd

(PHP4 >= 4.0RC2)

unixtojd -- Konvertiert UNIX-Timestamp in Julianisches Datum

Beschreibung

int **unixtojd** ([int timestamp])

Diese Funktion konvertiert den in *timestamp* übergebenen Wert (in Sekunden seit dem 1.1.1970) in das entsprechende Julianische Datum. Wird kein *timestamp* übergeben, so wird das Julianische Datum des aktuellen Tages zurückgegeben.

Siehe auch [jdtounix\(\)](#).

Anmerkung: Diese Funktion ist nur in PHP-Versionen nach PHP4RC1 verfügbar.

jdtounix

(PHP4 >= 4.0RC2)

jdtounix -- Konvertiert Julianisches Datum in UNIX-Timestamp

Beschreibung

`int jdtounix (int jday)`

Diese Funktion liefert einen UNIX Timestamp (in Sekunden seit dem 1.1.1970) zurück, der dem in *jday* übergebenen Julianischen Datum entspricht. Liegt *jday* außerhalb des darstellbaren Zeitbereichs (Gregorianische Jahre von 1970 bis 2037 bzw. 2440588 <= *jday* <= 2465342), so wird false zurückgegeben.

Siehe auch [unixtojd\(\)](#).

Anmerkung: Diese Funktion ist nur in PHP-Versionen nach PHP4RC1 verfügbar.

VI. COM support functions for Windows

These functions are only available on the Windows version of PHP. These functions have been added in PHP4.

Inhaltsverzeichnis

[com_load](#) ???[com_invoke](#) ???[com_propget](#) ???[com_get](#) ???[com_propput](#) ???[com_propset](#) ???[com_set](#) ???

com_load

(PHP3 >= 3.0.3, PHP4)

com_load -- ???

Description

string **com_load** (string module name [, string server name])

[Zurück](#)[Anfang](#)[Vor](#)

COM support functions for
Windows

[Hoch](#)

com_invoke

com_invoke

(PHP3 >= 3.0.3, PHP4)

com_invoke -- ???

Description

mixed **com_invoke** (resource object, string function_name [, mixed function parameters, ...])

com_propget

(PHP3 >= 3.0.3, PHP4)

com_propget -- ???

Description

mixed **com_propget** (resource object, string property)

[Zurück](#)[Anfang](#)[Vor](#)

com_invoke

[Hoch](#)

com_get

com_get

(PHP3 >= 3.0.3, PHP4)

com_get -- ???

Description

mixed **com_get** (resource object, string property)

[Zurück](#)[Anfang](#)[Vor](#)

com_propget

[Hoch](#)

com_propput

com_propput

(PHP3 >= 3.0.3, PHP4)

com_propput -- ???

Description

void **com_propput** (resource object, string property, mixed value)

[Zurück](#)[Anfang](#)[Vor](#)

com_get

[Hoch](#)

com_propset

com_propset

(PHP3 >= 3.0.3, PHP4)

com_propset -- ???

Description

void **com_propset** (resource object, string property, mixed value)

This function is an alias for [com_propput\(\)](#).

com_set

(PHP3 >= 3.0.3, PHP4)

com_set -- ???

Description

void **com_set** (resource object, string property, mixed value)

This function is an alias for **com_set()**.

VII. Klassen- und Objekt Funktionen

Introduction

About

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 this 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).

An example of use

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.

Beispiel 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.

Beispiel 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

```

Inhaltsverzeichnis

[get_class](#) Returns the name of the class of an object

[get_parent_class](#) Returns the name of the parent class of an object

[get_class_methods](#) Liefert die Namen aller Methoden einer Klasse

[get_class_vars](#) Liefert die Standard-Elemente einer Klasse

[get_object_vars](#) Liefert die Elemente eines Objekts als assoziatives Array

[is_subclass_of](#) Determines if an object belongs to a subclass of the specified class

[class_exists](#) Checks if the class has been defined

[method_exists](#) Prüft, ob Methode in einer Klasse definiert ist

[get_declared_classes](#) Returns an array with the name of the defined classes

[Zurück](#)

com_set

[Anfang](#)

[Hoch](#)

[Vor](#)

get_class

get_class

(PHP4 >= 4.0b2)

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.

See also [get_parent_class\(\)](#), [is_subclass_of\(\)](#)

get_parent_class

(PHP4 >= 4.0b2)

`get_parent_class` -- Returns the name of the parent class of an object

Description

string **get_parent_class** (object *obj*)

This function returns the name of the parent class to the class of which the object *obj* is an instance.

See also [get_class\(\)](#), [is_subclass_of\(\)](#)

get_class_methods

(PHP4 >= 4.0RC1)

`get_class_methods` -- Liefert die Namen aller Methoden einer Klasse

Beschreibung

array **get_class_methods** (string *class_name*)

Diese Funktion liefert ein String-Array zurück, das mit den Namen aller in der Klasse *class_name* definierten Methoden gefüllt ist.

Siehe auch [get_class_vars\(\)](#), [get_object_vars\(\)](#)

get_class_vars

(PHP4 >= 4.0RC1)

get_class_vars -- Liefert die Standard-Elemente einer Klasse

Beschreibung

array **get_class_vars** (string class_name)

Diese Funktion liefert ein String-Array zurück, das mit den Standard-Elementen der Klasse *class_name* gefüllt ist. Standard-Elemente sind all die Elemente, die bereits in der Klassendefinition mit 'var' definiert wurden.

Siehe auch [get_class_methods\(\)](#), [get_object_vars\(\)](#)

get_object_vars

(PHP4 >= 4.0RC1)

get_object_vars -- Liefert die Elemente eines Objekts als assoziatives Array

Beschreibung

array **get_object_vars** (object *obj*)

Diese Funktion liefert einen assoziatives Array zurück, das mit den Namen aller Elemente des in *obj* übergebenen Objekts gefüllt ist.

Siehe auch [get_class_methods\(\)](#), [get_class_vars\(\)](#)

is_subclass_of

(PHP4 >= 4.0b4)

is_subclass_of -- Determines if an object belongs to a subclass of the specified class

Description

bool **is_subclass_of** (object *obj*, string *superclass*)

This function returns true if the object *obj*, belongs to a class which is a subclass of *superclass*, false otherwise.

See also [get_class\(\)](#), [get_parent_class\(\)](#)

class_exists

(PHP4 >= 4.0b4)

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.

[Zurück](#)[Anfang](#)[Vor](#)

is_subclass_of

[Hoch](#)

method_exists

method_exists

(PHP4 >= 4.0b2)

method_exists -- Prüft, ob Methode in einer Klasse definiert ist

Beschreibung

`bool method_exists (object object, string method_name)`

Diese Funktion liefert true, wenn eine Methode mit Namen *method_name* für das in *object* übergebene Objekt definiert wurde, sonst false.

Siehe auch [function_exists\(\)](#)

get_declared_classes

(PHP4 >= 4.0RC2)

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.

Anmerkung: In PHP 4.0.1pl2, three extra classes are returned at the beginning of the array: `stdClass` (defined in `Zend/zend.c`), `Over loadedTestClass` (defined in `ext/standard/basic_functions.c`) and `Directory` (defined in `ext/standard/dir.c`).

VIII. ClibPDF functions

ClibPDF lets you create PDF documents with PHP. It is available at [FastIO](#) but it isn't free software. You should definitely read the licence before you start playing with ClibPDF. If you cannot fulfil the licence agreement consider using pdflib by Thomas Merz, which is also very powerful. ClibPDF functionality and API is similar to Thomas Merz's pdflib but, according to FastIO, ClibPDF is faster and creates smaller documents. This may have changed with the new version 2.0 of pdflib. A simple benchmark (the pdfclock.c example from pdflib 2.0 turned into a php script) actually shows no difference in speed at all. The file size is also similar if compression is turned off. So, try them both and see which one does the job for you.

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.

Anmerkung: The function [cpdf_set_font\(\)](#) has changed since PHP3 to support asian fonts. The encoding parameter is no longer an integer but a string.

One big advantage of ClibPDF over pdflib is the possibility 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 is a handy feature but can be simulated with [pdf_translate\(\)](#).

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 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.

Beispiel 1 Simple ClibPDF Example

```
<?php
$cpdf = cpdf_open(0);
cpdf_page_init($cpdf, 1, 0, 595, 842);
cpdf_add_outline($cpdf, 0, 0, 0, 1, "Page 1");
cpdf_set_font($cpdf, "Times-Roman", 30, "WinAnsiEncoding");
cpdf_set_text_rendering($cpdf, 1);
cpdf_text($cpdf, "Times Roman outlined", 50, 750);
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:

Beispiel 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);
?>

```

Inhaltsverzeichnis

[cpdf_global_set_document_limits](#) Sets document limits for any pdf document

[cpdf_set_creator](#) Sets the creator field in the pdf document

[cpdf_set_title](#) Sets the title field of the pdf document

[cpdf_set_subject](#) Sets the subject field of the pdf document

[cpdf_set_keywords](#) Sets the keywords field of the pdf document

[cpdf_open](#) Opens a new pdf document

[cpdf_close](#) Closes the pdf document

[cpdf_page_init](#) Starts new page

[cpdf_finalize_page](#) Ends page

[cpdf_finalize](#) Ends document

[cpdf_output_buffer](#) Outputs the pdf document in memory buffer

[cpdf_save_to_file](#) Writes the pdf document into a file

[cpdf_set_current_page](#) Sets current page

[cpdf_begin_text](#) Starts text section

[cpdf_end_text](#) Ends text section

[cpdf_show](#) Output text at current position

[cpdf_show_xy](#) Output text at position

[cpdf_text](#) Output text with parameters

[cpdf_set_font](#) Select the current font face and size

[cpdf_set_leading](#) Sets distance between text lines

[cpdf_set_text_rendering](#) Determines how text is rendered

[cpdf_set_horiz_scaling](#) Sets horizontal scaling of text

[cpdf_set_text_rise](#) Sets the text rise

[cpdf_set_text_matrix](#) Sets the text matrix

[cpdf_set_text_pos](#) Sets text position

[cpdf_set_char_spacing](#) Sets character spacing

[cpdf_set_word_spacing](#) Sets spacing between words

[cpdf_continue_text](#) Output text in next line

[cpdf_stringwidth](#) Returns width of text in current font

[cpdf_save](#) Saves current enviroment

[cpdf_restore](#) Restores formerly saved enviroment

[cpdf_translate](#) Sets origin of coordinate system

[cpdf_scale](#) Sets scaling

[cpdf_rotate](#) Sets rotation

[cpdf_setflat](#) Sets flatness

[cpdf_setlinejoin](#) Sets linejoin parameter

[cpdf_setlinecap](#) Sets linecap aparameter

[cpdf_setmiterlimit](#) Sets miter limit

[cpdf_setlinewidth](#) Sets line width

[cpdf_setdash](#) Sets dash pattern

[cpdf_newpath](#) Startet einen neuen Pfad

[cpdf_moveto](#) Sets current point

[cpdf_rmoveto](#) Sets current point

[cpdf_curveto](#) Draws a curve

[cpdf_lineto](#) Draws a line

[cpdf_rlineto](#) Draws a line

[cpdf_circle](#) Draw a circle

[cpdf_arc](#) Draws an arc

[cpdf_rect](#) Draw a rectangle
[cpdf_closepath](#) Close path
[cpdf_stroke](#) Draw line along path
[cpdf_closepath_stroke](#) Close path and draw line along path
[cpdf_fill](#) Fill current path
[cpdf_fill_stroke](#) Fill and stroke current path
[cpdf_closepath_fill_stroke](#) Close, fill and stroke current path
[cpdf_clip](#) Clips to current path
[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_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_add_outline](#) Adds bookmark for current page
[cpdf_set_page_animation](#) Sets duration between pages
[cpdf_import_jpeg](#) Opens a JPEG image
[cpdf_place_inline_image](#) Places an image on the page
[cpdf_add_annotation](#) Adds annotation

[Zurück](#)

get_declared_classes

[Anfang](#)

[Hoch](#)

[Vor](#)

cpdf_global_set_document_limits

cpdf_global_set_document_limits

(PHP4 >= 4.0b4)

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_set_creator

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

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_title

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

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_subject

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

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_keywords

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

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_open

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

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\(\)](#).

Anmerkung: The return value will be needed in futher 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_close

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

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_page_init

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_page_init -- Starts new page

Description

void **cpdf_page_init** (int pdf document, int page number, int orientation, double height, double width, double 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 koordinate 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_finalize_page

(PHP3 >= 3.0.10, PHP4 >= 4.0b4)

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

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

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_output_buffer

(PHP3 >= 3.0.9, PHP4 >= 4.0b4)

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_save_to_file

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

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_set_current_page

(PHP3 >= 3.0.9, PHP4 >= 4.0b4)

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_begin_text

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

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\(\)](#).

Beispiel 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_end_text

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

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\(\)](#).

Beispiel 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_show

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

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_show_xy

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_show_xy -- Output text at position

Description

void **cpdf_show_xy** (int pdf document, string text, double x-koor, double y-koor, int mode)

The **cpdf_show_xy()** function outputs the string *text* at position with coordinates (*x-koor*, *y-koor*). The last optional parameter determines the unit length. If is 0 or omitted the default unit as specified for the page is used. Otherwise the koodinates are measured in postscript points disregarding the current unit.

Anmerkung: The function **cpdf_show_xy()** is identical to [cpdf_text\(\)](#) without the optional parameters.

See also [cpdf_text\(\)](#).

cpdf_text

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_text -- Output text with parameters

Description

void **cpdf_text** (int pdf document, string text, double x-koor, double y-koor, int mode, double orientation, int alignmode)

The **cpdf_text()** function outputs the string *text* at position with coordinates (*x-koor*, *y-koor*). The optional parameter determines the unit length. If is 0 or omitted the default unit as specified for the page is used. Otherwise the koodinates 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 align. See the ClibPDF documentation for possible values.

See also [cpdf_show_xy\(\)](#).

cpdf_set_font

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_set_font -- Select the current font face and size

Description

void **cpdf_set_font** (int pdf document, string font name, double size, string encoding)

The **cpdf_set_font()** function sets the 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_leading

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_set_leading -- Sets distance between text lines

Description

void **cpdf_set_leading** (int pdf document, double 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_text_rendering

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

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_horiz_scaling

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_set_horiz_scaling -- Sets horizontal scaling of text

Description

void **cpdf_set_horiz_scaling** (int pdf document, double scale)

The **cpdf_set_horiz_scaling()** function sets the horizontal scaling to *scale* percent.

[Zurück](#)[Anfang](#)[Vor](#)[cpdf_set_text_rendering](#)[Hoch](#)[cpdf_set_text_rise](#)

cpdf_set_text_rise

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_set_text_rise -- Sets the text rise

Description

void **cpdf_set_text_rise** (int pdf document, double value)

The **cpdf_set_text_rise()** function sets the text rising to *value* units.

[Zurück](#)[Anfang](#)[Vor](#)[cpdf_set_horiz_scaling](#)[Hoch](#)[cpdf_set_text_matrix](#)

cpdf_set_text_matrix

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

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

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_set_text_pos -- Sets text position

Description

void **cpdf_set_text_pos** (int pdf document, double x-koor, double y-koor, int mode)

The **cpdf_set_text_pos()** function sets the position of text for the next [cpdf_show\(\)](#) function call.

The last optional parameter *mode* determines the unit length. If is 0 or omitted the default unit as specified for the page is used. Otherwise the koodinates are measured in postscript points disregarding the current unit.

See also [cpdf_show\(\)](#), [cpdf_text\(\)](#).

cpdf_set_char_spacing

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_set_char_spacing -- Sets character spacing

Description

void **cpdf_set_char_spacing** (int pdf document, double space)

The **cpdf_set_char_spacing()** function sets the spacing between characters.

See also [cpdf_set_word_spacing\(\)](#), [cpdf_set_leading\(\)](#).

cpdf_set_word_spacing

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_set_word_spacing -- Sets spacing between words

Description

void **cpdf_set_word_spacing** (int pdf document, double space)

The **cpdf_set_word_spacing()** function sets the spacing between words.

See also [cpdf_set_char_spacing\(\)](#), [cpdf_set_leading\(\)](#).

cpdf_continue_text

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

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_stringwidth

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_stringwidth -- Returns width of text in current font

Description

double **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_save

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_save -- Saves current enviroment

Description

void **cpdf_save** (int pdf document)

The **cpdf_save()** function saves the current enviroment. 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_restore

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_restore -- Restores formerly saved enviroment

Description

void **cpdf_restore** (int pdf document)

The **cpdf_restore()** function restores the enviroment 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.

Beispiel 1 Save/Restore

```
<?php cpdf_save($pdf);  
// do all kinds of rotations, transformations, ...  
cpdf_restore($pdf) ?>
```

See also [cpdf_save\(\)](#).

cpdf_translate

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_translate -- Sets origin of coordinate system

Description

void **cpdf_translate** (int pdf document, double x-koor, double y-koor, int mode)

The **cpdf_translate()** function set the origin of coordinate system to the point (*x-koor*, *y-koor*).

The last optional parameter determines the unit length. If is 0 or omitted the default unit as specified for the page is used. Otherwise the koodinates are measured in postscript points disregarding the current unit.

cpdf_scale

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_scale -- Sets scaling

Description

void **cpdf_scale** (int pdf document, double x-scale, double y-scale)

The **cpdf_scale()** function set the scaling factor in both directions.

[Zurück](#)[Anfang](#)[Vor](#)

cpdf_translate

[Hoch](#)

cpdf_rotate

cpdf_rotate

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_rotate -- Sets rotation

Description

void **cpdf_rotate** (int pdf document, double angle)

The **cpdf_rotate()** function set the rotation in degress to *angle*.

cpdf_setflat

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_setflat -- Sets flatness

Description

void **cpdf_setflat** (int pdf document, double value)

The **cpdf_setflat()** function set the flatness to a value between 0 and 100.

[Zurück](#)[Anfang](#)[Vor](#)

cpdf_rotate

[Hoch](#)

cpdf_setlinejoin

cpdf_setlinejoin

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

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.

[Zurück](#)

cpdf_setflat

[Anfang](#)[Hoch](#)[Vor](#)

cpdf_setlinecap

cpdf_setlinecap

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_setlinecap -- Sets linecap aparameter

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_setmiterlimit

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_setmiterlimit -- Sets miter limit

Description

void **cpdf_setmiterlimit** (int pdf document, double value)

The **cpdf_setmiterlimit()** function set the miter limit to a value greater or equal than 1.

[Zurück](#)[Anfang](#)[Vor](#)

cpdf_setlinecap

[Hoch](#)

cpdf_setlinewidth

cpdf_setlinewidth

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_setlinewidth -- Sets line width

Description

void **cpdf_setlinewidth** (int pdf document, double width)

The **cpdf_setlinewidth()** function set the line width to *width*.

[Zurück](#)[Anfang](#)[Vor](#)

cpdf_setmiterlimit

[Hoch](#)

cpdf_setdash

cpdf_setdash

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_setdash -- Sets dash pattern

Description

void **cpdf_setdash** (int pdf document, double white, double 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_newpath

(PHP3 >= 3.0.9, PHP4 >= 4.0b4)

cpdf_newpath -- Startet einen neuen Pfad

Beschreibung

void **cpdf_newpath** (int pdf Dokument)

Die Funktion **cpdf_newpath()** startet einen neuen Pfad in dem angegebenen *pdf Dokument*.

[Zurück](#)[Anfang](#)[Vor](#)

cpdf_setdash

[Hoch](#)

cpdf_moveto

cpdf_moveto

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_moveto -- Sets current point

Description

void **cpdf_moveto** (int pdf document, double x-koor, double y-koor, int mode)

The **cpdf_moveto()** function set the current point to the coordinates *x-koor* and *y-koor*.

The last optional parameter determines the unit length. If is 0 or omitted the default unit as specified for the page is used. Otherwise the koodinates are measured in postscript points disregarding the current unit.

cpdf_rmoveto

(PHP3 >= 3.0.9, PHP4 >= 4.0b4)

cpdf_rmoveto -- Sets current point

Description

void **cpdf_rmoveto** (int pdf document, double x-koor, double y-koor, int mode)

The **cpdf_rmoveto()** function set the current point relative to the coordinates *x-koor* and *y-koor*.

The last optional parameter determines the unit length. If is 0 or omitted the default unit as specified for the page is used. Otherwise the koodinates are measured in postscript points disregarding the current unit.

See also [cpdf_moveto\(\)](#).

cpdf_curveto

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_curveto -- Draws a curve

Description

void **cpdf_curveto** (int pdf document, double x1, double y1, double x2, double y2, double x3, double 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 last optional parameter determines the unit length. If is 0 or omitted the default unit as specified for the page is used. Otherwise the koodinates are measured in postscript points disregarding the current unit.

See also [cpdf_moveto\(\)](#), [cpdf_rmoveto\(\)](#), [cpdf_rlineto\(\)](#), [cpdf_lineto\(\)](#).

cpdf_lineto

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_lineto -- Draws a line

Description

void **cpdf_lineto** (int pdf document, double x-koor, double y-koor, int mode)

The **cpdf_lineto()** function draws a line from the current point to the point with coordinates (*x-koor*, *y-koor*).

The last optional parameter determines the unit length. If is 0 or omitted the default unit as specified for the page is used. Otherwise the koodinates are measured in postscript points disregarding the current unit.

See also [cpdf_moveto\(\)](#), [cpdf_rmoveto\(\)](#), [cpdf_curveto\(\)](#).

cpdf_rlineto

(PHP3 >= 3.0.9, PHP4 >= 4.0b4)

cpdf_rlineto -- Draws a line

Description

void **cpdf_rlineto** (int pdf document, double x-koor, double y-koor, int mode)

The **cpdf_rlineto()** function draws a line from the current point to the relative point with coordinates (*x-koor*, *y-koor*).

The last optional parameter determines the unit length. If is 0 or omitted the default unit as specified for the page is used. Otherwise the koodinates are measured in postscript points disregarding the current unit.

See also [cpdf_moveto\(\)](#), [cpdf_rmoveto\(\)](#), [cpdf_curveto\(\)](#).

cpdf_circle

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_circle -- Draw a circle

Description

void **cpdf_circle** (int pdf document, double x-koor, double y-koor, double radius, int mode)

The **cpdf_circle()** function draws a circle with center at point (*x-koor*, *y-koor*) and radius *radius*.

The last optional parameter determines the unit length. If is 0 or omitted the default unit as specified for the page is used. Otherwise the koodinates are measured in postscript points disregarding the current unit.

See also [cpdf_arc\(\)](#).

cpdf_arc

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_arc -- Draws an arc

Description

void **cpdf_arc** (int pdf document, double x-koor, double y-koor, double radius, double start, double end, int mode)

The **cpdf_arc()** function draws an arc with center at point (*x-koor*, *y-koor*) and radius *radius*, starting at angle *start* and ending at angle *end*.

The last optional parameter determines the unit length. If is 0 or omitted the default unit as specified for the page is used. Otherwise the koodinates are measured in postscript points disregarding the current unit.

See also [cpdf_circle\(\)](#).

cpdf_rect

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_rect -- Draw a rectangle

Description

void **cpdf_rect** (int pdf document, double x-koor, double y-koor, double width, double height, int mode)

The **cpdf_rect()** function draws a rectangle with its lower left corner at point (*x-koor*, *y-koor*). This width is set to *width*. This height is set to *height*.

The last optional parameter determines the unit length. If is 0 or omitted the default unit as specified for the page is used. Otherwise the koodinates are measured in postscript points disregarding the current unit.

cpdf_closepath

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_closepath -- Close path

Description

```
void cpdf_closepath (int pdf document)
```

The **cpdf_closepath()** function closes the current path.

[Zurück](#)[Anfang](#)[Vor](#)

cpdf_rect

[Hoch](#)

cpdf_stroke

cpdf_stroke

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

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_closepath_stroke

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

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\(\)](#). Than clears the path.

See also [cpdf_closepath\(\)](#), [cpdf_stroke\(\)](#).

cpdf_fill

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

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_fill_stroke

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

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_closepath_fill_stroke

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

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_clip

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_clip -- Clips to current path

Description

```
void cpdf_clip (int pdf document)
```

The **cpdf_clip()** function clips all drawing to the current path.

[Zurück](#)[Anfang](#)[Vor](#)[cpdf_closepath_fill_stroke](#)[Hoch](#)[cpdf_setgray_fill](#)

cpdf_setgray_fill

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_setgray_fill -- Sets filling color to gray value

Description

void **cpdf_setgray_fill** (int pdf document, double value)

The **cpdf_setgray_fill()** function sets the current gray value to fill a path.

See also [cpdf_setrgbcolor_fill\(\)](#).

cpdf_setgray_stroke

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_setgray_stroke -- Sets drawing color to gray value

Description

void **cpdf_setgray_stroke** (int pdf document, double gray value)

The **cpdf_setgray_stroke()** function sets the current drawing color to the given gray value.

See also [cpdf_setrgbcolor_stroke\(\)](#).

cpdf_setgray

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_setgray -- Sets drawing and filling color to gray value

Description

void **cpdf_setgray** (int pdf document, double gray value)

The [cpdf_setgray_stroke\(\)](#) function sets the current drawing and filling color to the given gray value.

See also [cpdf_setrgbcolor_stroke\(\)](#), [cpdf_setrgbcolor_fill\(\)](#).

cpdf_setrgbcolor_fill

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_setrgbcolor_fill -- Sets filling color to rgb color value

Description

void **cpdf_setrgbcolor_fill** (int pdf document, double red value, double green value, double 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

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_setrgbcolor_stroke -- Sets drawing color to rgb color value

Description

void **cpdf_setrgbcolor_stroke** (int pdf document, double red value, double green value, double 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

(PHP3 >= 3.0.8, PHP4 >= 4.0b4)

cpdf_setrgbcolor -- Sets drawing and filling color to rgb color value

Description

void **cpdf_setrgbcolor** (int pdf document, double red value, double green value, double blue value)

The [cpdf_setrgbcolor_stroke\(\)](#) function sets the current drawing and filling color to the given rgb color value.

See also [cpdf_setrgbcolor_stroke\(\)](#), [cpdf_setrgbcolor_fill\(\)](#).

cpdf_add_outline

(PHP3 >= 3.0.9, PHP4 >= 4.0b4)

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.

Beispiel 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_set_page_animation

(PHP3 >= 3.0.9, PHP4 >= 4.0b4)

cpdf_set_page_animation -- Sets duration between pages

Description

void **cpdf_set_page_animation** (int pdf document, int transition, double 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_import_jpeg

(PHP3 >= 3.0.9, PHP4 >= 4.0b4)

cpdf_import_jpeg -- Opens a JPEG image

Description

`int cpdf_open_jpeg (int pdf document, string file name, double x-koor, double y-koor, double angle, double width, double height, double x-scale, double 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-koor*, *y-koor*). The image is rotated by *angle* degrees.

The last optional parameter determines the unit length. If is 0 or omitted the default unit as specified for the page is used. Otherwise the koodinates are measured in postscript points disregarding the current unit.

See also [cpdf_place_inline_image\(\)](#).

cpdf_place_inline_image

(PHP3 >= 3.0.9, PHP4 >= 4.0b4)

cpdf_place_inline_image -- Places an image on the page

Description

void **cpdf_place_inline_image** (int pdf document, int image, double x-koor, double y-koor, double angle, double width, double height, int mode)

The **cpdf_place_inline_image()** function places an image created with the php image functions on the page at postion (*x-koor*, *y-koor*). The image can be scaled at the same time.

The last optional parameter determines the unit length. If is 0 or omitted the default unit as specified for the page is used. Otherwise the koodinates are measured in postscript points disregarding the current unit.

See also [cpdf_import_jpeg\(\)](#),

cpdf_add_annotation

(PHP3 >= 3.0.12, PHP4 >= 4.0b4)

cpdf_add_annotation -- Adds annotation

Description

void **cpdf_add_annotation** (int pdf document, double llx, double lly, double urx, double 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 last optional parameter determines the unit length. If is 0 or omitted the default unit as specified for the page is used. Otherwise the koodinates are measured in postscript points disregarding the current unit.

IX. Cybercash payment functions

These functions are only available if the interpreter has been compiled with the - -with-cybercash=[DIR]. These functions have been added in PHP4.

Inhaltsverzeichnis

[cybercash_encr](#) ???

[cybercash_decr](#) ???

[cybercash_base64_encode](#) ???

[cybercash_base64_decode](#)

[Zurück](#)

[Anfang](#)

[Vor](#)

[cpdf_add_annotation](#)

[Hoch](#)

[cybercash_encr](#)

cybercash_encr

(PHP4 >= 4.0b4)

cybercash_encr -- ???

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).

cybercash_decr

(PHP4 >= 4.0b4)

cybercash_decr -- ???

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_base64_encode

(PHP4 >= 4.0b4)

cybercash_base64_encode -- ???

Description

string **cybercash_base64_encode** (string inbuff)

[Zurück](#)[Anfang](#)[Vor](#)

cybercash_decr

[Hoch](#)

cybercash_base64_decode

cybercash_base64_decode

(PHP4 >= 4.0b4)

cybercash_base64_decode --

Description

string **cybercash_base64_decode** (string inbuff)

[Zurück](#)[Anfang](#)[Vor](#)[cybercash_base64_encode](#)[Hoch](#)

Database (dbm-style) abstraction
layer functions

X. Database (dbm-style) abstraction layer functions

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 subset of features modern databases such as Sleepycat Software's DB2 support. (This is not to be confused with IBM's DB2 software, which is supported through the [ODBC functions](#).)

The behaviour of various aspects depend 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.

The following handlers are supported:

- 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 is a newer type and more flexible than dbm. It still has most of the arbitrary limits of dbm (therefore it is deprecated).
- gdbm is the GNU database manager.
- 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."
- 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.

Beispiel 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. 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.

Beispiel 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);
```

```
?>
```

Inhaltsverzeichnis

dba_close	Close database
dba_delete	Delete entry specified by key
dba_exists	Check whether key exists
dba_fetch	Fetch data specified by key
dba_firstkey	Fetch first key
dba_insert	Insert entry
dba_nextkey	Fetch next key
dba_popen	Open database persistently
dba_open	Open database
dba_optimize	Optimize database
dba_replace	Replace or insert entry
dba_sync	Synchronize database

[Zurück](#)

[Anfang](#)

[Vor](#)

cybercash_base64_decode

[Hoch](#)

dba_close

dba_close

(PHP3 >= 3.0.8, PHP4 >= 4.0b2)

dba_close -- Close database

Description

void **dba_close** (int 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\(\)](#) [dba_popen\(\)](#)

dba_delete

(PHP3 >= 3.0.8, PHP4 >= 4.0b2)

dba_delete -- Delete entry specified by key

Description

string **dba_delete** (string key, int 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\(\)](#) [dba_replace\(\)](#)

dba_exists

(PHP3 >= 3.0.8, PHP4 >= 4.0b2)

dba_exists -- Check whether key exists

Description

bool **dba_exists** (string key, int 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\(\)](#) [dba_replace\(\)](#)

dba_fetch

(PHP3 >= 3.0.8, PHP4 >= 4.0b2)

dba_fetch -- Fetch data specified by key

Description

string **dba_fetch** (string key, int handle)

dba_fetch() fetches the data specified by *key* from the database specified with *handle*.

key is the key the data is specified by.

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\(\)](#) [dba_replace\(\)](#)

dba_firstkey

(PHP3 >= 3.0.8, PHP4 >= 4.0b2)

dba_firstkey -- Fetch first key

Description

string **dba_firstkey** (int 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\(\)](#)

dba_insert

(PHP3 >= 3.0.8, PHP4 >= 4.0b2)

dba_insert -- Insert entry

Description

bool **dba_insert** (string key, string value, int 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 or fails, respectively.

See also: [dba_exists\(\)](#) [dba_delete\(\)](#) [dba_fetch\(\)](#) [dba_replace\(\)](#)

dba_nextkey

(PHP3 >= 3.0.8, PHP4 >= 4.0b2)

dba_nextkey -- Fetch next key

Description

string **dba_nextkey** (int handle)

dba_nextkey() returns the next key of the database specified by *handle* and increments 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\(\)](#)

dba_popen

(PHP3 >= 3.0.8, PHP4 >= 4.0b2)

dba_popen -- Open database persistently

Description

int **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 handler id or false, in the case the open is successful or fails, respectively.

See also: [dba_open\(\)](#) [dba_close\(\)](#)

dba_open

(PHP3 >= 3.0.8, PHP4 >= 4.0b2)

dba_open -- Open database

Description

int **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.

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 handler id or false, in the case the open is successful or fails, respectively.

See also: [dba_popen\(\)](#) [dba_close\(\)](#)

dba_optimize

(PHP3 >= 3.0.8, PHP4 >= 4.0b2)

dba_optimize -- Optimize database

Description

bool **dba_optimize** (int 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_replace

(PHP3 >= 3.0.8, PHP4 >= 4.0b2)

dba_replace -- Replace or insert entry

Description

bool **dba_replace** (string key, string value, int 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 or fails, respectively.

See also: [dba_exists\(\)](#) [dba_delete\(\)](#) [dba_fetch\(\)](#) [dba_insert\(\)](#)

dba_sync

(PHP3 >= 3.0.8, PHP4 >= 4.0b2)

dba_sync -- Synchronize database

Description

bool **dba_sync** (int 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\(\)](#)

XI. Datums- und Zeit-Funktionen

Inhaltsverzeichnis

- [checkdate](#) Prüft ein Datum auf Gültigkeit
- [date](#) Formatiert ein(e) angegebene(s) Zeit/Datum
- [getdate](#) Gibt Datums- und Zeitinformationen zurück
- [gettimeofday](#) Gibt die aktuelle Zeit zurück
- [gmdate](#) Formatiert eine GMT Zeitangabe
- [gmmktime](#) Gibt den UNIX timestamp (Zeitstempel) als GMT zurück
- [gmstrftime](#) Wandelt eine lokale Zeitangabe in GMT-Format um
- [localtime](#) Ermittelt die lokalen Zeitwerte
- [microtime](#) Gibt den aktuellen UNIX Timestamp/Zeitstempel in Mikrosekunden zurück
- [mktime](#) Gibt den UNIX Timestamp/Zeitstempel für ein Datum zurück
- [strftime](#) Formatiert eine Zeit-/Datumsangabe nach den lokalen Einstellungen
- [time](#) Gibt den gegenwärtigen UNIX-Timestamp/Zeitstempel zurück
- [Strtotime](#) Wandelt ein beliebiges Datum (englisches Format) in einen Unix-Zeitstempel (timestamp) um

checkdate

(PHP3 , PHP4)

checkdate -- Prüft ein Datum auf Gültigkeit

Beschreibung:

`int checkdate (int Monat, int Tag, int Jahr)`

Gibt true/wahr zurück, wenn das angegebene Datum gültig ist, sonst wird false/falsch zurück gegeben.
Prüft die Gültigkeit der angegebenen Datumswerte. Ein Datum wird als gültig angesehen, wenn

- das Jahr zwischen 0 und 32767 (inclusive) ist,
- der Monat zwischen 1 und 12 (inclusive) ist und
- der Tag innerhalb der erlaubten Werte des angegebenen Monats ist. Schaltjahre werden berücksichtigt.

date

(PHP3 , PHP4)

date -- Formatiert ein(e) angegebene(s) Zeit/Datum

Beschreibung:

string **date** (string format [, int timestamp])

Gibt einen formatierten String anhand eines vorzugebenden Musters zurück. Dabei wird entweder der angegebene *timestamp* oder die gegenwärtige lokale Zeit berücksichtigt, wenn kein timestamp gegeben ist.

Die folgenden Platzhalter werden zur Formatierung benutzt:

- a - "am" oder "pm"
- A - "AM" oder "PM"
- d - Tag des Monats, 2-stellig mit führender Null: "01" bis "31"
- D - Tag der Woche als 3 Buchstaben, z.B. "Fri"
- F - Monat als ganzes Wort, z.B. "January"
- h - Stunde im 12-Stunden-Format: "01" bis "12"
- H - Stunde im 24-Stunden-Format: "00" bis "23"
- g - Stunde im 12-Stunden-Format ohne führende Null: "1" bis "12"
- G - Stunde im 24-Stunden-Format ohne führende Null: "0" bis "23"
- i - Minuten: "00" bis "59"
- j - Tag des Monats ohne führende Null: "1" bis "31"
- l - (kleines 'L') ausgeschriebener Tag der Woche, z.B. "Friday"
- L - boolscher Wert für Schaltjahre: "0" oder "1"
- m - Monat: "01" to "12"
- n - Monat ohne führende Null: "1" bis "12"
- M - Monat als 3 Buchstaben, z.B. "Jan"

- s - Sekunden: "00" bis "59"
- S - Anhang der englischen Aufzählung, 2 Buchstaben, z.B. "th" [bei 4 für fourth], "nd" [bei 2 für second]
- t - Anzahl der Tage des angegebenen Monats: "28" bis "31"
- T - Zeitzoneneinstellung des Rechners, z.B. "MDT"
- U - Sekunden seit Beginn der UNIX-Epoche (1.1.1970)
- w - Wochentag beginnend mit "0" (Sonntag) bis "6" (Samstag)
- Y - Jahr als vierstellige Zahl, z.B. "1999"
- y - Jahr als zweistellige Zahl, z.B. "99"
- z - Tag des Jahres: "0" bis "365"
- Z - Offset der Zeitzone gegen GTM ("-43200" bis "43200") in Minuten

Nicht definierte Zeichen werden unverändert zurück gegeben. Das "Z"-Format gibt "0" zurück beim Gebrauch von [gmdate\(\)](#).

Beispiel 1 Date() Beispiel

```
print (date("l dS of F Y h:i:s A"));
print ("July 1, 2000 is on a " . date ("l", mktime(0,0,0,7,1,2000)));
```

Es ist möglich, **date()** und [mktime\(\)](#) gleichzeitig zu verwenden, um Datumsangaben der Zukunft/Vergangenheit herauszufinden.

Beispiel 2 Date() und [mktime\(\)](#) Beispiel

```
$heute = mktime(0, 0, 0, date ("m") , date ("d")+1, date("Y"));
$letztermonat = mktime(0, 0, 0, date("m")-1,date("d"), date("Y"));
$naechstesjahr = mktime(0, 0, 0, date("m"), date("d"), date("Y")+1);
```

Um Datumsangaben in anderen Sprach-/Zeitzone-Formaten auszugeben, sollten sie die Funktionen [setlocale\(\)](#) und [strftime\(\)](#) nutzen.

Siehe auch [gmdate\(\)](#) und [mktime\(\)](#).

[Zurück](#)

checkdate

[Anfang](#)[Hoch](#)[Vor](#)

getdate

getdate

(PHP3 , PHP4)

getdate -- Gibt Datums- und Zeitinformationen zurück

Beschreibung:

array **getdate** (int timestamp)

Gibt ein assoziatives Array mit Datums- und Zeitangaben des angegebenen Timestamp zurück. Die Arrayelemente sind folgende:

- "seconds" - Sekunden
- "minutes" - Minuten
- "hours" - Stunden
- "mday" - Tag des Monats
- "wday" - numerischer Tag der Woche
- "mon" - Monat als Zahl
- "year" - Jahr als Zahl
- "yday" - Tag des Jahres als Zahlwert, z.B. "299"
- "weekday" - ausgeschriebener Wochentag, z.B. "Friday"
- "month" - ausgeschriebener Monatsname, z.B. "January"

gettimeofday

(PHP3 >= 3.0.7, PHP4 >= 4.0b4)

gettimeofday -- Gibt die aktuelle Zeit zurück

Beschreibung:

array **gettimeofday** (void)

Es handelt sich um eine Schnittstelle zum Systemaufruf von gettimeofday(2). Die Zeitangaben werden als assoziatives Array zurück gegeben. Folgende Arrayelemente sind vorhanden:

- "sec" - Sekunden
- "usec" - Microsekunden
- "minuteswest" - Minuten westlich von Greenwich
- "dsttime" - Art der Korrektur der Sommerzeit

gmdate

(PHP3 , PHP4)

gmdate -- Formatiert eine GMT Zeitangabe

Beschreibung:

string **gmdate** (string format, int timestamp)

Entspricht der [date\(\)](#) Funktion, ausser das als Zeitangabe immer Greenwich Mean Time (GMT) zurück gegeben wird. Steht ihr System in Deutschland (GMT + 01:00) wird im Beispiel unten (1. Zeile) "Jan 01 1998 00:00:00" ausgegeben, wogegen die 2. Zeile "Dec 31 1997 23:00:00" zurück gibt.

Beispiel 1 Gmdate() Beispiel

```
echo date ("M d Y H:i:s", mktime (0,0,0,1,1,1998))."\n";  
echo gmdate("M d Y H:i:s", mktime (0,0,0,1,1,1998))."\n";
```

Siehe auch [date\(\)](#), [mktime\(\)](#) und [gmmktime\(\)](#).

gmmktime

(PHP3 , PHP4)

gmmktime -- Gibt den UNIX timestamp (Zeitstempel) als GMT zurück

Beschreibung:

int **gmmktime** (int Stunde, int Minute, int Secunde, int Monat, int Tag, int Jahr [, int is_dst])

Identisch zu [mktime\(\)](#). Die angegebenen Parameter werden aber als GMT-Datum/Zeit zurück gegeben.

gmstrftime

(PHP3 >= 3.0.12, PHP4 >= 4.0RC2)

gmstrftime -- Wandelt eine lokale Zeitangabe in GMT-Format um

Beschreibung:

string **gmstrftime** (string format, int timestamp)

Anmerkung: Erst ab Version 3.0.12

Bewirkt das gleiche wie [strftime\(\)](#) mit dem Unterschied, dass die GMT-Zeit zurück gegeben wird. In Deutschland (GMT + 01:00) wird im folgenden Beispiel bei strftime "Dec 31 1998 00:00:00", bei gmstrftime dagegen "Dec 31 1998 23:00:00" ausgegeben.

Beispiel 1 Gmstrftime() Beispiel

```
setlocale ('LC_TIME', 'ge');  
echo strftime ("%b %d %Y %H:%M:%S", mktime(0,0,0,12,31,98))."\n";  
echo gmstrftime ("%b %d %Y %H:%M:%S", mktime(0,0,0,12,31,98))."\n";
```

Siehe auch [strftime\(\)](#).

localtime

(PHP4 >= 4.0RC2)

localtime -- Ermittelt die lokalen Zeitwerte

Beschreibung:

array **localtime** ([int timestamp [, bool is_associative]])

Die **localtime()**-Funktion gibt ein Array zurück, welches hinsichtlich seiner Struktur identisch ist mit dem des Funktionsaufrufs in C. Das erste (optionale) Argument von **localtime()** ist der UNIX-Zeitstempel (Timestamp). Fehlt dieser, wird die aktuelle Zeit genommen. Als zweites wird der Parameter *is_associative* übergeben. Ist dieser 0 oder nicht gesetzt, wird das Werte-Array als reguläres, numerisch indiziertes Array zurück gegeben. Ist er dagegen 1, liefert **localtime()** ein assoziatives Array zurück, welches alle Elemente der Struktur des localtime-Aufrufs in C enthält. Die Bezeichner der verschiedenen Schlüssel dieses assoziativen Arrays lauten:

- "tm_sec" - Sekunde
- "tm_min" - Minute
- "tm_hour" - Stunde
- "tm_mday" - Tag des Monats
- "tm_mon" - Monat des Jahres
- "tm_year" - Jahr (nicht Y2K sicher)
- "tm_wday" - Tag der Woche
- "tm_yday" - Tag des Jahres
- "tm_isdst" - für das Datum ist die Sommerzeit zu berücksichtigen

microtime

(PHP3 , PHP4)

microtime -- Gibt den aktuellen UNIX Timestamp/Zeitstempel in Mikrosekunden zurück

Beschreibung:

```
string microtime(void);
```

Gibt die Zeichenkette "msec sec" zurück, wobei sec die aktuelle Zeit ist, ausgedrückt als Anzahl der Sekunden seit Beginn der UNIX-Epoche (0:00:00 January 1, 1970 GMT). Msec stellt den Mikrosekunden-Teil dar. Diese Funktion ist nur auf Systemen möglich, die den Systemaufruf gettimeofday() unterstützen.

Siehe auch [time\(\)](#).

mktime

(PHP3 , PHP4)

mktime -- Gibt den UNIX Timestamp/Zeitstempel für ein Datum zurück

Beschreibung:

`int mktime (int Stunde, int Minute, int Sekunde, int Monat, int Tag, int Jahr [, int is_dst])`

Warnung: Beachten sie die andersartige Anordnung der Argumente. Diese unterscheidet sich von der Reihenfolge des UNIX-Befehls `mktime()`. Das ist eine häufige Fehlerursache in Scripts.

Gibt den UNIX Timestamp/Zeitstempel an Hand der gegebenen Argumente zurück. Der Zeitstempel ist ein long-integer-Wert, der die Anzahl der Sekunden zwischen der UNIX Epoche (01.01.1970) und der angegebenen Zeit enthält.

Einzelne Argumente können von rechts nach links weggelassen werden. Sie werden dann ersetzt mit den Werten der lokalen Systemzeit bzw. des lokalen Systemdatums.

is_dst kann bei Sommerzeit auf 1 gesetzt werden. 0 bedeutet Winter-/Normalzeit, -1 (Vorgabewert) heisst, dass nicht bekannt ist, ob Sommer- oder Winterzeit herrscht.

Anmerkung: *is_dst* wurde in 3.0.10. eingeführt.

Mktime() ist hilfreich bei Datumsberechnungen und -überprüfungen, z.B. wird versucht, automatisch das korrekte Datum zu berechnen. So wird in den folgenden Beispielen immer die Zeichenkette "Jan-01-1998" ausgegeben.

Beispiel 1 mktime() Beispiel

```
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 darf auch zweistellig angegeben werden. Werte von 0 bis 69 stehen dann für die Jahre 2000 bis 2069, und 70 bis 99 für die Jahre von 1970-1999 (auf Systemen, die `time_t` als 32Bit-Integer abbilden, liegt der gültige Bereich für *year* zwischen 1902 und 2037).

Der letzte Tag eines gegebenen Monats kann als der Tag "0" des nächsten Monats ausgedrückt werden (nicht als der Tag "-1"). Jedes der folgenden Beispiele gibt die Zeichenkette "Letzter Tag im Feb. 2000

[mktime](#)

ist der 29." zurück.

Beispiel 2 Letzter Tag des nächsten Monats

```
$lastday = mktime (0,0,0,3,0,2000);  
echo strftime ("Letzter Tag im Feb. 2000 ist der %d.", $lastday);  
  
$lastday=mktime(0,0,0,4,-31,2000);  
echo strftime ("Letzter Tag im Feb. 2000 ist der %d.", $lastday);
```

Siehe auch [date\(\)](#) und [time\(\)](#).

[Zurück](#)

microtime

[Anfang](#)

[Hoch](#)

[Vor](#)

strftime

strftime

(PHP3 , PHP4)

strftime -- Formatiert eine Zeit-/Datumsangabe nach den lokalen Einstellungen

Beschreibung:

string **strftime** (string format [, int timestamp])

Gibt einen String zurück, der den angegebenen Formatierungs-Merkmalen entspricht. Dabei wird der gegebene Timestamp/Zeitstempel oder - falls dieser fehlt - die momentane lokale Zeit benutzt. Monats- und Wochentagsname wird entsprechend des per [setlocale\(\)](#) eingestellten Wertes gesetzt.

Nachfolgend die im Formatierungsstring zu gebrauchenden Platzhalter:

- %a - abgekürzter Name des Wochentages, abhängig von der gesetzten Umgebung
- %A - ausgeschriebener Name des Wochentages, abhängig von der gesetzten Umgebung
- %b - abgekürzter Name des Monats, abhängig von der gesetzten Umgebung
- %B - ausgeschriebener Name des Monats, abhängig von der gesetzten Umgebung
- %c - Wiedergabewerte für Datum und Zeit, abhängig von der gesetzten Umgebung
- %C - Jahrhundert (Jahr geteilt durch 100, gekürzt auf Integer, Wertebereich 00 bis 99)
- %d - Tag des Monats als Zahl (Bereich 00 bis 31)
- %D - so wie %m/%d/%y
- %e - Tag des Monats als Dezimal-Wert, einstelligen Werten wird ein Leerzeichen voran gestellt (Wertebereich ´ 0 ´ bis ´ 31 ´)
- %h - so wie %b
- %H - Stunde als Zahl im 24-Stunden-Format (Bereich 00 bis 23)
- %I - Stunde als Zahl im 12-Stunden-Format (Bereich 01 bis 12)
- %j - Tag des Jahres als Zahl (Bereich 001 bis 366)
- %m - Monat als Zahl (Bereich 1 bis 12)
- %M - Minute als Dezimal-Wert

- %n - neue Zeile
- %p - entweder `am' oder `pm' (abhängig von der gesetzten Umgebung) oder die entsprechenden Zeichenketten der gesetzten Umgebung
- %r - Zeit in "am"- oder "pm"-Formatierung
- %R - Zeit in der 24-Stunden-Formatierung
- %S - Sekunden als Dezimal-Wert
- %t - Tabulator
- %T - aktuelle Zeit, genau wie %H:%M:%S
- %u - Tag der Woche als Dezimal-Wert [1,7], dabei ist 1 der Montag
- %U - Nummer der Woche des aktuellen Jahres als Dezimal-Wert, beginnend mit dem ersten Sonntag als erstem Tag der ersten Woche
- %V - Kalenderwoche (nach ISO 8601:1988) des aktuellen Jahres. Als Dezimal-Zahl mit dem Wertebereich 01 bis 53, wobei die Woche 01 die erste Woche mit mindestens 4 Tagen im aktuellen Jahr ist. Die Woche beginnt montags (nicht sonntags)
- %w - Wochentag als Dezimal-Wert, Sonntag ist 0
- %W - Nummer der Woche des aktuellen Jahres, beginnend mit dem ersten Montag als erstem Tag der ersten Woche
- %x - bevorzugte Datumswiedergabe (ohne Zeit), abhängig von der gesetzten Umgebung
- %X - bevorzugte Zeitwiedergabe (ohne Datum), abhängig von der gesetzten Umgebung
- %y - Jahr als 2-stellige-Zahl (Bereich 00 bis 99)
- %Y - Jahr als 4-stellige-Zahl
- %Z - Zeitzone
- %% - ein %-Zeichen

Beispiel 1 Strftime() Beispiel

```
setlocale ("LC_TIME", "C");
print (strftime ("%A in Finnland ist "));
setlocale ("LC_TIME", "fi_FI");
print (strftime ("%A, in Frankreich "));
setlocale ("LC_TIME", "fr_CA");
print (strftime ("%A und in Deutschland "));
setlocale ("LC_TIME", "de_DE");
print (strftime ("%A.\n"));
```


Diese Beispiel funktioniert, wenn sie die jeweilige Umgebung in ihrem System eingestellt haben.

Siehe auch [setlocale\(\)](#) und [mktime\(\)](#).

[Zurück](#)

mktime

[Anfang](#)[Hoch](#)[Vor](#)

time

time

(PHP3 , PHP4)

time -- Gibt den gegenwärtigen UNIX-Timestamp/Zeitstempel zurück

Beschreibung

```
int time(void);
```

Gibt die seit Beginn der UNIX-Epoche (Januar 1 1970 00:00:00 GMT) bis jetzt vergangenen Sekunden zurück.

Siehe auch [date\(\)](#).

[Zurück](#)[Anfang](#)[Vor](#)

strftime

[Hoch](#)

Strtotime

Strtotime

(unknown)

Strtotime -- Wandelt ein beliebiges Datum (englisches Format) in einen Unix-Zeitstempel (timestamp) um

Beschreibung:

`int strtotime (string time [, int now])`

Diese Funktion erwartet einen String mit einem Datum im englischen Format und versucht, dieses in einen UNIX-Timestamp umzusetzen.

Beispiel 1 Strtotime()-Beispiel:

```
echo strtotime ("10 march 2000") . "\n";
```

XII. dBase Funktionen

Diese Funktionen erlauben ihnen den Zugriff auf Datensätze, die im dBase-Format (also in dBase-Datenbanken) (dbf) vorliegen.

Es gibt keine Unterstützung für Indizes oder Memo-Felder. Auch Sperrfunktionen für Datensätze / -banken sind nicht vorhanden. Zwei gleichzeitig ablaufende Webserver-Prozesse, die die selbe dBase-Datei bearbeiten, werden ihre Datenbank zerstören.

Anders als bei SQL-Datenbanken können in dBase-"Datenbanken" die Felddefinitionen nicht nachträglich geändert werden. Nachdem die dbf-Datei erzeugt wurde, sind die Definitionen festgelegt. Es werden keine Index-Funktionen unterstützt, die Abfragen beschleunigen oder etwa ihre Datenorganisation unterstützen. dBase-Dateien sind einfache sequentielle Dateien mit fester Datensatzlänge. Neue Datensätze werden am Dateiende angehängt und gelöschte Datensätze werden beibehalten, bis sie [dbase_pack\(\)](#) aufrufen.

Wir empfehlen, keine Produktions-DBF-Datenbanken zu benutzen. Wählen sie zur Nutzung von Produktions-Daten mit PHP nur echte SQL-Server-Datenbanken; MySQL oder Postgres sind dabei eine gute Wahl. dBase-Unterstützung durch PHP dient hauptsächlich dem Im- und Export von Daten zu oder aus ihrer Web-Datenbank, wobei das DBF-Datei-Format vor allem wegen der Windows-Tabellenkalkulationen und -Organizer unterstützt wird. Im- und Export ist der einzige Grund für die dBase-Unterstützung durch PHP.

Inhaltsverzeichnis

[dbase_create](#) Erzeugt eine dBase-Datenbank

[dbase_open](#) Öffnet eine dBase-Datenbank

[dbase_close](#) Schließt eine dBase-Datenbank

[dbase_pack](#) Entfernt die als gelöscht markierten Datensätze aus der Datenbank

[dbase_add_record](#) Hängt einen neuen Datensatz an eine dBase-Datenbank an

[dbase_replace_record](#) Ersetzt einen Datensatz in einer dBase-Datenbank

[dbase_delete_record](#) Markiert einen Datensatz als gelöscht

[dbase_get_record](#) Liest einen Datensatz aus einer dBase-Datenbank

[dbase_get_record_with_names](#) Gibt einen Datensatz einer dBase-Datenbank als assoziatives Array zurück

[dbase_numfields](#) Stellt fest, wieviele Felder eine dBase-Datenbank hat

[dbase_numrecords](#) Ermittelt die Anzahl der Datensätze einer dBase-Datenbank

Strtotime

[Hoch](#)

dbase_create

dbase_create

(PHP3 , PHP4)

dbase_create -- Erzeugt eine dBase-Datenbank

Beschreibung:

int dbase_create (string filename, array fields)

Der *fields*-Parameter ist ein Array, das aus weiteren Arrays besteht. Jedes Array beschreibt das Format eines Feldes der Datenbank. Jedes Feld-Array besteht aus einem Namen, einem Zeichen zur Festlegung des Feldtyps, der Länge und dem Wert für die Anzahl der Nachkommastellen.

Die verfügbaren Feldtypen sind:

L

Boolean. Diese haben keine Länge oder Nachkommastellen.

M

Memo. (Beachten sie, dass diese von PHP nicht unterstützt werden.) Auch diese haben keine Länge oder Nachkommastellen.

D

Datum (gespeichert als JJJJMMTT). Ebenfalls keine Länge und Nachkommastellen.

N

Zahl. Diese haben sowohl eine Angabe der Länge als auch der Nachkommastellen.

C

Zeichenkette / String / Charakter. Hier ist eine Längenangabe erforderlich.

Konnte die Datenbank erfolgreich erzeugt werden, wird ein `dbase_identifizier` zurück gegeben, anderenfalls `false` / falsch.

Beispiel 1 Erzeugen einer dBase-Datenbank-Datei

```
// "Datenbank"-Name
$dbname = "/tmp/test.dbf";

// Datenbank-"Definitionen"
$def =
    array(
        array("Datum",      "D"),
        array("Name",       "C",  50),
        array("Alter",      "N",   3,  0),
        array("Email",      "C", 128),
        array("IstMitglied", "L")
    );

// Erzeugung
if (!dbase_create($dbname, $def))
    print "<strong>Error!</strong>";
```

[Zurück](#)

dBase Funktionen

[Anfang](#)[Hoch](#)[Vor](#)

dbase_open

dbase_open

(PHP3 , PHP4)

dbase_open -- Öffnet eine dBase-Datenbank

Beschreibung:

int **dbase_open** (string filename, int flags)

Die Flags entsprechen denen des open()-Systemaufrufs (normalerweise bedeutet 0 Nur-Lesen, 1 Nur-Schreiben und 2 Lesen und Schreiben).

Gibt einen dbase_identifier für die geöffnete Datenbank zurück oder false / falsch, wenn die Datenbank nicht geöffnet werden konnte.

dbase_close

(PHP3 , PHP4)

dbase_close -- Schließt eine dBase-Datenbank

Beschreibung:

bool **dbase_close** (int dbase_identifizier)

Schließt die durch *dbase_identifizier* angegebene Datenbank.

[Zurück](#)[Anfang](#)[Vor](#)

dbase_open

[Hoch](#)

dbase_pack

dbase_pack

(PHP3 , PHP4)

dbase_pack -- Entfernt die als gelöscht markierten Datensätze aus der Datenbank

Beschreibung:

bool **dbase_pack** (int dbase_identifizier)

Entfernt die in der angegebenen Datenbank als gelöscht markierten Datensätze endgültig (physisch).
Betroffen sind alle vorher per [dbase_delete_record\(\)](#) als gelöscht markierten Datensätze.

dbase_add_record

(PHP3 , PHP4)

dbase_add_record -- Hängt einen neuen Datensatz an eine dBase-Datenbank an

Beschreibung:

bool **dbase_add_record** (int dbase_identifizier, array record)

Hängt den per *record* definierten Datensatz an die Datenbank an. Wenn die Anzahl der Einträge im angegebenen Record nicht gleich der Anzahl der Felder in der Datenbank ist, wird dieser Befehl scheitern und false / falsch zurück geben.

dbase_replace_record

(PHP3 >= 3.0.11, PHP4)

dbase_replace_record -- Ersetzt einen Datensatz in einer dBase-Datenbank

Beschreibung:

bool **dbase_replace_record** (int dbase_identifizier, array record, int dbase_record_number)

Überschreibt die Daten des Datensatzes *record_number* mit den Daten aus *record* in der Datenbank. Falls die Anzahl der definierten Einträge ungleich der Anzahl der Datenfelder in dem angesprochenen Datensatz ist, wird die Ausführung dieses Befehls abgewiesen und false / falsch zurück gegeben.

dbase_record_number ist ein Integer-Wert, der von 1 bis zur Anzahl der Datensatz-Anzahl der Datenbank reicht (wie er bei Aufruf der Funktion [dbase_numrecords\(\)](#) zurück gegeben wird).

dbase_delete_record

(PHP3 , PHP4)

dbase_delete_record -- Markiert einen Datensatz als gelöscht

Beschreibung:

bool **dbase_delete_record** (int dbase_identifizier, int record)

Markiert den Datensatz *record* in der Datenbank als gelöscht. Um ihn wirklich (physisch) zu entfernen, müssen sie noch die Funktion [dbase_pack\(\)](#) aufrufen.

dbase_get_record

(PHP3 , PHP4)

dbase_get_record -- Liest einen Datensatz aus einer dBase-Datenbank

Beschreibung:

array **dbase_get_record** (int dbase_identifizier, int record)

Gibt die Daten von *record* in einem Array zurück. Das Array hat einen bei 0 beginnenden Index und enthält einen zusätzlichen Bereich der mit 'deleted' bezeichnet wird. Dieser ist auf 1 gesetzt, wenn der Datensatz als gelöscht markiert ist (siehe [dbase_delete_record\(\)](#)).

Jedes Feld wird in den entsprechenden PHP-Typ umgesetzt (Datumsangaben werden zu Zeichenketten / Strings).

dbase_get_record_with_names

(PHP3 >= 3.0.4, PHP4)

dbase_get_record_with_names -- Gibt einen Datensatz einer dBase-Datenbank als assoziatives Array zurück

Beschreibung:

array **dbase_get_record_with_names** (int dbase_identifizier, int record)

Gibt die Daten von *record* in einem assoziativen Array zurück. Das Array enthält auch einen assoziativen Bereich, der 'deleted' heisst und auf 1 steht, wenn der Datensatz als gelöscht markiert ist (siehe [dbase_delete_record\(\)](#)).

Jedes Feld wird in den entsprechenden PHP-Typ konvertiert (Datumsangaben werden zu Zeichenketten / Strings).

dbase_numfields

(PHP3 , PHP4)

dbase_numfields -- Stellt fest, wieviele Felder eine dBase-Datenbank hat

Beschreibung:

int dbase_numfields (int dbase_identifizier)

Gibt die Anzahl der Datenfelder (Spalten) in der angegebenen Datenbank zurück. Feld-Nummern sind Werte zwischen 0 und dbase_numfields(\$db)-1, wogegen die Datensatz-Werte zwischen 1 und dbase_numrecords(\$db) sein können.

Beispiel 1 Gebrauch von 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

(PHP3 , PHP4)

dbase_numrecords -- Ermittelt die Anzahl der Datensätze einer dBase-Datenbank

Beschreibung:

```
int dbase_numrecords (int dbase_identifizier)
```

Gibt die Anzahl der Datensätze (Zeilen) in der angegebenen Datenbank zurück. Datensatz-Nummern können Werte zwischen 1 und dbase_numrecords(\$db) sein, wogegen Feld-Werte zwischen 0 und dbase_numfields(\$db)-1 sein können.

[Zurück](#)[Anfang](#)[Vor](#)

dbase_numfields

[Hoch](#)

dbm functions

XIII. dbm functions

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).

Beispiel 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);
```

Inhaltsverzeichnis

- [dbmopen](#) opens a dbm database
- [dbmclose](#) closes a dbm database
- [dbmexists](#) tells if a value exists for a key in a dbm database
- [dbmfetch](#) fetches a value for a key from a dbm database
- [dbminsert](#) inserts a value for a key in a dbm database
- [dbmreplace](#) replaces the value for a key in a dbm database
- [dbmdelete](#) deletes the value for a key from a dbm database
- [dbmfirstkey](#) retrieves the first key from a dbm database
- [dbmnextkey](#) retrieves the next key from a dbm database
- [dblist](#) describes the dbm-compatible library being used

dbmopen

(PHP3 , PHP4)

dbmopen -- opens a dbm database

Description

int 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 from <ftp://prep.ai.mit.edu/pub/gnu>.

dbmclose

(PHP3 , PHP4)

dbmclose -- closes a dbm database

Description

```
bool dbmclose (int dbm_identifier)
```

Unlocks and closes the specified database.

[Zurück](#)

dbmopen

[Anfang](#)[Hoch](#)[Vor](#)

dbmexists

dbmexists

(PHP3 , PHP4)

dbmexists -- tells if a value exists for a key in a dbm database

Description

```
bool dbmexists (int dbm_identifizier, string key)
```

Returns true if there is a value associated with the *key*.

dbmfetch

(PHP3 , PHP4)

dbmfetch -- fetches a value for a key from a dbm database

Description

string **dbmfetch** (int dbm_identifier, string key)

Returns the value associated with *key*.

dbminsert

(PHP3 , PHP4)

dbminsert -- inserts a value for a key in a dbm database

Description

int **dbminsert** (int dbm_identifizier, 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\(\)](#).)

dbmreplace

(PHP3 , PHP4)

dbmreplace -- replaces the value for a key in a dbm database

Description

bool **dbmreplace** (int dbm_identifrier, 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.

[Zurück](#)[Anfang](#)[Vor](#)

dbminsert

[Hoch](#)

dbmdelete

dbmdelete

(PHP3 , PHP4)

dbmdelete -- deletes the value for a key from a dbm database

Description

```
bool dbmdelete (int dbm_identifizier, string key)
```

Deletes the value for *key* in the database.

Returns false if the key didn't exist in the database.

dbmfirstkey

(PHP3 , PHP4)

dbmfirstkey -- retrieves the first key from a dbm database

Description

string **dbmfirstkey** (int 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.

dbmnextkey

(PHP3 , PHP4)

dbmnextkey -- retrieves the next key from a dbm database

Description

string **dbmnextkey** (int 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:

Beispiel 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);
}
```

dblist

(PHP3 , PHP4)

dblist -- describes the dbm-compatible library being used

Description

string **dblist** (void)

[Zurück](#)[Anfang](#)[Vor](#)

dbmnextkey

[Hoch](#)

Verzeichnis-Funktionen

XIV. Verzeichnis-Funktionen

Inhaltsverzeichnis

[chdir](#) Wechseln des Verzeichnisses

[dir](#) Verzeichnis-Klasse

[closedir](#) Beenden eines Verzeichnis-Handles

[opendir](#) Öffnen eines Verzeichnis-Handles

[readdir](#) Liest den Eintrag eines Verzeichnis-Handles

[rewinddir](#) Zurücksetzen des Verzeichnis-Handles

chdir

(PHP3 , PHP4)

chdir -- Wechseln des Verzeichnisses

Beschreibung:

int **chdir** (string directory)

Wechsel des aktuellen PHP-Verzeichnisses in das durch *directory* angegebene Verzeichnis. Gibt FALSE zurück, wenn der Wechsel nicht möglich war, sonst TRUE.

[Zurück](#)

Verzeichnis-Funktionen

[Anfang](#)

[Hoch](#)

[Vor](#)

dir

dir

(PHP3 , PHP4)

dir -- Verzeichnis-Klasse

Beschreibung:

new **dir** (string directory)

Ein Pseudo-Objekt-Orientierter Mechanismus zum Lesen eines Verzeichnisses. Das per *directory* angegebene Verzeichnis wird geöffnet. Sobald das Verzeichnis geöffnet ist, stehen 2 Eigenschaften zur Verfügung. Die Eigenschaft Handle kann mit anderen Directory-Funktionen wie [readdir\(\)](#), [rewinddir\(\)](#) und [closedir\(\)](#) benutzt werden (Handles sind Verweise auf Einträge in einer internen Tabelle zur Verwaltung von Objekten). Die Eigenschaft Path enthält die Pfadangabe des geöffneten Verzeichnisses. Es sind 3 Methoden verfügbar: read, rewind und close.

Beispiel 1 Dir()-Beispiel:

```
$d = dir("/etc");  
echo "Handle: ".$d->handle."<br>\n";  
echo "Path: ".$d->path."<br>\n";  
while($entry=$d->read()) {  
    echo $entry."<br>\n";  
}  
$d->close();
```

closedir

(PHP3 , PHP4)

closedir -- Beenden eines Verzeichnis-Handles

Beschreibung:

void **closedir** (int dir_handle)

Beendet den mit *dir_handle* angegebenen Verzeichnis-Stream (Ein-/Ausgabe-Kanal für Verzeichnis-Objekte). Dieser muss vorher durch [opendir\(\)](#) geöffnet worden sein.

opendir

(PHP3 , PHP4)

opendir -- Öffnen eines Verzeichnis-Handles

Beschreibung:

int **opendir** (string path)

Gibt ein Verzeichnis-Handle zurück, das in späteren Aufrufen von [closedir\(\)](#), [readdir\(\)](#) und [rewinddir\(\)](#) gebraucht werden kann.

readdir

(PHP3 , PHP4)

readdir -- Liest den Eintrag eines Verzeichnis-Handles

Beschreibung:

string **readdir** (int dir_handle)

Ermittelt den nächsten Dateinamen im Verzeichnis. Die Dateinamen werden ungeordnet zurück gegeben.

Beispiel 1 Liste alle Dateien des aktuellen Verzeichnisses auf

```
<?php
$handle=opendir ('.');
echo "Verzeichnis-Handle: $handle\n";
echo "Dateien:\n";
while ($file = readdir ($handle)) {
    echo "$file\n";
}
closedir($handle);
?>
```

Beachten sie, dass **readdir()** auch die Einträge "." und ".." zurück gibt. Wollen sie das nicht, müssen sie diese ausschliessen:

Beispiel 2 Liste alle Dateien des aktuellen Verzeichnisses unter Ausschluss von "." und ".." auf

```
<?php
$handle=opendir('.');
while ($file = readdir ($handle)) {
    if ($file != "." && $file != "..") {
        echo "$file\n";
    }
}
closedir($handle);
?>
```

readdir

[Zurück](#)

opendir

[Anfang](#)

[Hoch](#)

[Vor](#)

rewinddir

rewinddir

(PHP3 , PHP4)

rewinddir -- Zurücksetzen des Verzeichnis-Handles

Beschreibung:

void **rewinddir** (int dir_handle)

Setzt den mit *dir_handle* angegebenen Verzeichnis-Stream auf den Anfang zurück (Reset).

XV. Dynamisch geladene Bibliothek

Inhaltsverzeichnis

[dl](#) Eine PHP-Bibliothek dynamisch bei Verwendung laden

[Zurück](#)

rewinddir

[Anfang](#)

[Hoch](#)

[Vor](#)

dl

dl

(PHP3 , PHP4)

dl -- Eine PHP-Bibliothek dynamisch bei Verwendung laden

Beschreibung

int **dl** (string library)

Lädt eine PHP-Bibliothek, welche in *library* angegeben wird. Weitere Informationen bei der Konfiguration der [extension_dir](#).

[Zurück](#)[Anfang](#)[Vor](#)

Dynamisch geladene Bibliothek

[Hoch](#)

DOM XML functions

XVI. DOM XML functions

These functions are only available if PHP was configured with `--with-dom=[DIR]`, using the GNOME xml library. These functions have been added in PHP4.

This module defines the following constants:

Tabelle 1 XML constants

Constant	Value	Description
XML_ELEMENT_NODE	1	
XML_ATTRIBUTE_NODE	2	
XML_TEXT_NODE	3	
XML_CDATA_SECTION_NODE	4	
XML_ENTITY_REF_NODE	5	
XML_ENTITY_NODE	6	
XML_PI_NODE	7	
XML_COMMENT_NODE	8	
XML_DOCUMENT_NODE	9	
XML_DOCUMENT_TYPE_NODE	10	
XML_DOCUMENT_FRAG_NODE	11	
XML_NOTATION_NODE	12	
XML_GLOBAL_NAMESPACE	1	
XML_LOCAL_NAMESPACE	2	

This module defines a number of classes. The DOM XML functions return a parsed tree of the XML document with each node being an object belonging to one of these classes.

Inhaltsverzeichnis

[xmldoc](#) Creates a DOM object of an XML document

[xmldocfile](#) Creates a DOM object from XML file

[xmltree](#) Creates a tree of php objects from XML document

xmldoc

(PHP4 >= 4.0b4)

xmldoc -- Creates a DOM object of an XML document

Description

object **xmldoc** (string *str*)

The function parses the XML document in *str* and returns an object of class "Dom document", having the properties "doc" (resource), "version" (string) and "type" (long).

xmldocfile

(PHP4 >= 4.0b4)

xmldocfile -- Creates a DOM object from XML file

Description

object **xmldocfile** (string filename)

The function parses the XML document in the file named *filename* and returns an object of class "Dom document", having the properties "doc" (resource), "version" (string).

xmldocfile

(PHP4 >= 4.0b4)

xmldocfile -- Creates a tree of php objects from XML document

Description

object **xmldocfile** (string *str*)

The function parses the XML document in *str* and returns a tree PHP objects as the parsed document.

[Zurück](#)[Anfang](#)[Vor](#)

xmldocfile

[Hoch](#)

filePro functions

XVII. filePro functions

These functions allow read-only access to data stored in filePro databases.

filePro is a registered trademark of Fiserv, Inc. You can find more information about filePro at <http://www.fileproplus.com/>.

Inhaltsverzeichnis

[filepro](#) read and verify the map file

[filepro_fieldname](#) gets the name of a field

[filepro_fieldtype](#) gets the type of a field

[filepro_fieldwidth](#) gets the width of a field

[filepro_retrieve](#) retrieves data from a filePro database

[filepro_fieldcount](#) find out how many fields are in a filePro database

[filepro_rowcount](#) find out how many rows are in a filePro database

filepro

(PHP3 , PHP4)

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.

filepro_fieldname

(PHP3 , PHP4)

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

(PHP3 , PHP4)

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

(PHP3 , PHP4)

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*.

[Zurück](#)[Anfang](#)[Vor](#)

filepro_fielddtype

[Hoch](#)

filepro_retrieve

filepro_retrieve

(PHP3 , PHP4)

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.

filepro_fieldcount

(PHP3 , PHP4)

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_rowcount

(PHP3 , PHP4)

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.

See also [filepro\(\)](#).

XVIII. Funktionen des Dateisystems

Inhaltsverzeichnis

- [basename](#) Gibt den Namen einer Datei aus einer vollständigen Pfadangabe zurück
- [chgrp](#) Wechselt die Gruppenzugehörigkeit einer Datei
- [chmod](#) Ändert die Zugriffsrechte einer Datei
- [chown](#) Ändert den Eigentümer einer Datei
- [clearstatcache](#) Löscht den Status Cache
- [copy](#) Kopiert eine Datei
- [delete](#) "Dummy Handbuch Eintrag"
- [dirname](#) Extrahiert den Verzeichnis-Namen aus einer vollständigen Pfadangabe
- [diskfree](#) Gibt den freien Speicherplatz in einem Verzeichnis zurück
- [fclose](#) Schließt einen offenen Dateizeiger
- [feof](#) Prüft, ob der Dateizeiger am Ende der Datei steht
- [fgetc](#) Liest das Zeichen, auf welches der Dateizeiger zeigt
- [fgetcsv](#) Liest eine Zeile von der Position des Dateizeigers und überprüft diese auf Komma-Separierte-Werte (CSV)
- [fgets](#) Liest eine Zeile von der Position des Dateizeigers
- [fgetss](#) Liest eine Zeile von der aktuellen Position des Dateizeigers und entfernt HTML und PHP-Tags.
- [file](#) Liest eine Datei komplett in einen Array
- [file_exists](#) Überprüft, ob eine Datei existiert
- [fileatime](#) Gibt Datum und Uhrzeit des letzten Zugriffs auf eine Datei zurück
- [filectime](#) Gibt Datum und Uhrzeit der letzten Änderung des Dateizeigers Inode zurück
- [filegroup](#) Gibt die Gruppenzugehörigkeit einer Datei zurück
- [fileinode](#) Gibt Inode-Nummer einer Datei aus
- [filemtime](#) Gibt Datum und Uhrzeit der letzten Dateiänderung aus
- [fileowner](#) Gibt den Dateieigentümer aus
- [fileperms](#) Gibt die Zugriffsrechte einer Datei aus
- [filesize](#) Gibt die Größe einer Datei aus
- [filetype](#) Gibt den Typ einer Datei zurück

- [flock](#) Portables Datei-Verriegelungs-Verfahren
 - [fopen](#) Öffnet eine Datei oder URL
 - [fpassthru](#) Gibt alle verbleibenden Daten eines Dateizeigers direkt aus.
 - [fputs](#) Schreibt Daten an die Position des Dateizeigers
 - [fread](#) Liest Binär-Dateien aus einer Datei
 - [fseek](#) Positioniert den Dateizeiger um einen Offset 'offset' vor oder zurück
 - [ftell](#) Ermittelt die aktuelle Position des Dateizeigers
 - [fwrite](#) Schreibt Binärdaten in eine Datei
 - [set_file_buffer](#) Setzt die Dateipufferung für einen gegebenen Dateizeiger
 - [is_dir](#) Ermittelt, ob der gegebene Dateiname ein Verzeichnis ist
 - [is_executable](#) Ermittelt, ob eine Datei ausführbar ist
 - [is_file](#) Ermittelt, ob der Dateiname eine reguläre Datei ist
 - [is_link](#) Ermittelt, ob der Dateiname ein symbolischer Link ist
 - [is_readable](#) Ermittelt ob eine Datei lesbar ist
 - [is_writable](#) Ermittelt, ob in eine Datei geschrieben werden kann
 - [link](#) Erzeugt einen absoluten Link
 - [linkinfo](#) Ermittelt Informationen über einen Link
 - [mkdir](#) Erstellt ein Verzeichnis
 - [pclose](#) Schließt einen Prozess-Dateizeiger
 - [popen](#) Öffnet einen Prozesszeiger
 - [readfile](#) Gibt eine Datei aus
 - [readlink](#) Gibt das Ziel eines symbolischen Links zurück
 - [rename](#) Benennt eine Datei um
 - [rewind](#) Setzt den Dateizeiger auf das erste Byte der Datei
 - [rmdir](#) Löscht ein Verzeichnis
 - [stat](#) Ermittelt diverse Informationen über eine Datei
 - [lstat](#) Ermittelt Informationen über einen symbolischen Link.
 - [realpath](#) Return canonicalized absolute pathname
 - [symlink](#) Erzeugt einen symbolischen Link
 - [tempnam](#) Erzeugt einen eindeutigen Dateinamen
 - [touch](#) Setzt das Datum der letzten Änderung einer Datei
 - [umask](#) Ändert die aktuelle umask (Zugriffsrechte)
 - [unlink](#) Löscht eine Datei
-

[Zurück](#)

filepro_rowcount

[Anfang](#)

[Hoch](#)

[Vor](#)

basename

basename

(PHP3 , PHP4)

basename -- Gibt den Namen einer Datei aus einer vollständigen Pfadangabe zurück

Beschreibung

string **basename** (string path)

Diese Funktion extrahiert aus einer vollständigen Pfadangabe den Namen der Datei und gibt diesen zurück.

Unter Windows wird sowohl der Slash (/) als auch der Backslash (\) als Trennzeichen bei Pfadangaben benutzt. Unter anderen Betriebssystemen hingegen nur der Slash (/).

Beispiel 1 Basename() Beispiel

```
$path = "/home/httpd/html/index.php3";  
$file = basename($path); // $file enthält den Wert "index.php3"
```

Siehe auch [dirname\(\)](#).

chgrp

(PHP3 , PHP4)

chgrp -- Wechselt die Gruppenzugehörigkeit einer Datei

Beschreibung

int **chgrp** (string filename, mixed group)

Weist einer Datei einer Benutzergruppe zu. Um diese Funktion ausführen zu können, müssen Sie Superuser sein. Andere Benutzer können die Gruppenzugehörigkeit einer Datei in jede Gruppe ändern, derer sie angehören.

Gibt bei Erfolg TRUE zurück, andernfalls FALSE.

Anmerkung: Unter Windows-Systemen bleibt diese Funktion ohne Effekt.

Siehe auch [chown\(\)](#) und [chmod\(\)](#).

chmod

(PHP3 , PHP4)

chmod -- Ändert die Zugriffsrechte einer Datei

Beschreibung

int **chmod** (string filename, int mode)

Diese Funktion ändert die Zugriffsrechte der Datei spezifiziert in *filename* in die Zugriffsrechte um, die in *mode* spezifiziert sind.

Bedenken Sie, das *mode* nicht automatisch für einen Oktalwert gehalten wird. Um sicher zu gehen, dass die Operation wie erwartet verläuft sollten Sie *mode* mit dem Prefix Null (0) versehen:

```
chmod ("/somedir/somefile", 755);    // dezimal; möglicherweise falsch  
chmod ("/somedir/somefile", 0755);   // oktal; richtiger Wert von mode
```

Gibt TRUE bei Erfolg zurück, andernfalls FALSE.

Siehe auch [chown\(\)](#) und [chgrp\(\)](#).

Anmerkung: Diese Funktion bleibt unter Windows-Systemen ohne Effekt.

chown

(PHP3 , PHP4)

chown -- Ändert den Eigentümer einer Datei

Beschreibung

int **chown** (string filename, mixed user)

Ändert den Eigentümer der Datei *filename* in Benutzer *user*. Nur der Superuser kann den Eigentümer einer Datei ändern.

Gibt bei Erfolg TRUE zurück, andernfalls FALSE.

Anmerkung: Unter Windows bleibt diese Funktion ohne Effekt.

Siehe auch **chown()** und [chmod\(\)](#).

clearstatcache

(PHP3 , PHP4)

clearstatcache -- Löscht den Status Cache

Beschreibung

```
void clearstatcache(void);
```

clearstatcache() löscht den Zwischenspeicher mit den Dateistatuswerten. Die Systemaufrufe stat oder lstat sind auf den meisten Systemen sehr "teure" Aufrufe. Daher wird das Ergebnis des letzten Aufrufs von einer der unten aufgeführten Funktionen für die nächste Benutzung eines ähnlichen Aufrufs mit dem gleichen Dateinamen gespeichert. Wenn Sie einen neuen Status-Check erzwingen wollen, z.B. wenn die Datei in der Zwischenzeit oftmals aufgerufen wurde, verändert oder verschwunden ist, können Sie diese Funktion dazu benutzen, das Ergebnis des letzten Aufrufes aus dem Speicher zu leeren.

Dieser Wert wird nur für die Zeit eines einzelnen Requests zwischengespeichert.

Folgende Funktionen betreffen den Dateistatus: [stat\(\)](#), [lstat\(\)](#), [file_exists\(\)](#), [is_writeable\(\)](#), [is_readable\(\)](#), [is_executable\(\)](#), [is_file\(\)](#), [is_dir\(\)](#), [is_link\(\)](#), [filectime\(\)](#), [fileatime\(\)](#), [filemtime\(\)](#), [fileinode\(\)](#), [filegroup\(\)](#), [fileowner\(\)](#), [filesize\(\)](#), [filetype\(\)](#), and [fileperms\(\)](#).

copy

(PHP3 , PHP4)

copy -- Kopiert eine Datei

Beschreibung

int **copy** (string source, string dest)

Die Funktion **copy()** kopiert eine Datei. Sie gibt TRUE zurück, wenn der Kopiervorgang erfolgreich war, andernfalls FALSE.

Beispiel 1 Copy() Beispiel

```
if (!copy ($file, $file.'.bak')) {  
    print ("failed to copy $file...<br>\n");  
}
```

Siehe auch: [rename\(\)](#).

delete

(unknown)

```
delete -- "Dummy Handbuch Eintrag"
```

Beschreibung

void **delete** (string file)

Dieses ist ein "Dummy Handbuch Eintrag" für diejenigen, die nach einer Funktion wie [unlink\(\)](#) oder [unset\(\)](#) an der falschen Stelle suchen.

Siehe auch [unlink\(\)](#) um Dateien zu löschen, [unset\(\)](#) um Variablen zu löschen.

dirname

(PHP3 , PHP4)

dirname -- Extrahiert den Verzeichnis-Namen aus einer vollständigen Pfadangabe

Beschreibung

string **dirname** (string path)

Bei einer vollständigen Pfadangabe gibt diese Funktion den Verzeichnispfad zurück.

Unter Windows wird sowohl der Slash (/) als auch der Backslash (\) als Trennzeichen bei Pfadangaben benutzt. Unter anderen Betriebssystemen hingegen nur der Slash (/)

Beispiel 1 dirname() Beispiel

```
$path = "/etc/passwd";  
$file = dirname ($path); // $file enthält den wert "/etc"
```

Siehe auch [basename\(\)](#).

diskfreespace

(PHP3 >= 3.0.7, PHP4 >= 4.0b4)

diskfreespace -- Gibt den freien Speicherplatz in einem Verzeichnis zurück

Beschreibung

float **diskfreespace** (string directory)

Diese Funktion gibt den freien Speicherplatz eines Verzeichnisses in Byte zurück.

Beispiel 1 diskfreespace() Beispiel

```
$df = diskfreespace("/"); // $df hält die Anzahl an freien Bytes  
                           // im Verzeichnis "/"
```

fclose

(PHP3 , PHP4)

fclose -- Schließt einen offenen Dateizeiger

Beschreibung

`int fclose (int fp)`

Die Datei, auf die der Dateizeiger *fp* zeigt wird geschlossen.

Gibt bei erfolgreichem Schließen der Datei TRUE, andernfalls FALSE zurück.

Der Datei-Zeiger muss gültig sein, d.h. die Datei, auf die gezeigt wird muss zuvor mit [fopen\(\)](#), [popen\(\)](#) oder [fsockopen\(\)](#) geöffnet worden sein.

feof

(PHP3 , PHP4)

feof -- Prüft, ob der Dateizeiger am Ende der Datei steht

Beschreibung

int **feof** (int fp)

Gibt TRUE zurück, falls der Dateizeiger am Ende der Datei steht oder ein Fehler aufgetreten ist, andernfalls FALSE.

Der Dateizeiger muss gültig sein, d.h. die Datei, auf die gezeigt wird, muss zuvor mit [fopen\(\)](#), [popen\(\)](#) oder [fsockopen\(\)](#) geöffnet worden sein.

[Zurück](#)[Anfang](#)[Vor](#)

fclose

[Hoch](#)

fgetc

fgetc

(PHP3 , PHP4)

fgetc -- Liest das Zeichen, auf welches der Dateizeiger zeigt

Beschreibung

string **fgetc** (int fp)

Gibt einen String zurück, der ein einzelnes Zeichen von der aktuellen Position des Dateizeigers *fp* enthält. Gibt FALSE zurück, wenn der Dateizeiger am Ende der Datei steht (wie die Funktion [feof\(\)](#)).

Der Dateizeiger muss gültig sein, d.h. die Datei, auf die gezeigt wird, muss zuvor mit [fopen\(\)](#), [popen\(\)](#) oder [fsockopen\(\)](#) geöffnet worden sein.

Siehe auch [fread\(\)](#), [fopen\(\)](#), [popen\(\)](#), [fsockopen\(\)](#) und [fgets\(\)](#).

fgetcsv

(PHP3 >= 3.0.8, PHP4)

fgetcsv -- Liest eine Zeile von der Position des Dateizeigers und überprüft diese auf Komma-Separierte-Werte (CSV)

Beschreibung

array **fgetcsv** (int *fp*, int *length* [, string *Trennzeichen* (Delimiter)])

Die Funktion **fgetcsv()** handelt ähnlich der Funktion [fgets\(\)](#), bis auf den Umstand, das **fgetcsv()** die eingeleseene Zeile auf Kommaseparierte Felder hin überprüft. Das Ergebnis wird als Array zurückgegeben. Als Trennzeichen (Delimiter) wird - wenn kein anderes Trennzeichen im dritten optionalen Parameter angegeben wird - das Komma benutzt.

Der Dateizeiger *fp* muss gültig sein, d.h. die Datei, auf die gezeigt wird, muss zuvor mit [fopen\(\)](#), [popen\(\)](#) oder [fsockopen\(\)](#) geöffnet worden sein.

'length' muss größer sein als die längste in der CSV-Datei vorhandene Zeile (um das Erkennen von EndOfLine(Ende der Zeile) Zeichen zu ermöglichen).

fgetcsv() gibt FALSE am Dateiende und bei Lesefehlern zurück.

Eine leere Zeile der CSV-Datei wird als Array mit einem Feld das den Wert NULL hält zurückgegeben. Sie wird nicht als Fehler behandelt.

Beispiel 1 Fgetcsv() Beispiel - Lesen und Ausgabe des Inhalts einer CSV-Datei

```
$row = 1; // Anzahl der Arrays
$fp = fopen ("test.csv","r"); // Datei wird zum Lesen
// geöffnet
while ($data = fgetcsv ($fp, 1000, ",")) { // Daten werden aus der Datei
// in einen Array $data gelesen
    $num = count ($data); // Felder im Array $data
// werden gezählt
    print "<p> $num fields in line $row: <br>";
    $row++; // Anzahl der Arrays wird
// inkrementiert
    for ($c=0; $c<$num; $c++) { // FOR-Schleife, um Felder
        print $data[$c] . "<br>"; // des Arrays auszugeben
    }
}
fclose ($fp);
```

[Zurück](#)
fgetc

[Anfang](#)
[Hoch](#)

[Vor](#)
fgets

fgets

(PHP3 , PHP4)

fgets -- Liest eine Zeile von der Position des Dateizeigers

Beschreibung

string **fgets** (int *fp*, int *length*)

Gibt eine Zeile zurück mit einer in *length* definierten Länge -1 Byte, welche aus der Datei von der aktuellen Position des Dateizeigers *fp* aus ausgelesen wird. Die Zeile endet an einem Zeilenumbruch, EOF (Ende der Datei) oder bei *length* Zeichen (Je nachdem, was eher auftritt).

Wenn ein Fehler auftritt, wird FALSE zurückgegeben.

Häufig auftretende Fehler:

Wenn Sie an die C-Semantik von fgets gewöhnt sind, sollten Sie auf die unterschiedliche Behandlung von EOF (genauer: wie EOF zurückgegeben wird) achten.

Der Datei-Zeiger *fp* muss gültig sein, d.h. die Datei, auf die gezeigt wird, muss zuvor mit [fopen\(\)](#), [popen\(\)](#) oder [fsockopen\(\)](#) geöffnet worden sein.

Ein einfaches Beispiel:

Beispiel 1 Eine Datei Zeile für Zeile einlesen

```
$fd = fopen ("/tmp/inputfile.txt", "r");
while (!feof($fd)) {
    $buffer = fgets($fd, 4096);
    echo $buffer;
}
fclose ($fd);
```

Siehe auch [fread\(\)](#), [fopen\(\)](#), [popen\(\)](#), [fgetc\(\)](#) und [fsockopen\(\)](#).

fgetss

(PHP3 , PHP4)

`fgetss` -- Liest eine Zeile von der aktuellen Position des Dateizeigers und entfernt HTML und PHP-Tags.

Beschreibung

string **fgetss** (int fp, int length [, string allowable_tags])

Diese Funktion ist identisch mit der Funktion [fgets\(\)](#), ausser das **fgetss()** versucht, vorhandene HTML und PHP-Tags aus dem gelesenen Text zu entfernen.

Optional können mit dem dritten Parameter *allowable_tags* Tags spezifiziert werden, die nicht entfernt werden sollen.

Anmerkung: *allowable_tags* wurden in PHP 3.0.13 hinzugefügt, PHP4B3.

Siehe auch [fgets\(\)](#), [fopen\(\)](#), [fsockopen\(\)](#), [popen\(\)](#) und [strip_tags\(\)](#).

file

(PHP3 , PHP4)

file -- Liest eine Datei komplett in einen Array

Beschreibung

array **file** (string filename [, int use_include_path])

Die Funktion **file()** ist identisch mit der Funktion [readfile\(\)](#), bis auf den Umstand, das die eingelesene Datei als Array zurückgegeben wird. Jedes Feld des Arrays korrespondiert mit einer Zeile der Datei. Der Zeilenumbruch bleibt erhalten.

Optional kann der zweiter Parameter benutzt und auf '1' gesetzt werden, wenn in dem [include_path](#) ebenfalls nach der Datei gesucht werden soll.

```
<?php
// liest eine Webseite in einen Array und gibt das Ergebnis aus
$fcontents = file ('http://www.php.net');
while (list ($line_num, $line) = each ($fcontents)) {
    echo "<b>Line $line_num:</b>" . htmlspecialchars($line) . "<br>\n";
}

// liefert eine Webseite in einen String
$fcontents = join ('', file ('http://www.php.net'));
?>
```

Siehe auch [readfile\(\)](#), [fopen\(\)](#), and [popen\(\)](#).

file_exists

(PHP3 , PHP4)

file_exists -- Überprüft, ob eine Datei existiert

Beschreibung

`int file_exists (string filename)`

Gibt TRUE zurück, wenn die mit *filename* spezifizierte Datei existiert, andernfalls FALSE.

File_exists() funktioniert nur mit lokalen Dateien. Die Datei, deren Existenz überprüft werden soll muss über das Dateisystem des Servers erreichbar sein.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe [clearstatcache\(\)](#) für weitere Einzelheiten.

[Zurück](#)

file

[Anfang](#)[Hoch](#)[Vor](#)

fileatime

fileatime

(PHP3 , PHP4)

fileatime -- Gibt Datum und Uhrzeit des letzten Zugriffs auf eine Datei zurück

Beschreibung

int **fileatime** (string filename)

Gibt Datum und Uhrzeit des letzten Zugriffs auf eine Datei zurück, oder FALSE wenn ein Fehler auftrat. Das Datum wird als UNIX-Timestamp zurückgegeben.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe [clearstatcache\(\)](#) für weitere Einzelheiten.

filectime

(PHP3 , PHP4)

filectime -- Gibt Datum und Uhrzeit der letzten Änderung des Dateizeigers Inode zurück

Beschreibung

`int filectime (string filename)`

Gibt Zeit der letzten Änderung des Dateizeigers Inode zurück, oder FALSE wenn ein Fehler auftrat. Die Zeit wird als UNIX-Timestamp zurückgegeben.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe [clearstatcache\(\)](#) für weitere Einzelheiten.

filegroup

(PHP3 , PHP4)

filegroup -- Gibt die Gruppenzugehörigkeit einer Datei zurück

Beschreibung

`int filegroup (string filename)`

Gibt die Gruppen ID des Besitzers einer Datei zurück, oder FALSE wenn ein Fehler auftrat. Die Gruppen ID wird im numerischen Format zurückgegeben. Mit der Funktion [posix_getgrgid\(\)](#) können Sie diese zu ihrem Gruppennamen auflösen.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe [clearstatcache\(\)](#) für weitere Einzelheiten.

Anmerkung: Diese Funktion hat unter Windows-Systemen keinen Effekt.

fileinode

(PHP3 , PHP4)

fileinode -- Gibt Inode-Nummer einer Datei aus

Beschreibung

```
int fileinode (string filename)
```

Gibt die Inode-Nummer einer Datei zurück, oder FALSE wenn ein Fehler auftrat.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe [clearstatcache\(\)](#) für weitere Einzelheiten.

Anmerkung: Diese Funktion hat unter Windows-Systemen keinen Effekt.

filemtime

(PHP3 , PHP4)

filemtime -- Gibt Datum und Uhrzeit der letzten Dateiänderung aus

Beschreibung

`int filemtime (string filename)`

Gibt Datum und Uhrzeit zurück, wann die Datei das letzte mal modifiziert wurde, oder FALSE wenn ein Fehler auftrat. Das Datum wird als UNIX-Timestamp zurückgegeben.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe [clearstatcache\(\)](#) für weitere Einzelheiten.

fileowner

(PHP3 , PHP4)

fileowner -- Gibt den Dateieigentümer aus

Beschreibung

`int fileowner (string filename)`

Gibt die Benutzer ID des Eigentümers einer Datei zurück, oder FALSE wenn ein Fehler auftrat. Die Benutzer ID wird in numerischem Format zurückgegeben und kann mit der Funktion [posix_getpwuid\(\)](#) zum Benutzernamen aufgelöst werden.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe [clearstatcache\(\)](#) für weitere Einzelheiten.

Anmerkung: Diese Funktion hat unter Windows keinen Effekt.

fileperms

(PHP3 , PHP4)

fileperms -- Gibt die Zugriffsrechte einer Datei aus

Beschreibung

```
int fileperms (string filename)
```

Gibt die Zugriffsrechte einer Datei zurück, oder FALSE wenn ein Fehler auftrat.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe [clearstatcache\(\)](#) für weitere Einzelheiten.

filesize

(PHP3 , PHP4)

filesize -- Gibt die Größe einer Datei aus

Beschreibung

`int filesize (string filename)`

Gibt die Größe einer Datei zurück, oder FALSE wenn ein Fehler auftrat.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe [clearstatcache\(\)](#) für weitere Einzelheiten.

filetype

(PHP3 , PHP4)

filetype -- Gibt den Typ einer Datei zurück

Beschreibung

string **filetype** (string filename)

Gibt den Typ einer Datei zurück. Mögliche Werte sind fifo, char, dir, block, link, file und unknown.

Gibt FALSE zurück, wenn ein Fehler auftrat.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe [clearstatcache\(\)](#) für weitere Einzelheiten.

flock

(PHP3 >= 3.0.7, PHP4)

flock -- Portables Datei-Verriegelungs-Verfahren

Beschreibung

bool **flock** (int *fp*, int *operation*)

PHP liefert einen portablen Weg, Dateien in einer "beratenden" Art und Weise zu verriegeln. D.h. alle zugreifende Programme müssen die gleiche Art und Weise der Verriegelung benutzen, oder es funktioniert nicht.

flock() arbeitet mit *fp* welcher ein existierender Dateizeiger sein muss. Der Parameter *operation* bestimmt die Art der Verriegelung und kann einen der folgenden Werte annehmen:

- Um eine verteilte, nur lesende Verriegelung zu erhalten, setzten Sie *operation* auf 1.
- Um eine exklusive, nur schreibende Verriegelung zu erhalten, setzen Sie *operation* auf 2.
- Um eine Verriegelung freizugeben (lesend oder schreibend), setzten Sie *operation* auf 3.
- Wenn Sie Zugriffe während der Verriegelung durch **flock()** erlauben wollen, fügen Sie 4 zu *operation* hinzu.

flock() erlaubt es Ihnen ein einfaches Leser/Schreiber Modell zu erstellen, welches auf fast jedem Betriebssystem benutzt werden kann (UNICES- und Windows- Systeme).

flock() gibt TRUE bei Erfolg, FALSE wenn ein Fehler auftrat (z.B. wenn eine Verriegelung nicht erstellt werden konnte) zurück.

fopen

(PHP3 , PHP4)

fopen -- Öffnet eine Datei oder URL

Beschreibung

`int fopen (string filename, string mode [, int use_include_path])`

Wenn *filename* mit "http://" (nicht case sensitive) beginnt, wird zu dem spezifiziertem Server eine HTTP 1.0 Connection geöffnet und ein Dateizeiger auf den Anfang des Textes der Antwort zurückgegeben.

Weiterleitungen (HTTP Redirects) werden nicht unterstützt, die Datei muss daher vollständig und mit abschließendem "/" beschrieben werden.

Wenn *filename* mit "ftp://" (nicht case sensitive) beginnt, wird eine ftp-Connection zu dem spezifiziertem Server geöffnet und ein Zeiger auf die angeforderte Datei zurückgegeben. Wenn der Server keinen passiven FTP-Modus unterstützt, wird diese Methode fehlschlagen. Sie können Dateien entweder zum Schreiben oder zum Lesen öffnen, jedoch nicht für beides gleichzeitig.

Wenn *filename* "php://stdin", "php://stdout", oder "php://stderr" ist, wird der korrespondierende stdio stream geöffnet (Feature wurde in Version 3.0.13 eingeführt, in früheren Versionen muss der Dateiname wie z.B. "/dev/stdin" oder "/dev/fd/0" benutzt werden, um auf die stdio streams zuzugreifen).

Wenn *filename* mit irgendetwas anderem beginnt, wird die Datei vom lokalen Dateisystem geöffnet und ein Dateizeiger auf die geöffnete Datei zurückgegeben.

Wenn das Öffnen scheitert, gibt die Funktion FALSE zurück.

mode kann einen der folgenden Werte einnehmen:

- 'r' - öffnet die Datei nur zum Lesen und positioniert den Dateizeiger auf den Anfang der Datei.
- 'r+' - Öffnet die Datei zum Lesen und Schreiben und setzt den Dateizeiger auf den Anfang der Datei.
- 'w' - Öffnet die Datei nur zum Schreiben und setzt den Dateizeiger auf den Anfang der Datei sowie die Länge der Datei auf 0 Byte. Wenn die Datei nicht existiert wird versucht sie anzulegen.
- 'w+' - Öffnet die Datei zum Lesen und Schreiben und setzt den Dateizeiger auf den Anfang der Datei sowie die Länge der Datei auf 0 Byte. Wenn die Datei nicht existiert, wird versucht sie anzulegen.

- 'a' - Öffnet die Datei nur zum Schreiben. Positioniert den Dateizeiger auf das Ende der Datei. Wenn die Datei nicht existiert, wird versucht sie anzulegen.
- 'a+' - Öffnet die Datei zum Lesen und Schreiben. Positioniert den Dateizeiger auf das Ende der Datei. Wenn die Datei nicht existiert, wird versucht sie anzulegen.

Zusätzlich kann *mode* der Buchstabe 'b' hinzugefügt werden, der die Behandlung von Binär-Dateien erlaubt. Dieses ist sinnvoll auf Systemen, welche zwischen Binär- und Text-Dateien unterscheiden. Wenn das Feature nicht gebraucht wird, wird es einfach ignoriert.

Sie können optional ein dritten Parameter benutzen und diesen auf '1' setzten, wenn Sie auch im [include_path](#) nach der Datei suchen möchten.

Beispiel 1 fopen() Beispiel

```
$fp = fopen ("/home/rasmus/simi.txt", "r");  
$fp = fopen ("http://www.php.net/", "r");  
$fp = fopen ("ftp://user:password@example.com/", "w");
```

Wenn Sie Probleme haben mit dem Lesen oder Schreiben von Dateien (Wenn Sie PHP als Servermodul benutzen), stellen Sie zunächst sicher, dass die Dateien und Verzeichnisse die Sie benutzen wollen auch für den Server-Prozeß zugänglich sind (Rechtevergabe).

Achten Sie auf Windows-Systemen darauf, dass Sie als Verzeichnistrenner normale Schrägstriche '/' benutzen um plattformunabhängig programmieren zu können. Sollten Sie dennoch Backslashes verwenden, vergessen Sie nicht diese zu escapen '\\'

```
$fp = fopen ("c:\\data\\info.txt", "r");
```

Siehe auch [fclose\(\)](#), [fsockopen\(\)](#) und [popen\(\)](#).

[Zurück](#)[flock](#)[Anfang](#)[Hoch](#)[Vor](#)[fpassthru](#)

fpassthru

(PHP3 , PHP4)

fpassthru -- Gibt alle verbleibenden Daten eines Dateizeigers direkt aus.

Beschreibung

```
int fpassthru (int fp)
```

Die Funktion liest alle Daten vom Dateizeiger bis EOF aus und gibt die Daten auf der Standartausgabe (Browser) aus.

Wenn ein Fehler auftritt, gibt **fpassthru()** FALSE zurück.

Der Dateizeiger muss gültig sein und auf eine erfolgreich durch [fopen\(\)](#), [popen\(\)](#) oder [fsockopen\(\)](#) geöffnete Datei zeigen. Die Datei wird geschlossen, wenn **fpassthru()** die Datei komplett gelesen hat (hinterläßt nutzlosen *fp*-Dateizeiger).

Wenn Sie den Inhalt einer Datei nur auf der Standartausgabe (Browser) ausgeben wollen, sollten Sie besser [readfile\(\)](#) benutzen, bei der Sie nicht zuvor die Datei mittels [fopen\(\)](#) geöffnet haben müssen.

Siehe auch [readfile\(\)](#), [fopen\(\)](#), [popen\(\)](#) und [fsockopen\(\)](#)

fputs

(PHP3 , PHP4)

fputs -- Schreibt Daten an die Position des Dateizeigers

Beschreibung

int **fputs** (int fp, string str [, int length])

Fputs() ist ein Alias für die Funktion [fwrite\(\)](#), und ist in jeder Hinsicht identisch mit ihr. Beachten Sie, dass der Parameter *length* optional ist, und dass, wenn er nicht benutzt wird, die gesamte Zeichenkette geschrieben wird.

fread

(PHP3 , PHP4)

fread -- Liest Binär-Dateien aus einer Datei

Beschreibung

string **fread** (int fp, int length)

Fread() liest *length* Bytes vom Dateizeiger, maximal jedoch bis zum Dateiende.

```
// liest den Inhalt einer Datei in einen String
$filename = "/usr/local/simi.txt";
$fd = fopen ($filename, "r");
$contents = fread ($fd, filesize ($filename));
fclose ($fd);
```

Siehe auch [fwrite\(\)](#), [fopen\(\)](#), [fsockopen\(\)](#), [popen\(\)](#), [fgets\(\)](#), [fgetss\(\)](#), [file\(\)](#) und [fpassthru\(\)](#).

fseek

(PHP3 , PHP4)

fseek -- Positioniert den Dateizeiger um einen Offset 'offset' vor oder zurück

Beschreibung

int **fseek** (int fp, int offset [, int whence])

Setzt den Dateizeiger einer mit *fp* angegebenen Datei auf eine neue Position. Diese neue Position, angegeben in Bytes vom Begin der Datei an, wird definiert durch Hinzufügen von *offset* zu der Position spezifiziert durch *whence*, deren Werte wie folgt definiert sind:

SEEK_SET - Setzt Position gleich *offset* bytes.

SEEK_CUR - Setzt Position auf die aktuelle Stelle plus *offset*.

SEEK_END - Setzt die Position ans Ende der Datei plus *offset*.

Wenn *whence* nicht angegeben wurde, wird SEEK_SET angenommen.

Bei Erfolg wird '0' zurückgegeben, andernfalls '-1'. Beachten Sie, dass die Überschreitung des Dateiendes (EOF) nicht als Fehler bewertet wird.

Diese Funktion sollte nicht auf Dateizeiger angewandt werden, die durch die Verwendung von "http://" oder "ftp://" bei der Funktion [fopen\(\)](#) erhalten wurden.

Siehe auch [ftell\(\)](#) und [rewind\(\)](#).

ftell

(PHP3 , PHP4)

ftell -- Ermittelt die aktuelle Position des Dateizeigers

Beschreibung

`int ftell (int fp)`

Gibt die aktuelle Position des Dateizeigers *fp* zurück, z.B. seinen Offset im Dateistream.

Wenn ein Fehler auftritt, wird FALSE zurückgegeben.

Der Dateizeiger muss gültig sein und auf eine erfolgreich durch [fopen\(\)](#), [popen\(\)](#) oder [fsockopen\(\)](#) geöffnete Datei zeigen.

Siehe auch [fopen\(\)](#), [popen\(\)](#), [fseek\(\)](#) und [rewind\(\)](#).

fwrite

(PHP3 , PHP4)

fwrite -- Schreibt Binärdaten in eine Datei

Beschreibung

`int fwrite (int fp, string string [, int length])`

Fwrite() schreibt den Inhalt einer Zeichenkette *string* in die Datei, auf welche der *fp* Dateizeiger zeigt. Wenn der *length* Parameter gegeben ist, wird das Schreiben nach *length* Bytes beendet, oder wenn das Dateiende (EOF) erreicht ist, je nachdem, was eher eintritt.

Bedenken Sie, dass wenn der *length* Parameter gesetzt ist, die [magic_quotes_runtime](#) Konfigurations-Option ignoriert wird und somit keine Slashes mehr von der Zeichenkette *string* entfernt werden.

Siehe auch [fread\(\)](#), [fopen\(\)](#), [fsockopen\(\)](#), [popen\(\)](#) und [fputs\(\)](#).

set_file_buffer

(PHP3 >= 3.0.8, PHP4 >= 4.0.1)

set_file_buffer -- Setzt die Dateipufferung für einen gegebenen Dateizeiger

Beschreibung

int set_file_buffer (int fp, int buffer)

set_file_buffer() setzt die Pufferung für Schreiboperationen eines gegebenen Dateizeigers *fp* auf *buffer* Bytes. Wenn *buffer* auf 0 gesetzt ist oder 0 ist, wird die Schreibpufferung ausgeschaltet.

Die Funktion gibt 0 bei Erfolg, oder EOF wenn die Anfrage nicht erfolgreich war zurück.

Bedenken Sie, dass der Standartwert beim Aufruf von [fopen\(\)](#) mit set_file_buffer immer 8K ist.

Siehe auch [fopen\(\)](#).

is_dir

(PHP3 , PHP4)

is_dir -- Ermittelt, ob der gegebene Dateiname ein Verzeichnis ist

Beschreibung

bool **is_dir** (string filename)

Gibt TRUE zurück, wenn der Dateiname existiert und es sich hierbei um ein Verzeichnis handelt.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe auch [clearstatcache\(\)](#) für weitere Einzelheiten.

Siehe auch [is_file\(\)](#) und [is_link\(\)](#).

is_executable

(PHP3 , PHP4)

is_executable -- Ermittelt, ob eine Datei ausführbar ist

Beschreibung

bool **is_executable** (string filename)

Gibt TRUE zurück, wenn die Datei existiert und ausführbar ist.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe auch [clearstatcache\(\)](#) für weitere Einzelheiten.

Siehe auch [is_file\(\)](#) und [is_link\(\)](#).

is_file

(PHP3 , PHP4)

is_file -- Ermittelt, ob der Dateiname eine reguläre Datei ist

Beschreibung

bool **is_file** (string filename)

Gibt TRUE zurück, wenn die Datei existiert und eine reguläre Datei ist.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe auch [clearstatcache\(\)](#) für weitere Einzelheiten.

Siehe auch [is_dir\(\)](#) und [is_link\(\)](#).

is_link

(PHP3 , PHP4)

is_link -- Ermittelt, ob der Dateiname ein symbolischer Link ist

Beschreibung

bool **is_link** (string filename)

Gibt TRUE zurück, wenn der Dateiname existiert und es sich um einen symbolischen Link handelt.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe auch [clearstatcache\(\)](#) für weitere Einzelheiten.

Siehe auch [is_dir\(\)](#) und [is_file\(\)](#).

Anmerkung: Diese Funktion hat auf Windows-Systemen keinen Effekt.

is_readable

(PHP3 , PHP4)

is_readable -- Ermittelt ob eine Datei lesbar ist

Beschreibung

bool **is_readable** (string filename)

Gibt TRUE zurück wenn die Datei existiert und sie lesbar ist.

Bedenken Sie, dass PHP mit der Benutzer ID auf die Datei zugreift, unter der der Webserver läuft (oftmals ist dieses 'nobody'). Beschränkungen durch safe_mode werden nicht berücksichtigt.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe auch [clearstatcache\(\)](#) für weitere Einzelheiten.

Siehe auch [is_writeable\(\)](#).

is_writeable

(PHP3 , PHP4)

is_writeable -- Ermittelt, ob in eine Datei geschrieben werden kann

Beschreibung

bool **is_writeable** (string filename)

Gibt TRUE zurück, wenn die Datei existiert und nicht schreibgeschützt ist. Das Dateinamen-Argument kann ein Verzeichnisname sein. So können Sie überprüfen, ob in ein Verzeichnis geschrieben werden kann.

Bedenken Sie, dass PHP mit der Benutzer ID auf die Datei zugreift, unter der der Webserver läuft (oftmals ist dieses 'nobody'). Beschränkungen durch safe_mode werden nicht berücksichtigt.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe auch [clearstatcache\(\)](#) für weitere Einzelheiten.

Siehe auch [is_readable\(\)](#).

link

(PHP3 , PHP4)

link -- Erzeugt einen absoluten Link

Beschreibung

int **link** (string target, string link)

Link() erzeugt einen absoluten Link mit dem Ziel *target*.

Siehe auch [symlink\(\)](#) um symbolische Links zu erstellen und [readlink\(\)](#) zusammen mit [linkinfo\(\)](#).

Anmerkung: Diese Funktion hat unter Windows-Systemen keinen Effekt.

linkinfo

(PHP3 , PHP4)

linkinfo -- Ermittelt Informationen über einen Link

Beschreibung

int **linkinfo** (string path)

Linkinfo() gibt das st_dev Feld des UNIX C stat structure resultierend aus dem lstat System Call zurück. Diese Funktion wird benutzt, um zu verifizieren ob ein Link (auf den mit *path* gezeigt wird) wirklich existiert (Hierbei wird die gleiche Methode benutzt, die in dem S_ISLNK Makro in stat.h definiert ist). Gibt 0 oder FALSE zurück wenn ein Fehler auftrat.

Siehe auch [symlink\(\)](#), [link\(\)](#), und [readlink\(\)](#).

Anmerkung: Diese Funktion hat unter Windows keinen Effekt.

mkdir

(PHP3 , PHP4)

mkdir -- Erstellt ein Verzeichnis

Beschreibung

`int mkdir (string pathname, int mode)`

Versucht ein Verzeichnis mit dem spezifiziertem Namen zu erstellen.

Bedenken Sie, dass sie wahrscheinlich den Mode als oktalen Wert spezifizieren wollen, was bedeutet, dass er eine voranstehende Null haben sollte.

```
mkdir ("/path/to/my/dir", 0700);
```

Gibt TRUE bei Erfolg, andernfalls FALSE zurück.

Siehe auch [rmdir\(\)](#).

pclose

(PHP3 , PHP4)

pclose -- Schließt einen Prozess-Dateizeiger

Beschreibung

`int pclose (int fp)`

Schließt einen Prozess-Dateizeiger, der durch einen Aufruf von [popen\(\)](#) geöffnet wurde.

Der Dateizeiger muss gültig sein und durch einen erfolgreichen Funktionsaufruf von [popen\(\)](#) zurückgegeben worden sein.

Gibt den End-Status eines laufenden Prozesses zurück.

Siehe auch [popen\(\)](#).

popen

(PHP3 , PHP4)

popen -- Öffnet einen Prozesszeiger

Beschreibung

int **popen** (string command, string mode)

Öffnet eine Verbindung zu einem Prozess, der durch die Anweisung *command* ausgeführt wurde.

Gibt einen Dateizeiger zurück, der identisch ist mit dem, der bei der Funktion [fopen\(\)](#) zurückgegeben wird, ausser dass dieser nur in eine Richtung funktioniert (entweder lesend oder schreibend) und mit der Funktion [pclose\(\)](#) geschlossen werden muss. Dieser Zeiger kann mit den Funktionen [fgets\(\)](#), [fgetss\(\)](#) und [fputs\(\)](#) benutzt werden.

Wenn ein Fehler auftritt, gibt diese Funktion FALSE zurück.

```
$fp = popen ("/bin/ls", "r");
```

Siehe auch [pclose\(\)](#).

readfile

(PHP3 , PHP4)

readfile -- Gibt eine Datei aus

Beschreibung

int readfile (string filename [, int use_include_path])

Liest den Inhalt einer Datei und gibt ihn an der Standartausgabe (Browser) aus.

Gibt die Anzahl der Bytes einer Datei zurück. Wenn ein Fehler auftritt wird FALSE zurückgegeben und eine Fehlermeldung wird auf der Standartausgabe ausgegeben. Die Ausgabe einer Fehlermeldung kann durch Voranstellen eines '@' verhindert werden (@readfile).

Wenn *filename* mit "http://" (nicht case sensitive) beginnt, wird eine HTTP 1.0 Connection zu dem spezifiziertem Server geöffnet und der Text der Antwort wird auf die Standartausgabe geschrieben.

Weiterleitungen (HTTP Redirects) werden nicht unterstützt, daher muss die Datei vollständig mit abschliessendem '/' geschrieben werden.

Wenn *filename* mit "ftp://" (nicht case sensitive) beginnt, wird eine FTP-Connection zu dem spezifiziertem Server geöffnet und die angeforderte Datei wird auf der Standartausgabe ausgegeben. Wenn der Server keinen passiven Mode unterstützt, wird diese Funktion fehlschlagen.

Wenn *filename* mit keinem dieser Zeichenketten beginnt, wird die Datei vom lokalen Dateisystem geöffnet und ihr Inhalt auf der Standartausgabe ausgegeben.

Sie können optional den zweiten Paratmeter benutzen und diesen auf '1' setzen, wenn Sie auch im [include_path](#) nach der Datei suchen möchten.

Siehe auch [fpassthru\(\)](#), [file\(\)](#), [fopen\(\)](#), [include\(\)](#), [require\(\)](#) und [virtual\(\)](#).

readlink

(PHP3 , PHP4)

readlink -- Gibt das Ziel eines symbolischen Links zurück

Beschreibung

string **readlink** (string path)

Readlink() ist von der Funktionalität her identisch mit der 'readlink C Funktion' und gibt den Inhalt des Pfades des symbolischen Links zurück oder 0, wenn der Aufruf fehlschlug.

Siehe auch [symlink\(\)](#), [readlink\(\)](#) und [linkinfo\(\)](#).

Anmerkung: Diese Funktion hat unter Windows keinen Effekt.

rename

(PHP3 , PHP4)

rename -- Benennt eine Datei um

Beschreibung

```
int rename (string oldname, string newname)
```

Versucht eine Datei *oldname* in *newname* umzubenennen.

Gibt TRUE bei Erfolg, andernfalls FALSE zurück.

rewind

(PHP3 , PHP4)

rewind -- Setzt den Dateizeiger auf das erste Byte der Datei

Beschreibung

int **rewind** (int fp)

Setzt den Dateizeiger (Datei-Positions-Indikator) für *fp* an den Anfang des Dateistreams.

Wenn ein Fehler auftritt, gibt der Funktionsaufruf 0 zurück.

Der Dateizeiger muss gültig sein und auf eine Datei zeigen, die zuvor erfolgreich durch [fopen\(\)](#) geöffnet wurde.

Siehe auch [fseek\(\)](#) und [ftell\(\)](#).

rmdir

(PHP3 , PHP4)

rmdir -- Löscht ein Verzeichnis

Beschreibung

int **rmdir** (string dirname)

Versucht ein Verzeichnis spezifiziert durch die Pfadangabe zu löschen. Das Verzeichnis muss leer sein und die relevanten Rechte müssen erteilt sein.

Wenn ein Fehler auftritt wird 0 zurückgegeben.

Siehe auch [mkdir\(\)](#).

stat

(PHP3 , PHP4)

stat -- Ermittelt diverse Informationen über eine Datei

Beschreibung

array **stat** (string filename)

Versucht Informationen über eine Datei spezifiziert in *filename* auszugeben.

Gibt einen Array mit folgenden statistischen Informationen als Array- Elemente zurück:

1. Laufwerk
2. inode
3. inode protection mode
4. Anzahl der Links
5. Benutzer ID des Inhabers
6. Gruppen ID des Inhabers
7. Laufwerkstyp wenn Inode-Laufwerk *
8. Größe in Bytes
9. Zeitpunkt des letzten Zugriffs
10. Zeitpunkt der letzten Modifizierung
11. Zeitpunkt der letzten Änderung
12. Blockgröße für das Dateisystem I/O *
13. Anzahl der zugewiesenen Blöcke

* - nur gültig auf Systemen, die st_blksize unterstützen. Andere Systeme (z.B. Windows) geben '-1' zurück.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe auch [clearstatcache\(\)](#) für weitere Einzelheiten.

lstat

(PHP3 >= 3.0.4, PHP4)

`lstat --` Ermittelt Informationen über einen symbolischen Link.

Beschreibung

array **lstat** (string filename)

Ermittelt statistische Informationen über einen symbolischen Link spezifiziert in *filename*. Diese Funktion ist identisch mit der Funktion [stat\(\)](#) ausser dass, wenn der Parameter *filename* ein symbolischer Link ist, dessen Status zurückgegeben wird und nicht der Status der Datei, auf die der symbolische Link zeigt.

Gibt einen Array mit statistischen Informationen über die Datei als Array-Elemente zurück:

1. Laufwerk
2. inode
3. Anzahl der Links
4. Benutzer ID des Inhabers
5. Gruppen ID des Inhabers
6. Laufwerkstyp wenn inode Laufwerk *
7. Größe in Bytes
8. Zeitpunkt des letzten Zugriffs
9. Zeitpunkt der letzten Modifikation
10. Zeitpunkt der letzten Änderung
11. Blockgröße für das Dateisystem I/O *
12. Anzahl der zugewiesenen Blöcke

* - nur gültig auf Systemen, die `st_blksize` unterstützen. Andere Systeme (z.B. Windows) geben '-1' zurück.

Das Ergebnis dieses Funktionsaufrufes wird zwischengespeichert. Siehe auch [clearstatcache\(\)](#) für weitere Einzelheiten.

realpath

(PHP4 >= 4.0b4)

realpath -- Return 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.

Beispiel 1 realpath() example

```
$real_path = realpath("../../index.php");
```

symlink

(PHP3 , PHP4)

symlink -- Erzeugt einen symbolischen Link

Beschreibung

int **symlink** (string target, string link)

Symlink() erzeugt einen symbolischen Link von dem gegebenen Parameter *target* mit dem spezifiziertem Namen *link*.

Siehe auch [link\(\)](#) um absolute Links zu erstellen, und [readlink\(\)](#) zusammen mit [linkinfo\(\)](#).

Anmerkung: Diese Funktion hat unter Windows-Systemen keinen Effekt.

tempnam

(PHP3 , PHP4)

tempnam -- Erzeugt einen eindeutigen Dateinamen

Beschreibung

string **tempnam** (string *dir*, string *prefix*)

Erzeugt einen eindeutigen einmaligen temporären Dateinamen in dem spezifizierten Verzeichnis. Wenn das Verzeichnis nicht existiert, wird **tempnam()** einen Dateinamen im temporären Verzeichnis des Systems erstellen.

Das Verhalten von **tempnam()** ist systemabhängig. Unter Windows wird die TMP-Umgebungsvariable den *dir* Parameter überschreiben, unter Linux-Systemen hat die TMPDIR Umgebungsvariable Vorrang, während SVR4 immer den spezifizierten Verzeichnisnamen benutzen wird, wenn dieses existiert. Wenn Sie nicht genau wissen, wie Ihr System **tempnam()** behandelt, dann konsultieren Sie bitte Ihre System-Dokumentation.

Gibt den neuen temporären Dateinamen bei Erfolg, oder eine NULL-Zeichenkette wenn ein Fehler auftritt zurück.

Beispiel 1 Tempnam() Beispiel

```
$tmpfname = tempnam ("/tmp", "FOO");
```

touch

(PHP3 , PHP4)

touch -- Setzt das Datum der letzten Änderung einer Datei

Beschreibung

int **touch** (string filename [, int time])

Versucht das Datum der letzten Modifikation einer Datei spezifiziert durch *filename* auf den Wert spezifiziert durch *time* zu setzten. Wenn die Option *time* nicht angegeben wurde, benutzt die Funktion die aktuelle Zeit.

Wenn die Datei nicht existiert, wird sie erzeugt.

Gibt TRUE bei Erfolg, andernfalls FALSE zurück.

Beispiel 1 Touch() Beispiel

```
if (touch ($DateiName)) {  
    print "$DateiName Modifizierungs-Zeit wurde auf das aktuelle  
    Datum und und die aktuelle Uhrzeit gesetzt.";  
} else {  
    print "Tut mir leid, Modifikations-Zeit von $DateiName konnte  
    nicht geändert werden.";  
}
```

umask

(PHP3 , PHP4)

umask -- Ändert die aktuelle umask (Zugriffsrechte)

Beschreibung

int **umask** (int mask)

Umask() setzt PHP's umask auf *mask* & 0777 und gibt die alte umask zurück. Wenn PHP als Servermodul benutzt wird, wird die umask zurückgespeichert wenn die jeweilige Anfrage beendet ist.

Umask() ohne Argumente gibt einfach die aktuelle umask zurück.

unlink

(PHP3 , PHP4)

unlink -- Löscht eine Datei

Beschreibung

int **unlink** (string filename)

Löscht *filename*. Ähnlich der UNIX C unlink() Funktion.

Gibt 0 oder FALSE zurück, wenn ein Fehler auftritt.

Siehe auch [rmdir\(\)](#) um Verzeichnisse zu löschen.

Anmerkung: Diese Funktion bleibt unter Windows-Systemen ohne Effekt.

XIX. Forms Data Format functions

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.

Anmerkung: Currently Adobe only provides a libc5 compatible version for Linux. Tests with glibc2 resulted in a segmentation fault. If somebody is able to make it work, please comment on this page.

Anmerkung: 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. They should be in fdftk-dir/include and fdftk-dir/lib. This will not be the case if you just unpack the FdfTk distribution.

The general idea of FDF is similar to HTML forms. The difference is basically the format how filled in 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.

The following examples shows just the evaluation of form data.

Beispiel 1 Evaluating a FDF document

```
<?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
// 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("test.fdf");
$volume = fdf_get_value($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);
?>

```

Inhaltsverzeichnis

[fdf_open](#) Open a FDF document
[fdf_close](#) Close an FDF document
[fdf_create](#) Create a new FDF document
[fdf_save](#) Save a FDF document
[fdf_get_value](#) Get the value of a field
[fdf_set_value](#) Set the value of a field
[fdf_next_field_name](#) Get the next field name
[fdf_set_ap](#) Set the appearance of a field
[fdf_set_status](#) Set the value of the /STATUS key
[fdf_get_status](#) Get the value of the /STATUS key
[fdf_set_file](#) Set the value of the /F key
[fdf_get_file](#) Get the value of the /F key

[Zurück](#)

unlink

[Anfang](#)

[Hoch](#)

[Vor](#)

fdf_open

fdf_open

(PHP3 >= 3.0.6, PHP4)

fdf_open -- Open a FDF document

Description

int **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. Currently, the file has to be created 'manually' by using [fopen\(\)](#) and writing the content of HTTP_FDF_DATA with [fwrite\(\)](#) into it. A mechanism like for HTML form data where for each input field a variable is created does not exist.

Beispiel 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_close\(\)](#).

fdf_close

(PHP3 >= 3.0.6, PHP4)

fdf_close -- Close an FDF document

Description

```
void fdf_close (int fdf_document)
```

The **fdf_close()** function closes the FDF document.

See also [fdf_open\(\)](#).

fdf_create

(PHP3 >= 3.0.6, PHP4)

fdf_create -- Create a new FDF document

Description

int **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.

Beispiel 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_save

(PHP3 >= 3.0.6, PHP4)

fdf_save -- Save a FDF document

Description

int **fdf_save** (string filename)

The **fdf_save()** function saves a FDF document. The FDF Toolkit provides a way to output the document to stdout if the parameter *filename* is '.'. This does not work if PHP is used as an apache module. In such a case one will have to write to a file and use e.g. [fpassthru\(\)](#) to output it.

See also [fdf_close\(\)](#) and example for [fdf_create\(\)](#).

fdf_get_value

(PHP3 >= 3.0.6, PHP4)

fdf_get_value -- Get the value of a field

Description

string **fdf_get_value** (int fdf_document, string fieldname)

The **fdf_get_value()** function returns the value of a field.

See also [fdf_set_value\(\)](#).

[Zurück](#)[Anfang](#)[Vor](#)

fdf_save

[Hoch](#)

fdf_set_value

fdf_set_value

(PHP3 >= 3.0.6, PHP4)

fdf_set_value -- Set the value of a field

Description

void **fdf_set_value** (int fdf_document, string fieldname, string value, int isName)

The **fdf_set_value()** function sets the value of a field. The last parameter determines if the field value is to be converted to a PDF Name (*isName* = 1) or set to a PDF String (*isName* = 0).

See also [fdf_get_value\(\)](#).

fdf_next_field_name

(PHP3 >= 3.0.6, PHP4)

fdf_next_field_name -- Get the next field name

Description

string **fdf_next_field_name** (int 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 paramter is NULL.

See also **fdf_set_field()**, **fdf_get_field()**.

fdf_set_ap

(PHP3 >= 3.0.6, PHP4)

fdf_set_ap -- Set the appearance of a field

Description

```
void fdf_set_ap (int 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 1=FDFNormalAP, 2=FDFRolloverAP, 3=FDFDownAP.

[Zurück](#)[Anfang](#)[Vor](#)

fdf_next_field_name

[Hoch](#)

fdf_set_status

fdf_set_status

(PHP3 >= 3.0.6, PHP4)

fdf_set_status -- Set the value of the /STATUS key

Description

void **fdf_set_status** (int fdf_document, string status)

The **fdf_set_status()** sets the value of the /STATUS key.

See also [fdf_get_status\(\)](#).

fdf_get_status

(PHP3 >= 3.0.6, PHP4)

fdf_get_status -- Get the value of the /STATUS key

Description

string **fdf_get_status** (int fdf_document)

The **fdf_get_status()** returns the value of the /STATUS key.

See also [fdf_set_status\(\)](#).

fdf_set_file

(PHP3 >= 3.0.6, PHP4)

fdf_set_file -- Set the value of the /F key

Description

void **fdf_set_file** (int fdf_document, string filename)

The **fdf_set_file()** sets the value of the /F key. The /F key is just a reference to a PDF form which is to be populated with data. In a web environment it is a URL (e.g. http://testfdf/resultlabel.pdf).

See also [fdf_get_file\(\)](#) and example for [fdf_create\(\)](#).

fdf_get_file

(PHP3 >= 3.0.6, PHP4)

fdf_get_file -- Get the value of the /F key

Description

string **fdf_get_file** (int fdf_document)

The [fdf_set_file\(\)](#) returns the value of the /F key.

See also [fdf_set_file\(\)](#).

XX. FTP-Funktionen

FTP ist die Abkürzung für File Transfer Protocol (Datei-Übertragungs-Protokoll).

Die folgenden Konstanten sind definiert, sobald das FTP-Modul benutzt wird: FTP_ASCII und FTP_BINARY.

Inhaltsverzeichnis

[ftp_connect](#) Stellt eine FTP-Verbindung her

[ftp_login](#) Anmelden einer FTP-Verbindung (Login)

[ftp_pwd](#) Gibt den aktuellen Verzeichnis-Namen zurück

[ftp_cdup](#) Wechselt in das um eine Ebene höhere Verzeichnis

[ftp_chdir](#) Verzeichnis-Wechsel auf einem FTP-Server

[ftp_mkdir](#) Erzeugt ein Verzeichnis

[ftp_rmdir](#) Löscht ein Verzeichnis

[ftp_nlist](#) Gibt eine Liste der im angegebenen Verzeichnis enthaltenen Dateien zurück

[ftp_rawlist](#) Gibt eine detaillierte Liste der Dateien in einem angegebenen Verzeichnis zurück

[ftp_systype](#) Ermittelt den Systemtyp des entfernten FTP-Servers

[ftp_pasv](#) Schaltet den passiven Modus ein oder aus

[ftp_get](#) Liest eine Datei des FTP-Servers und speichert sie lokal (download)

[ftp_fget](#) Lädt eine Datei vom FTP-Server und speichert sie in eine geöffnete, lokale Datei (download)

[ftp_put](#) Überträgt eine Datei auf einen FTP-Server (upload)

[ftp_fput](#) Überträgt eine geöffnete Datei auf einen FTP-Server (upload)

[ftp_size](#) Ermittelt die Dateigrösse einer angegebenen Datei

[ftp_mdtm](#) Ermittelt die letzte Änderungszeit der angegebenen Datei

[ftp_rename](#) Benennt eine Datei auf dem FTP-Server um

[ftp_delete](#) Löscht eine Datei auf dem FTP-Server

[ftp_site](#) Sendet ein SITE-Kommando zum Server

[ftp_quit](#) Schließt / beendet eine FTP-Verbindung

ftp_connect

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

ftp_connect -- Stellt eine FTP-Verbindung her

Beschreibung:

int **ftp_connect** (string Host [, int Port])

Gibt bei Erfolg einen Stream zurück, sonst False.

ftp_connect() stellt eine FTP-Verbindung zum angegebenen *Host* her. Der *Port*-Parameter enthält optional einen alternativer Port für die Verbindung. Fehlt er oder ist er Null, wird der Standard-FTP-Port 21 benutzt.

ftp_login

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

ftp_login -- Anmelden einer FTP-Verbindung (Login)

Beschreibung:

int **ftp_login** (int ftp_stream, string Benutzername, string Passwort)

Wenn die Anmeldung erfolgreich war, wird true, anderenfalls false zurück gegeben.

Anmeldung beim angegebenen FTP-Stream.

[Zurück](#)[Anfang](#)[Vor](#)

ftp_connect

[Hoch](#)

ftp_pwd

ftp_pwd

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

ftp_pwd -- Gibt den aktuellen Verzeichnis-Namen zurück

Beschreibung:

```
int ftp_pwd (int ftp_stream)
```

Gibt das aktuelle Verzeichnis oder - im Fehlerfall - false zurück.

[Zurück](#)

ftp_login

[Anfang](#)[Hoch](#)[Vor](#)

ftp_cdup

ftp_cdup

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

ftp_cdup -- Wechselt in das um eine Ebene höhere Verzeichnis

Beschreibung:

int **ftp_cdup** (int ftp_stream)

Bei Erfolg wird true, ansonsten false zurück gegeben.

Geht eine Verzeichnisebene höher.

ftp_chdir

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

ftp_chdir -- Verzeichnis-Wechsel auf einem FTP-Server

Beschreibung:

int **ftp_chdir** (int ftp_stream, string Verzeichnis)

Bei Erfolg wird true, ansonsten false zurück gegeben.

Wechselt in den durch *Verzeichnis* angegebenen Datei-Ordner.

ftp_mkdir

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

ftp_mkdir -- Erzeugt ein Verzeichnis

Beschreibung:

string **ftp_mkdir** (int ftp_stream, string Verzeichnis)

Bei Erfolg wird der Name des erzeugten Ordners, ansonsten false zurück gegeben.

Erzeugt das angegebene *Verzeichnis*.

ftp_rmdir

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

ftp_rmdir -- Löscht ein Verzeichnis

Beschreibung:

int **ftp_rmdir** (int ftp_stream, string Verzeichnis)

Bei Erfolg wird true, ansonsten false zurück gegeben.

Löscht das angegebene *Verzeichnis*.

ftp_nlist

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

ftp_nlist -- Gibt eine Liste der im angegebenen Verzeichnis enthaltenen Dateien zurück

Beschreibung:

```
int ftp_nlist (int ftp_stream, string Verzeichnis)
```

Gibt ein Array aus Dateinamen oder - bei Fehler - false zurück.

[Zurück](#)

ftp_rmdir

[Anfang](#)[Hoch](#)[Vor](#)

ftp_rawlist

ftp_rawlist

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

`ftp_rawlist` -- Gibt eine detaillierte Liste der Dateien in einem angegebenen Verzeichnis zurück

Beschreibung:

`int ftp_rawlist (int ftp_stream, string Verzeichnis)`

Ftp_rawlist() führt den FTP LIST-Befehl aus und gibt das Ergebnis in einem Array zurück. Jedes Array-Element entspricht einer Textzeile. Die Ausgabe wird von PHP in keinsten Weise ausgewertet. Das System-Art-Merkmal, welches per [ftp_systype\(\)](#)-Funktion zurück gegeben wird, kann benutzt werden um die Auswertungsart der Ergebnissrückgabe zu ermitteln.

ftp_systype

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

ftp_systype -- Ermittelt den Systemtyp des entfernten FTP-Servers

Beschreibung:

int **ftp_systype** (int ftp_stream)

Gibt den entfernten Systemtyp an bzw. false bei einem Fehler.

ftp_pasv

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

ftp_pasv -- Schaltet den passiven Modus ein oder aus

Beschreibung:

int **ftp_pasv** (int ftp_stream, int pasv)

Bei Erfolg wird true, ansonsten false zurück gegeben.

Ftp_pasv() schaltet in den passiven Modus, wenn der *pasv*-Parameter true ist (ausschalten per *pasv*-Parameter gleich false). Im passiven Modus werden Datenverbindungen durch den Client statt durch den Server initiiert.

ftp_get

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

ftp_get -- Liest eine Datei des FTP-Servers und speichert sie lokal (download)

Beschreibung:

```
int ftp_get (int ftp_stream, string local_file, string remote_file,  
int mode)
```

Bei Erfolg wird true, sonst false zurück gegeben.

Ftp_get() lädt das *remote_file* vom FTP-Server und speichert es unter *local_file* auf ihren lokalen Arbeitsplatz. Der anzugebende Transfer-Parameter *mode* muss entweder FTP_ASCII oder FTP_BINARY sein.

ftp_fget

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

`ftp_fget` -- Lädt eine Datei vom FTP-Server und speichert sie in eine geöffnete, lokale Datei (download)

Beschreibung:

`int ftp_fget (int ftp_stream, int fp, string remote_file, int mode)`

Sofern erfolgreich wird `true`, ansonsten `false` zurück gegeben.

`Ftp_fget()` lädt das *remote_file* vom FTP-Server und schreibt es in die per Datei-Pointer angegebene Datei *fp*. Der Transfer-Modus-Parameter *mode* muss entweder `FTP_ASCII` oder `FTP_BINARY` sein.

ftp_put

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

ftp_put -- Überträgt eine Datei auf einen FTP-Server (upload)

Beschreibung:

```
int ftp_put (int ftp_stream, string remote_file, string local_file,  
int mode)
```

Bei Erfolg wird true, sonst false zurück gegeben.

Ftp_put() speichert die per *local_file* bestimmte Datei auf den FTP-Server als Datei namens *remote_file*. Der Übertragungs-Modus-Parameter *mode* muss entweder FTP_ASCII oder FTP_BINARY lauten.

ftp_fput

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

ftp_fput -- Überträgt eine geöffnete Datei auf einen FTP-Server (upload)

Beschreibung:

int **ftp_fput** (int ftp_stream, string remote_file, int fp, int mode)

Bei Erfolg wird true, sonst false zurück gegeben.

Ftp_fput() überträgt das per Datei-Zeiger *fp* benannte und geöffnete File zum FTP-Server. Die übertragenen Daten werden in der über *remote_file* bestimmten Datei auf dem FTP-Server gespeichert. Der *mode*-Übertragungs-Parameter muss entweder FTP_ASCII oder FTP_BINARY sein.

ftp_size

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

ftp_size -- Ermittelt die Dateigrösse einer angegebenen Datei

Beschreibung:

```
int ftp_size (int ftp_stream, string remote_file)
```

Gibt die Dateigrösse oder -1 bei Fehlern zurück.

Ftp_size() ermittelt die Grösse einer Datei. Wenn ein Fehler vorliegt, so z.B. wenn die Datei nicht existiert, wird -1 zurück gegeben. Nicht alle FTP-Server unterstützen diese Funktionalität.

ftp_mdtm

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

ftp_mdtm -- Ermittelt die letzte Änderungszeit der angegebenen Datei

Beschreibung:

```
int ftp_mdtm (int ftp_stream, string remote_file)
```

Bei Erfolg wird ein UNIX-Timestamp zurück gegeben, sonst -1.

Ftp_mdtm() prüft den Zeitpunkt der letzten Dateiänderung und liefert diesen als UNIX-Timestamp zurück. Beim Auftreten eines Fehlers wird -1 zurück gegeben. Nicht alle FTP-Server unterstützen diese Funktionalität.

ftp_rename

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

ftp_rename -- Benennt eine Datei auf dem FTP-Server um

Beschreibung:

```
int ftp_rename (int ftp_stream, string from, string to)
```

Falls erfolgreich wird true, sonst false zurück gegeben.

Ftp_rename() benennt die mit *from* angegebene Datei in *to* um.

ftp_delete

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

ftp_delete -- Löscht eine Datei auf dem FTP-Server

Beschreibung:

`int ftp_delete (int ftp_stream, string path)`

Falls erfolgreich wird true, sonst false zurück gegeben.

Ftp_delete() löscht die per *path* angegebene Datei auf dem FTP-Server.

ftp_site

(PHP3 >= 3.0.15, PHP4 >= 4.0RC1)

ftp_site -- Sendet ein SITE-Kommando zum Server

Beschreibung:

int **ftp_site** (int ftp_stream, string cmd)

Bei erfolgreicher Ausführung wird true, sonst false zurück gegeben

Ftp_site() sendet das in *cmd* stehende Kommando zum FTP-Server. SITE-Kommandos sind nicht standardisiert und von Server zu Server verschieden. Sie sind nützlich, um Sachen wie Datei-Zugriffsrechte und Gruppen-Mitgliedschaften zu handhaben.

ftp_quit

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

ftp_quit -- Schließt / beendet eine FTP-Verbindung

Beschreibung:

```
int ftp_quit (int ftp_stream)
```

Ftp_connect() schließt die per *ftp_stream* bezeichnete FTP-Sitzung.

[Zurück](#)[Anfang](#)[Vor](#)

ftp_site

[Hoch](#)

GNU Gettext

XXI. GNU Gettext

Inhaltsverzeichnis

- [bindtextdomain](#) Sets the path for a domain
 - [dcgettext](#) Overrides the domain for a single lookup
 - [dgettext](#) Override the current domain
 - [gettext](#) Lookup a message in the current domain
 - [textdomain](#) Sets the default domain
-

bindtextdomain

(PHP3 >= 3.0.7, PHP4)

bindtextdomain -- Sets the path for a domain

Description

string **bindtextdomain** (string domain, string directory)

The **bindtextdomain()** function sets the path for a domain.

dcgettext

(PHP3 >= 3.0.7, PHP4)

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.

dgettext

(PHP3 >= 3.0.7, PHP4)

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.

gettext

(PHP3 >= 3.0.7, PHP4)

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 an underscore character as an alias to this function.

Beispiel 1 Gettext()-check

```
<?php
// Set language to German
putenv ("LANG=de");

// Specify location of translation tables
bindtextdomain ("myPHPApp", "./locale");

// Choose domain
textdomain ("myPHPApp");

// Print a test message
print (gettext ("Welcome to My PHP Application"));
?>
```

textdomain

(PHP3 >= 3.0.7, PHP4)

textdomain -- Sets the default domain

Description

int **textdomain** ([string library])

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 no parameters to get the current setting without changing it.

XXII. HTTP-Funktionen

Diese Funktionen ermöglichen die Manipulation der an den Browser geschickten Informationen bis hinunter auf HTTP-Protokoll-Ebene.

Inhaltsverzeichnis

[header](#) Sendet einen HTTP-Header

[setcookie](#) Sendet ein Cookie

header

(PHP3 , PHP4)

header -- Sendet einen HTTP-Header

Beschreibung:

`int header (string string)`

Die **Header()**-Funktion wird zu Beginn einer HTML-Datei benutzt um HTTP-Anfangsinformationen zu senden. Siehe [HTTP 1.1 Spezifikationen](#) für weitere Informationen. *Hinweis:* Beachten sie, dass die **Header()**-Funktion aufgerufen werden muss, bevor irgendeine Ausgabe durch normale HTML-Tags oder PHP erzeugt wird. Es ist ein häufig anzutreffender Fehler, per [include\(\)](#) Programm-Code einzubinden (bzw. per `auto_prepend`) oder Leerzeichen oder -zeilen vor dem Code stehen zu haben, die bereits Ausgaben bewirken (auch wenn diese leer sind) bevor **header()** aufgerufen wird.

Es gibt zwei Spezialfälle von Header-Aufrufen. Der Erste ist der "Location"-Header. Es wird nicht nur der Header an den Browser geschickt, der Apache erhält auch noch den Status-Code REDIRECT. Aus Sicht des Skript-Schreibers ist das nicht wichtig, aber für alle, die die Internas von Apache verstehen wollen, ist dies von Belang.

```
header ("Location: http://www.php.net"); /* Umleitung des Browsers
                                          zur PHP-Web-Seite. */
exit; /* Sicher stellen, das nicht trotz Umleitung nachfolgender
      Code ausgeführt wird. */
```

Der zweite Spezialfall ist ein Header der mit der Zeichenfolge "HTTP/" beginnt (Groß-/Kleinschreibung ist egal). Falls sie z.B. die Dokument-Fehler-Anweisung 404 des Apache auf ein PHP-Skript zeigen lassen, ist es gut, wenn dieses Skript einen 404-Fehler erzeugt. In diesem Skript sollten sie also zuerst stehen haben:

```
header ("HTTP/1.0 404 Not Found");
```

PHP-Skripte erzeugen oft dynamisches HTML, das weder vom Browser noch von irgendeinem Proxy zwischen Web-Server und Client-Browser gepuffert ("gecached") werden soll bzw. darf. Bei vielen Proxies und Browsern kann das Cachen unterbunden werden und zwar mit:


```
header ("Expires: Mon, 26 Jul 1997 05:00:00 GMT");    // Datum der
Vergangenheit
header ("Last-Modified: " . gmdate ("D, d M Y H:i:s") . " GMT");
// immer geändert
header ("Cache-Control: no-cache, must-revalidate"); // HTTP/1.1
header ("Pragma: no-cache");                        // HTTP/1.0
```

[Zurück](#)

HTTP-Funktionen

[Anfang](#)[Hoch](#)[Vor](#)

setcookie

setcookie

(PHP3 , PHP4)

setcookie -- Sendet ein Cookie

Beschreibung:

`int setcookie (string name, string value, int expire, string path, string domain, int secure)`

Setcookie() definiert ein mit den Header-Informationen zu übertragendes Cookie. Cookies müssen *vor* irgendwelchen anderen Headern gesendet werden (dies ist eine Einschränkung der Cookies, nicht von PHP). Sie müssen deshalb solche Funktionsaufrufe vor irgendwelchen `<html>`- oder `<head>`-Tags im Skript einsetzen.

Alle Angaben ausser dem *name*-Argument können angegeben werden. Ist nur der Name angegeben wird das Cookie mit diesem Namen auf dem Client gelöscht. Sie können beliebige Argumente auch durch einen Leerstring ("") ersetzen, um diese Argument zu übergehen. Der *expire*-Parameter und das *secure*-Argument sind Integer-Werte und können nicht mit einem Leerstring übersprungen werden. Setzen sie statt dessen Null (0). Der *expire*-Parameter ist ein normaler UNIX-Zeitwert als Integer-Zahl, wie er von der [time\(\)](#) oder [mktime\(\)](#)-Funktion zurück gegeben wird. Das *secure*-Argument bedeutet, dass das Cookie nur über eine sichere HTTP-Verbindung geschickt werden soll.

Anwendungsbereiche:

Cookies werden nicht sichtbar, bevor nicht eine Seite geladen wird, in der das Cookie sichtbar sein soll.

In PHP3 werden mehrfache Aufrufe von **setcookie()** im selben Skript in umgekehrter Reihenfolge abgearbeitet. Sollten sie also ein Cookie löschen wollen bevor sie ein anderes setzen, sollten sie das Setzen vor dem Löschen vornehmen. In PHP4 werden mehrfache Aufrufe von **setcookie()** in der Reihenfolge ihres Aufrufs behandelt.

Einige Beispiele:

Beispiel 1 Setcookie()-Beispiele:

```
setcookie ("TestCookie", "Test Value");
setcookie ("TestCookie", $value, time()+3600); /* verfällt in 1 Stunde */
setcookie ("TestCookie", $value, time()+3600, "/~rasmus/", ".utoronto.ca",1);
```

Beachten sie, dass der Wertebereich des Cookies automatisch URL-konform codiert (urlencoded) wird, sobald sie das Cookie senden, und wenn es gelesen wird, wird es automatisch URL-konform decodiert und einer Variablen zugewiesen, die den selben Namen wie das Cookie trägt. Um die Inhalte unserer Test-Cookies zu sehen, brauchen sie nur eines der folgenden Beispiel zu benutzen:

```
echo $TestCookie;  
echo $HTTP_COOKIE_VARS["TestCookie"];
```

Sie können auch ein Array an Cookies setzen, in dem sie die Array-Schreibweise benutzen. Dadurch werden so viele Cookies gesetzt, wie ihr Array Elemente hat. Sobald das Cookie aber von ihrem Skript gelesen wird, werden alle Werte in ein einziges Array mit dem Cookie-Namen eingelesen:

```
setcookie ("cookie[three]", "cookiethree");  
setcookie ("cookie[two]", "cookietwo");  
setcookie ("cookie[one]", "cookieone");  
if (isset ($cookie)) {  
    while (list ($name, $value) = each ($cookie)) {  
        echo "$name == $value<br>\n";  
    }  
}
```

Weitere Informationen zu Cookies und deren Spezifikationen erhalten sie bei Netscape unter http://www.netscape.com/newsref/std/cookie_spec.html.

Microsofts Internet-Explorer 4 mit Service-Pack 1 geht nicht korrekt mit Cookies um, die den Pfad-Parameter beinhalten.

Netscape Communicator 4.05 und Microsoft Internet Explorer 3.x scheinen mit Cookies Probleme zu haben, wenn die Argumente für Pfad und Zeit nicht angegeben sind.

[Zurück](#)

header

[Anfang](#)[Hoch](#)[Vor](#)

Hyperwave functions

XXIII. 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 (If I remember properly it was in 1996).

Hyperwave is not free software. The current version, 4.1, is available at <http://www.hyperwave.com/>. A time limited version can be ordered for free (30 days).

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 possible 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/php3_script/my_object'` and the script 'php3_script' evaluates the \$PATH_INFO variable and retrieves the object with name 'my_object' from the Hyperwave server. There 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 inserted 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 it 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. '_'. to be continued.

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_object2array()**. Several functions return object records. The names of those functions end with obj.
- **object array:** An associated array with all attributes of an object. The key is the attribute name. If an attribute occurs more than once in an object record it will result in another indexed or associated array. Attributes which are language depended (like the title, keyword, description) will form an associated array with the key set to the language abbreviation. 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 optimised for HTML documents but may be used for any format.

Several functions which return an array of object records do also return an associated 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.

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 I will assume that the Apache server will only serve as a Hyperwave web interface. 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 `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 then 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. Finally, don't forget to turn on the rewriting engine with

```
RewriteEngine on
```

My experiences have shown that you will need the following 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.

Todo

There are still some things todo:

- 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 from object record into object array needs to handle any multiple attribute.

Inhaltsverzeichnis

[hw_Array2Objrec](#) convert attributes from object array to object record

[hw_Children](#) object ids of children

[hw_ChildrenObj](#) object records of children

[hw_Close](#) closes the Hyperwave connection

[hw_Connect](#) opens a connection

[hw_Cp](#) copies objects

[hw_Deleteobject](#) deletes object

[hw_DocByAnchor](#) object id object belonging to anchor

[hw_DocByAnchorObj](#) object record object belonging to anchor

[hw_DocumentAttributes](#) object record of `hw_document`

[hw_DocumentBodyTag](#) body tag of `hw_document`

[hw_DocumentContent](#) returns content of `hw_document`

[hw_DocumentSetContent](#) sets/replaces content of `hw_document`

[hw_DocumentSize](#) size of `hw_document`

[hw_ErrorMsg](#) returns error message

[hw_EditText](#) retrieve text document

[hw_Error](#) error number

[hw_Free_Document](#) frees `hw_document`

[hw_GetParents](#) object ids of parents

[hw_GetParentsObj](#) object records of parents

[hw_GetChildColl](#) object ids of child collections

[hw_GetChildCollObj](#) object records of child collections

[hw_GetRemote](#) Gets a remote document

[hw_GetRemoteChildren](#) Gets children of remote document

[hw_GetSrcByDestObj](#) Returns anchors pointing at object

[hw_GetObject](#) object record

[hw_GetAndLock](#) return bject record and lock object

[hw_GetText](#) retrieve text document

[hw_GetObjectByQuery](#) search object

[hw_GetObjectByQueryObj](#) search object

[hw_GetObjectByQueryColl](#) search object in collection

[hw_GetObjectByQueryCollObj](#) search object in collection

[hw_GetChildDocColl](#) object ids of child documents of collection

[hw_GetChildDocCollObj](#) object records of child documents of collection

[hw_GetAnchors](#) object ids of anchors of document

[hw_GetAnchorsObj](#) object records of anchors of document

[hw_Mv](#) moves objects

[hw_Identify](#) identifies as user

[hw_InCollections](#) check if object ids in collections

[hw_Info](#) info about connection

[hw_InsColl](#) insert collection

[hw_InsDoc](#) insert document

[hw_InsertDocument](#) upload any document

[hw_InsertObject](#) inserts an object record

[hw_mapid](#) Maps global id on virtual local id

[hw_Modifyobject](#) modifies object record

[hw_New_Document](#) create new document

[hw_Objrec2Array](#) convert attributes from object record to object array

[hw_OutputDocument](#) prints hw_document

[hw_pConnect](#) make a persistent database connection

[hw_PipeDocument](#) retrieve any document

[hw_Root](#) root object id

[hw_Unlock](#) unlock object

[hw_Who](#) List of currently logged in users

[hw Username](#) name of currently logged in user

hw_Array2Objrec

(unknown)

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_Children

(unknown)

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

(unknown)

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

(unknown)

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

(unknown)

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_Cp

(unknown)

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

(unknown)

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

(unknown)

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

(unknown)

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_DocumentAttributes

(unknown)

hw_DocumentAttributes -- object record of hw_document

Description

string **hw_documentattributes** (int hw_document)

Returns the object record of the document.

See also **hw_DocumentBodyTag()**, **hw_DocumentSize()**.

hw_DocumentBodyTag

(unknown)

hw_DocumentBodyTag -- body tag of hw_document

Description

string **hw_documentbodytag** (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_DocumentAttributes()**, **hw_DocumentSize()**.

hw_DocumentContent

(unknown)

hw_DocumentContent -- returns content of hw_document

Description

string **hw_documentcontent** (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_DocumentAttributes()**, **hw_DocumentSize()**, **hw_DocumentSetContent()**.

hw_DocumentSetContent

(unknown)

hw_DocumentSetContent -- sets/replaces content of hw_document

Description

string **hw_documentsetcontent** (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_DocumentAttributes()**, **hw_DocumentSize()**, **hw_DocumentContent()**.

hw_DocumentSize

(unknown)

hw_DocumentSize -- size of hw_document

Description

```
int hw_documentsize (int hw_document)
```

Returns the size in bytes of the document.

See also [hw_DocumentBodyTag\(\)](#), [hw_DocumentAttributes\(\)](#).

hw_ErrorMsg

(unknown)

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_EditText

(unknown)

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_FreeDocument\(\)](#), [hw_DocumentBodyTag\(\)](#), [hw_DocumentSize\(\)](#), [hw_OutputDocument\(\)](#), [hw_GetText\(\)](#).

hw_Error

(unknown)

hw_Error -- error number

Description

```
int hw_error (int connection)
```

Returns the last error number. If the return value is 0 no error has occurred. The error relates to the last command.

[Zurück](#)[Anfang](#)[Vor](#)

hw_EditText

[Hoch](#)

hw_Free_Document

hw_Free_Document

(unknown)

hw_Free_Document -- frees hw_document

Description

```
int hw_free_document (int hw_document)
```

Frees the memory occupied by the Hyperwave document.

hw_GetParents

(unknown)

hw_GetParents -- object ids of parents

Description

array **hw_getparentsobj** (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

(unknown)

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_GetChildColl

(unknown)

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_GetChildren()**, **hw_GetChildDocColl()**.

hw_GetChildCollObj

(unknown)

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()**, **hw_GetChildDocCollObj()**.

hw_GetRemote

(unknown)

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 through 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

(unknown)

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 formatted 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

(unknown)

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_GetObject

(unknown)

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()**, **hw_GetObjectByQuery()**.

hw_GetAndLock

(unknown)

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()**, **hw_GetObject()**.

hw_GetText

(unknown)

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 . The actual location of the source and destination object in the document hierarchy 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. In 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 separator between hierarchy levels on the Hyperwave server and the source document is located at 'a-b-d-source' the resulting HTML link would be: . This is useful if you want to download the whole server content onto disk and map the document hierarchy 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_FreeDocument\(\)](#), [hw_DocumentBodyTag\(\)](#), [hw_DocumentSize\(\)](#), [hw_OutputDocument\(\)](#).

hw_GetObjectByQuery

(unknown)

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_GetObjectByQueryObj

(unknown)

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_GetObjectByQueryColl

(unknown)

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

(unknown)

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_GetChildDocColl

(unknown)

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_GetChildren()**, **hw_GetChildColl()**.

hw_GetChildDocCollObj

(unknown)

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()**, **hw_GetChildCollObj()**.

hw_GetAnchors

(unknown)

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

(unknown)

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_Mv

(unknown)

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 0 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\(\)](#), [hw_deleteobject\(\)](#).

hw_Identify

(unknown)

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 think 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

(unknown)

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

(unknown)

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

(unknown)

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

(unknown)

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()**, **hw_InsColl()**.

hw_InsertDocument

(unknown)

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_NewDocument()**. 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

(unknown)

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()`, `hw_InsColl()`.

hw_mapid

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

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

(unknown)

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 recursive modification. If some of the objects cannot be modified they will be skipped 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 necessary if you want 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()**.

Beispiel 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.

Beispiel 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);
```

Anmerkung: 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:

Beispiel 3 modifying Title attribute

```
$remarr = array("Title" => "en:Books");
$addarr = array("Title" => "en:Articles");
$hw_modifyobject($connect, $objid, $remarr, $addarr);
```

or

Beispiel 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'.

Beispiel 5 removing attribute

```
$remarr = array("Title" => "");
$addarr = array("Title" => "en:Articles");
$hw_modifyobject($connect, $objid, $remarr, $addarr);
```

Anmerkung: 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.

Anmerkung: If you need to delete all attributes with a certain name you will have to pass an empty string as the attribute value.

Anmerkung: 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.

Anmerkung: 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.

Anmerkung: You may not surround this function by calls to [hw_getandlock\(\)](#) and [hw_unlock\(\)](#). [hw_modifyobject\(\)](#) does this internally.

Returns TRUE if no error occurs otherwise FALSE.

[Zurück](#)

hw_mapid

[Anfang](#)

[Hoch](#)

[Vor](#)

hw_New_Document

hw_New_Document

(unknown)

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_FreeDocument()`, `hw_DocumentSize()`, `hw_DocumentBodyTag()`, `hw_OutputDocument()`, `hw_InsertDocument()`.

hw_Objrec2Array

(unknown)

hw_Objrec2Array -- convert attributes from object record to object array

Description

array **hw_objrec2array** (string object_record)

Converts an *object_record* into an object array. The keys of the resulting array are the attributes names. Multiple 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. Currently only the attributes 'Title', 'Description' and 'Keyword' are treated properly. Other multiple attributes form an index array. Currently only the attribute 'Group' is handled properly.

See also [hw_array2objrec\(\)](#).

hw_OutputDocument

(unknown)

hw_OutputDocument -- prints hw_document

Description

```
int hw_outputdocument (int hw_document)
```

Prints the document without the BODY tag.

hw_pConnect

(unknown)

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

(unknown)

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_FreeDocument()**, **hw_DocumentSize()**, **hw_DocumentBodyTag()**, **hw_OutputDocument()**.

hw_Root

(unknown)

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_Unlock

(unknown)

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

(unknown)

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 initiated the request.

hw_Username

(unknown)

hw_Username -- name of currently logged in user

Description

string **hw_getusername** (int connection)

Returns the username of the connection.

XXIV. ICAP Functions

To get these functions to work, you have to compile PHP with `--with-icap`. That requires the icap library to be installed. Grab the latest version from <http://icap.chek.com/> and compile and install it.

Inhaltsverzeichnis

[icap_open](#) Opens up an ICAP connection

[icap_close](#) Close an ICAP stream

[icap_fetch_event](#) Fetches an event from the calendar stream/

[icap_list_events](#) Return a list of events between two given datetimes

[icap_store_event](#) Store an event into an ICAP calendar

[icap_delete_event](#) Delete an event from an ICAP calendar

[icap_snooze](#) turn off an alarm for an event

[icap_list_alarms](#) Return a list of events that has an alarm triggered at the given datetime

icap_open

(PHP4 >= 4.0b4)

icap_open -- Opens up an ICAP connection

Description

stream **icap_open** (stringcalendar, stringusername, stringpassword, stringoptions)

Returns an ICAP stream on success, false on error.

icap_open() opens up an ICAP connection to the specified *calendar* store. If the optional *options* is specified, passes the *options* to that mailbox also.

icap_close

(unknown)

icap_close -- Close an ICAP stream

Description

```
int icap_close (int icap_stream, int flags)
```

Closes the given icap stream.

[Zurück](#)[Anfang](#)[Vor](#)

icap_open

[Hoch](#)

icap_fetch_event

icap_fetch_event

(PHP4 >= 4.0b4)

icap_fetch_event -- Fetches an event from the calendar stream/

Description

object**icap_fetch_event** (streamicap_stream, idevent id, optionsoptions)

icap_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.

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

icap_list_events

(PHP4 >= 4.0RC1)

icap_list_events -- Return a list of events between two given datetimes

Description

array **icap_list_events** (stream icap_stream, datetime begin_date, datetime end_date)

Returns an array of event ID's that are between the two given datetimes.

icap_list_events() function takes in a beginning datetime and an end datetime for a calendar stream. An array of event id's that are between the given datetimes are returned.

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

icap_store_event

(PHP4 >= 4.0b4)

icap_store_event -- Store an event into an ICAP calendar

Description

int icap_store_event (int icap_stream, object event)

icap_store_event() Stores an event into an ICAP calendar. an event object consists of:

- int public - 1 if public, 0 if private;
- string caegory - 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 sned out an alarm.
- datetime start - datetime object of the start of the event.
- datetime end - datetime object of the end of the event.

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

Returns true on success and false on error.

icap_delete_event

(PHP4 >= 4.0b4)

icap_delete_event -- Delete an event from an ICAP calendar

Description

int **icap_delete_event** (int uid)

icap_delete_event() deletes the calendar event specified by the *uid*.

Returns true.

icap_snooze

(PHP4 >= 4.0b4)

icap_snooze -- turn off an alarm for an event

Description

```
int icap_snooze (int uid)
```

icap_snooze() turns off an alarm for a calendar event specified by the *uid*.

Returns true.

icap_list_alarms

(PHP4 >= 4.0b4)

icap_list_alarms -- Return a list of events that has an alarm triggered at the given datetime

Description

array **icap_list_alarms** (stream icap_stream, datetime alarm_date)

Returns an array of event ID's that has an alarm going off at the given datetime.

icap_list_alarms() function takes in a datetime for a calendar stream. An array of event id's that has an alarm should be going off at the datetime are returned.

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

XXV. Grafik-Funktionen

Sie können die Grafik-Funktionen von PHP nicht nur benutzen, um die Grösse von JPEG, GIF, PNG und SWF-Bild-Dateien zu ermitteln, sondern auch - sofern sie die GD-Bibliothek (verfügbar unter <http://www.boutell.com/gd/>) eingebunden haben - Grafiken bzw. Bilder dynamisch (also zur Laufzeit ihres Skripts) erzeugen bzw. verändern.

Inhaltsverzeichnis

[getImageSize](#) Ermittelt die Ausmaße einer GIF-, JPEG-, PNG- oder SWF-Grafik-Datei

[imageArc](#) Zeichnen einer Teil-Ellipse

[imageChar](#) Stellt ein Zeichen mit horizontaler Ausrichtung dar

[imageCharUp](#) Zeichnet einen vertikal ausgerichteten Charakter

[imageColorAllocate](#) Bestimmt die Farbe einer Grafik

[imageColorAt](#) Ermittelt den Farbwert eines Bildpunktes

[imageColorClosest](#) Ermittelt den Farbwert-Index, der den angegebenen Farben am nächsten liegt

[imageColorDeAllocate](#) Löscht eine Farbdefinition

[imageColorExact](#) Ermittelt den Index-Wert der angegebenen Farbe

[imageColorResolve](#) Ermittelt den Index-Wert der angegebenen Farbe oder die nächst mögliche Alternative dazu

[imageColorSet](#) Setzt die Farbe für den angegebenen Paletten-Index

[imageColorsForIndex](#) Ermittelt die Farbwerte einer angegebenen Farb-Palette

[imageColorsTotal](#) Ermittelt die Anzahl der definierten Farben eines Bildes

[imageColorTransparent](#) Definiert eine Farbe als transparent

[imageCopy](#) Kopiert einen Bildausschnitt

[imageCopyResized](#) Kopieren und Ändern der Grösse eines Bild-Teiles

[imageCreate](#) Erzeugt ein neues Bild

[imageCreateFromGIF](#) Erzeugt ein neues Bild im GIF-Format, welches aus einer Datei oder von einer URL gelesen wird

[imageCreateFromJPEG](#) Erzeugt ein neues Bild im JPEG-Format, welches aus einer Datei oder von einer URL gelesen wird

[imageCreateFromPNG](#) Erzeugt ein neues Bild im PNG-Format, welches aus einer Datei oder von einer URL gelesen wird

[imageDashedLine](#) Zeichnen einer gestrichelten Linie

[ImageDestroy](#) Löscht ein Bild

[ImageFill](#) Füllen mit Farbe ("flood fill")

[ImageFilledPolygon](#) Zeichnet ein gefülltes Vieleck (Polygon)

[ImageFilledRectangle](#) Zeichnet ein gefülltes Rechteck

[ImageFillToBorder](#) Flächen-Farbfüllung ("flood fill") mit einer angegebenen Farbe

[ImageFontHeight](#) Ermittelt die Font-Höhe

[ImageFontWidth](#) Ermittelt die Font-Breite

[ImageGammaCorrect](#) Anwendung einer Gamma-Korrektur auf ein GD-Bild

[ImageGIF](#) Ausgabe eines Bildes an den Browser oder in eine Datei

[ImageJPEG](#) Ausgabe des Bildes im Browser oder als Datei

[ImagePNG](#) Ausgabe eines Bildes an den Browser oder in eine Datei

[ImageInterlace](#) Schaltet die Interlaced-Darstellung eines Bildes an oder aus

[ImageLine](#) Zeichnen einer Linie

[ImageLoadFont](#) Lädt einen neuen Font

[ImagePolygon](#) Zeichnen eines Vielecks (Polygon)

[ImagePSBBox](#) Ermittelt die Ausmaße des Rechtecks, das für die Ausgabe eines Textes unter Verwendung eines PostScript-Fonts (Typ 1) notwendig ist

[ImagePSEncodeFont](#) Ändert die Vektor-Beschreibung eines Fonts

[ImagePsExtendFont](#) Vergrößert oder komprimiert einen Font

[ImagePSFreeFont](#) Gibt den durch einen Typ 1 PostScript-Font belegten Speicher wieder frei

[ImagePSLoadFont](#) Lädt einen Typ 1 PostScript-Font aus einer Datei

[ImagePsSlantFont](#) Setzt einen Font schräg

[ImagePSText](#) Ausgabe eines Textes auf einem Bild unter Verwendung von Typ 1 PostScript-Fonts

[ImageRectangle](#) Zeichnet ein Rechteck

[ImageSetPixel](#) Setzt ein einzelnes Pixel

[ImageString](#) Zeichnet einen horizontalen String

[ImageStringUp](#) Zeichnet einen vertikalen String

[ImageSX](#) Ermittelt die Bild-Breite

[ImageSY](#) Ermittelt die Bild-Höhe

[ImageTTFBBox](#) Ermittelt die Rahmenmaße für die Ausgabe eines Textes im True-Type-Format

[ImageTTFText](#) Erzeugt TTF-Text im Bild

[ImageTypes](#) Gibt die von der aktuell verwendeten PHP-Version unterstützten Grafik-Formate zurück

GetImageSize

(unknown)

GetImageSize -- Ermittelt die Ausmaße einer GIF-, JPEG-, PNG- oder SWF-Grafik-Datei

Beschreibung:

array **getimagesize** (string filename [, array imageinfo])

Mit **GetImageSize()** ermitteln sie die Daten jeder GIF-, JPG-, PNG oder SWF-Datei. Es werden die Ausmaße in Abhängigkeit vom Grafik-Typ sowie die Höhe und Breite als *ein* String zurückgegeben. Dieser kann innerhalb eines regulären HTML IMG-Tags verwendet werden.

Es wird ein Array aus 4 Elementen zurück gegeben. Im Array-Index 0 steht die Breite. Index 1 enthält die Höhe, 2 ein Flag je nach Grafik-Typ (1 = GIF, 2 = JPG, 3 = PNG, SWF = 4) und Index 3 die richtige Zeichenkette im Format "height=xxx width=xxx" zur Verwendung im IMG-Tag von HTML.

Beispiel 1 Grafik-Ausmaße ermitteln per GetImageSize:

```
<?php $size = GetImageSize ("img/flag.jpg"); ?>
<IMG SRC="img/flag.jpg" <?php echo $size[3]; ?>
```

Mit dem optionalen *imageinfo*-Parameter können sie weitere Informationen der Grafik-Datei auslesen. Z.Zt. unterstützt dies die verschiedenen JPG APP-Markierungen per assoziativem Array. Einige Programme nutzen diese Markierungen, um Text in diese Grafiken einzubinden. Gebräuchlich ist die Einbettung von IPTC <http://www.xe.net/iptc/> Informationen in die APP13-Marke. Sie können [iptcparse\(\)](#) benutzen, um die binäre APP13-Marke in ein lesbares Format zu übertragen.

Beispiel 2 GetImageSize mit IPTC-Informationen

```
<?php
    $size = GetImageSize ("testimg.jpg",&$info);
    if (isset ($info["APP13"])) {
        $iptc = iptcparse ($info["APP13"]);
        var_dump ($iptc);
    }
?>
```

Anmerkung: Dieser Befehl funktioniert auch ohne GD-Bibliothek.

[Zurück](#)

Grafik-Funktionen

[Anfang](#)

[Hoch](#)

[Vor](#)

ImageArc

ImageArc

(unknown)

ImageArc -- Zeichnen einer Teil-Ellipse

Beschreibung:

`int imagearc (int im, int cx, int cy, int w, int h, int s, int e, int col)`

ImageArc() zeichnet den Teil einer Ellipse mit Mittelpunkt bei *cx* und *cy* (oben links ist 0, 0) in dem durch den *im*-Handle definiertem Bild. *w* und *h* benennen die Breite und Höhe der Ellipse, *s* und *e* den Start- bzw. Endwinkel des Bogens in Grad. Mit *col* bestimmen sie die Farbe der Ellipse.

ImageChar

(unknown)

ImageChar -- Stellt ein Zeichen mit horizontaler Ausrichtung dar

Beschreibung:

`int imagechar (int im, int font, int x, int y, string c, int col)`

ImageChar() zeichnet das erste Zeichen (Charakter) des Strings *c* in das Bild, auf das per *id* gezeigt wird. Die linke obere Ecke des Zeichens wird durch *x* / *y* bestimmt. Dabei wird von den Koordinaten 0 / 0 als Zeichnungsanfang links-oben ausgegangen. Die Farbe des Zeichens wird mit *col* bestimmt. Ist *font* als 1, 2, 3, 4 oder 5 definiert, wird ein eingebauter Font genutzt (je grösser die Nummer, desto grösser der Font).

Beachten sie auch [imageloadfont\(\)](#).

ImageCharUp

(unknown)

ImageCharUp -- Zeichnet einen vertikal ausgerichteten Charakter

Beschreibung:

`int imagecharup (int im, int font, int x, int y, string c, int col)`

ImageCharUp() zeichnet das erste Zeichen (Charakter) des Strings *c* in das Bild, auf das per *id* gezeigt wird. Die Koordinaten des Zeichens werden durch *x* / *y* bestimmt. Dabei wird von den Koordinaten 0 / 0 als Zeichnungsanfang links-oben ausgegangen. Die Farbe des Zeichens wird mit *col* bestimmt. Ist *font* als 1, 2, 3, 4 oder 5 definiert, wird ein eingebauter Font genutzt (je grösser die Nummer, desto grösser der Font).

Beachten sie auch [imageloadfont\(\)](#).

ImageColorAllocate

(unknown)

ImageColorAllocate -- Bestimmt die Farbe einer Grafik

Beschreibung:

`int imagecolorallocate (int im, int red, int green, int blue)`

ImageColorAllocate() gibt eine Farb-ID, die durch die angegebenen RGB-Werte bestimmt wird, zurück. Der Parameter *im* wird ihnen von der [imagecreate\(\)](#)-Funktion zurück gegeben. **ImageColorAllocate()** muss für jede Farbe aufgerufen werden, die innerhalb der Grafik *im* verwendet werden soll.

```
$weiss    = ImageColorAllocate ($im, 255, 255, 255);  
$schwarz = ImageColorAllocate ($im, 0, 0, 0);
```

ImageColorAt

(unknown)

ImageColorAt -- Ermittelt den Farbwert eines Bildpunktes

Beschreibung:

`int imagecolorat (int im, int x, int y)`

Ermittelt den Farb-Wert eines Pixels an den Koordinaten x / y der mit *im* bestimmten Grafik.

Beachten sie auch [imagecolorset\(\)](#) und [imagecolorsforindex\(\)](#).

ImageColorClosest

(unknown)

ImageColorClosest -- Ermittelt den Farbwert-Index, der den angegebenen Farben am nächsten liegt

Beschreibung:

`int imagecolorclosest (int im, int red, int green, int blue)`

Gibt den Index der Farbwert-Palette des Bildes *im* zurück, der den angegebenen RGB-Werten am nächsten kommt.

Die "Entfernung" zwischen der angeforderten Farbe und jedem Farb-Paletten-Wert wird berechnet, als ob die angegebenen RGB-Werte Punkte in einem dreidimensionalen Raum darstellen.

Siehe auch [imagecolorexact\(\)](#).

ImageColorDeAllocate

(unknown)

ImageColorDeAllocate -- Löscht eine Farbdefinition

Beschreibung:

int **imagecolordeallocate** (int im, int index)

Die Funktion **ImageColorDeAllocate()** löscht eine vorher über **ImageColorAllocate()** ermittelte Farbe.

```
$weiss = ImageColorAllocate($im, 255, 255, 255);  
ImageColorDeAllocate($im, $weiss);
```


ImageColorExact

(unknown)

ImageColorExact -- Ermittelt den Index-Wert der angegebenen Farbe

Beschreibung:

`int imagecolorexact (int im, int red, int green, int blue)`

Gibt den Index der angegebenen Farbe in der Farb-Palette des Bildes *im* zurück.

Existiert die Farbe in der Bild-Farb-Palette nicht, wird -1 zurück gegeben.

Siehe auch [imagecolorclosest\(\)](#).

ImageColorResolve

(unknown)

ImageColorResolve -- Ermittelt den Index-Wert der angegebenen Farbe oder die nächst mögliche Alternative dazu

Beschreibung:

`int imagecolorresolve (int im, int red, int green, int blue)`

Diese Funktion ermittelt garantiert einen Farb-Index zu der angegebenen Farbe, nämlich entweder den exakten oder den nächst möglichen Wert.

Siehe auch [imagecolorclosest\(\)](#) und [imagecolorexact\(\)](#).

ImageColorSet

(unknown)

ImageColorSet -- Setzt die Farbe für den angegebenen Paletten-Index

Beschreibung:

`bool imagecolorset (int im, int index, int red, int green, int blue)`

Hierdurch setzen sie den Farbwert der Palette auf die angegebene Farbe. Dies ist nützlich bei Aktionen wie "Fülle Farbe X mit Farbe Y" (Flood-Fill-Effekt) ohne den sonst notwendigen Aufwand für solche Aktionen betreiben zu müssen.

Siehe auch [imagecolorat\(\)](#).

ImageColorsForIndex

(unknown)

ImageColorsForIndex -- Ermittelt die Farbwerte einer angegebenen Farb-Palette

Beschreibung:

array **imagecolorsforindex** (int *im*, int *index*)

Gibt ein assoziatives Array mit Rot-, Grün- und Blau-Werten zurück, die die Werte des Farb-Paletten-Index von *im* darstellen.

Siehe auch [imagecolorat\(\)](#) und [imagecolorexact\(\)](#).

ImageColorsTotal

(unknown)

ImageColorsTotal -- Ermittelt die Anzahl der definierten Farben eines Bildes

Beschreibung:

```
int imagecolorstotal (int im)
```

Ergibt die Anzahl der Farben der Palette des angegebenen Bildes *im*.

Siehe auch [imagecolorat\(\)](#) und [imagecolorsforindex\(\)](#).

ImageColorTransparent

(unknown)

ImageColorTransparent -- Definiert eine Farbe als transparent

Beschreibung:

`int imagecolortransparent (int im [, int col])`

ImageColorTransparent() setzt die transparente Farbe des Bildes *im* auf die Farbe *col*. *im* ist der Bild-Pointer, der von **ImageCreate()** zurück gegeben wird und *col* ist ein Zeiger auf die Farbe, die von **ImageColorAllocate()** ermittelt wird.

Es wird ein Zeiger auf die neue - jetzt transparente - Farbe (oder die aktuelle, falls keine Farbe angegeben wurde) zurück gegeben.

ImageCopy

(unknown)

ImageCopy -- Kopiert einen Bildausschnitt

Beschreibung:

```
int ImageCopy (int dst_im, int src_im, int dst_x, int dst_y, int  
src_x, int src_y, int src_w, int src_h)
```

Kopiert einen Teil des Quellbildes *src_im* nach *dst_im*. Der zu kopierende Teil wird angegeben über die x/y-Koordinaten *src_x* und *src_y*. Die Breite wird in *src_w* und die Höhe in *src_h* angegeben. Der Ausschnitt wird an die x/y-Koordinaten *dst_x* und *dst_y* kopiert.

ImageCopyResized

(unknown)

ImageCopyResized -- Kopieren und Ändern der Grösse eines Bild-Teiles

Beschreibung:

```
int imagecopyresized (int dst_im, int src_im, int dstX, int dstY, int srcX, int srcY, int dstW, int dstH, int srcW, int srcH)
```

ImageCopyResized() kopiert einen rechteckigen Ausschnitt eines Bildes in ein anderes. *Dst_im* gibt das Zielbild, *src_im* die Bildquelle an. Unterscheiden sich die Angaben der Quelle und des Ziels bezogen auf die Koordinaten, Höhe oder Breite, wird das Teilbild entsprechend gedehnt oder geschrumpft. Die Koordinaten beziehen sich auf die linke obere Ecke. Sie können hiermit auch Teilbilder innerhalb ein- und desselben Bildes kopieren, sofern *dst_im* und *src_im* gleich sind. Falls sich dabei aber die kopierten Teile überlappen, führt das zu unvorhersehbaren Ergebnissen.

ImageCreate

(unknown)

ImageCreate -- Erzeugt ein neues Bild

Beschreibung:

`int imagecreate (int x_size, int y_size)`

ImageCreate() gibt den Zeiger auf ein neues Bild zurück. Dieser Zeiger stellt ein leeres Bild mit der Größe *x_size* und *y_size* dar.

Beispiel 1 Erzeugung eines neuen GD-Bild-Streams und Ausgabe des Bildes:

```
<?php
header (Content-type:"image/png");
$im = @ImageCreate (50, 100)
    or die ("Kann keinen neuen GD-Bild-Stream erzeugen");
$background_color = ImageColorAllocate ($im, 255, 255, 255);
$text_color = ImageColorAllocate ($im, 233, 14, 91);
ImageString ($im, 1, 5, 5, "Ein Test-String", $text_color);
ImagePNG ($im);
?>
```

ImageCreateFromGIF

(unknown)

ImageCreateFromGIF -- Erzeugt ein neues Bild im GIF-Format, welches aus einer Datei oder von einer URL gelesen wird

Beschreibung:

`int imagecreatefromgif (string filename)`

ImageCreateFromGIF() gibt den Bezeichner auf ein Bild zurück, der das aus der angegebenen Datei eingelesene Bild darstellt.

Im Fehlerfall gibt **ImageCreateFromGIF()** eine leere Zeichenkette zurück. Zudem wird eine Fehlermeldung erzeugt, die im Browser als Verbindungsabbruch dargestellt wird. Zum besseren Debuggen wird das folgende Beispiel einen GIF-Fehler erzeugen:

Beispiel 1 Beispiel, um die Handhabung eines Fehlers bei der Bilderzeugung zu sehen (Dank an vic@zysms.com):

```
function LoadGif ($imgname) {
    $im = @ImageCreateFromGIF ($imgname); /* Versuch, Datei zu öffnen */
    if (!$im) {                          /* Prüfen, ob fehlgeschlagen */
        $im = ImageCreate (150, 30);     /* Erzeugen eines leeren Bildes */
        $bgc = ImageColorAllocate ($im, 255, 255, 255);
        $tc = ImageColorAllocate ($im, 0, 0, 0);
        ImageFilledRectangle ($im, 0, 0, 150, 30, $bgc);
        /* Ausgabe einer Fehlermeldung */
        ImageString($im, 1, 5, 5, "Fehler beim Öffnen von: $imgname", $tc);
    }
    return $im;
}
```

Anmerkung: ACHTUNG: Ab Version 1.6 wurde die komplette GIF-Unterstützung aus der GD-Bibliothek entfernt. Diese Funktion ist also nur verfügbar, wenn sie eine Version vor 1.6 verwenden.

ImageCreateFromJPEG

(unknown)

ImageCreateFromJPEG -- Erzeugt ein neues Bild im JPEG-Format, welches aus einer Datei oder von einer URL gelesen wird

Beschreibung:

int imagecreatefromjpeg (string filename)

ImageCreateFromJPEG() gibt den Bezeichner auf ein Bild zurück, der das aus der angegebenen Datei eingelesene Bild darstellt.

Im Fehlerfall gibt **ImageCreateFromJPEG()** eine leere Zeichenkette zurück. Zudem wird eine Fehlermeldung erzeugt, die im Browser als Verbindungsabbruch dargestellt wird. Zum besseren Debuggen wird das folgende Beispiel einen JPEG-Fehler erzeugen:

Beispiel 1 Beispiel, um die Handhabung eines Fehlers bei der Bilderzeugung zu sehen (Dank an vic@zysys.com):

```
function LoadJpeg ($imgname) {
    $im = @ImageCreateFromJPEG ($imgname); /* Versuch, Datei zu öffnen */
    if (!$im) {                             /* Prüfen, ob fehlgeschlagen */
        $im = ImageCreate (150, 30);        /* Erzeugen eines leeren Bildes */
        $bgc = ImageColorAllocate ($im, 255, 255, 255);
        $tc  = ImageColorAllocate ($im, 0, 0, 0);
        ImageFilledRectangle ($im, 0, 0, 150, 30, $bgc);
        /* Ausgabe einer Fehlermeldung */
        ImageString($im, 1, 5, 5, "Fehler beim Öffnen von: $imgname", $tc);
    }
    return $im;
}
```

ImageCreateFromPNG

(unknown)

ImageCreateFromPNG -- Erzeugt ein neues Bild im PNG-Format, welches aus einer Datei oder von einer URL gelesen wird

Beschreibung:

int imagecreatefrompng (string filename)

ImageCreateFromPNG() gibt den Bezeichner auf ein Bild zurück, der das aus der angegebenen Datei eingelesene Bild darstellt.

Im Fehlerfall gibt **ImageCreateFromPNG()** eine leere Zeichenkette zurück. Zudem wird eine Fehlermeldung erzeugt, die im Browser als Verbindungsabbruch dargestellt wird. Zum besseren Debuggen wird das folgende Beispiel einen JPEG-Fehler erzeugen:

Beispiel 1 Beispiel, um die Handhabung eines Fehlers bei der Bilderzeugung zu sehen (Dank an vic@zysys.com):

```
function Loadpng ($imgname) {
    $im = @ImageCreateFromPNG ($imgname); /* Versuch, Datei zu öffnen */
    if (!$im) {                          /* Prüfen, ob fehlgeschlagen */
        $im = ImageCreate (150, 30);     /* Erzeugen eines leeren Bildes */
        $bgc = ImageColorAllocate ($im, 255, 255, 255);
        $tc = ImageColorAllocate ($im, 0, 0, 0);
        ImageFilledRectangle ($im, 0, 0, 150, 30, $bgc);
        /* Ausgabe einer Fehlermeldung */
        ImageString($im, 1, 5, 5, "Fehler beim Öffnen von: $imgname", $tc);
    }
    return $im;
}
```

ImageDashedLine

(unknown)

ImageDashedLine -- Zeichnen einer gestrichelten Linie

Beschreibung:

int **imagedashedline** (int *im*, int *x1*, int *y1*, int *x2*, int *y2*, int *col*)

ImageDashedLine() zeichnet eine gestrichelte Linie, beginnend am Punkt *x1*, *y1* bis zum Punkt *x2*, *y2* (oben links ist 0, 0) in das Bild image *im* mit der Farbe *col*.

Siehe auch **ImageLine()**.

ImageDestroy

(unknown)

ImageDestroy -- Löscht ein Bild

Beschreibung:

```
int imagedestroy (int im)
```

ImageDestroy() gibt den durch das Bild *im* belegten Speicher wieder frei. *Im* ist der Bezeichner, der ihnen beim Aufruf der Funktion **ImageCreate()** zurück gegeben worden ist.

ImageFill

(unknown)

ImageFill -- Füllen mit Farbe ("flood fill")

Beschreibung:

`int imagefill (int im, int x, int y, int col)`

ImageFill() bewirkt das Füllen eines Bildes beginnend bei der Koordinate x, y (oben links ist 0, 0) mit der Farbe col im Bild im .

[Zurück](#)

ImageDestroy

[Anfang](#)[Hoch](#)[Vor](#)

ImageFilledPolygon

ImageFilledPolygon

(unknown)

ImageFilledPolygon -- Zeichnet ein gefülltes Vieleck (Polygon)

Beschreibung:

`int imagefilledpolygon (int im, array points, int num_points, int col)`

ImageFilledPolygon() erzeugt ein Vieleck im Bild *im*, gefüllt mit der Farbe *col*. *Points* ist ein PHP-Array, das die Eckpunkte des Vielecks enthält. *Points*[0] ist die X-Koordinate (x0), *points*[1] die Y-Koordinate (y0) des ersten Eckpunktes. *Points*[2] ist x1, *points*[3] = y1 usw. *Num_points* enthält die Anzahl der Punkte.

[Zurück](#)[Anfang](#)[Vor](#)

ImageFill

[Hoch](#)

ImageFilledRectangle

ImageFilledRectangle

(unknown)

ImageFilledRectangle -- Zeichnet ein gefülltes Rechteck

Beschreibung:

`int imagefilledrectangle (int im, int x1, int y1, int x2, int y2, int col)`

ImageFilledRectangle() erzeugt ein mit der Farbe **col()** gefülltes Rechteck innerhalb des Bildes *im*. Die obere linke Eck-Koordinate wird mittels *x1* und *y1*, die untere rechte Eck-Koordinate mittels *x2* und *y2* definiert. 0, 0 ist dabei die linke obere Ecke des Bildes *im*.

ImageFillToBorder

(unknown)

ImageFillToBorder -- Flächen-Farbfüllung ("flood fill") mit einer angegebenen Farbe

Beschreibung:

int **imagefilltoborder** (int *im*, int *x*, int *y*, int *border*, int *col*)

ImageFillToBorder() bewirkt eine Farb-Füllung in einem Bereich, dessen Grenze durch die Farbe in *border* definiert wird. Der Startpunkt wird durch *x* und *y* angegeben (oben links ist 0, 0). Der Bereich wird gefüllt mit der Farbe *col*.

ImageFontHeight

(unknown)

ImageFontHeight -- Ermittelt die Font-Höhe

Beschreibung:

```
int imagefontheight (int font)
```

Ermittelt die Höhe eines Zeichens (in Pixel) im angegebenen Font.

Siehe auch **ImageFontWidth()** und **ImageLoadFont()**.

ImageFontWidth

(unknown)

ImageFontWidth -- Ermittelt die Font-Breite

Beschreibung:

```
int imagefontwidth (int font)
```

Gibt die Zeichenbreite des angegebenen Fonts in Pixeln zurück.

Siehe auch **ImageFontHeight()** und **ImageLoadFont()**.

ImageGammaCorrect

(unknown)

ImageGammaCorrect -- Anwendung einer Gamma-Korrektur auf ein GD-Bild

Beschreibung:

`int imagegammaconvert (int im, double inputgamma, double outputgamma)`

Die Funktion **ImageGammaCorrect()** bewirkt die Anwendung einer Gamma-Korrektur auf das mittels Bezeichner *im* angegebene GD-Bild. Anzugeben ist der Eingabe-Gammawert *inputgamma* und der Ausgabe-Gammawert *outputgamma*.

ImageGIF

(unknown)

ImageGIF -- Ausgabe eines Bildes an den Browser oder in eine Datei

Beschreibung:

```
int imagegif (int im [, string filename])
```

ImageGIF() erzeugt eine Datei im Format GIF aus *im*. Falls sie [*filename*] angegeben haben, wird die Ausgabe in diese Datei geschrieben, ansonsten wird die Ausgabe direkt an den Browser geleitet. *Im* ist das Ergebniss der [imagecreate\(\)](#)-Funktion.

Das Format wird GIF87a sein, es sei denn, sie haben das Bild per **ImageColorTransparent()** als transparent formatiert. In diesem Fall wird das Bild im GIF89a-Format ausgegeben.

Die Angabe des Dateinamens ist optional. Falls er fehlt, werden die Bilddaten direkt ausgegeben. Durch Verwendung der Funktion [header\(\)](#) mit der "content-type"-Angabe "image/gif" können sie PHP-Skripte erstellen, welche die GIF-Ausgabe direkt vornehmen.

Anmerkung: ACHTUNG: Ab Version 1.6 wurde die GIF-Unterstützung komplett aus der GD-Bibliothek entfernt. Diese Funktion ist also nur verfügbar, wenn sie eine Version vor 1.6 verwenden.

ImageJPEG

(unknown)

ImageJPEG -- Ausgabe des Bildes im Browser oder als Datei

Beschreibung:

`int imagejpeg (int im [, string filename [, int quality]])`

ImageJPEG() erzeugt die Datei *filename* im JPEG-Format aus dem Zeiger auf *im*. Das Argument *im* wurde ihnen vorher durch den Aufruf der Funktion **ImageCreate()** mitgeteilt.

Die Angabe des Dateinamens *filename* ist optional. Fehlt er, werden die erzeugten Bilddaten direkt ausgegeben. Um die Angabe des Dateinamens zu umgehen, dennoch aber den Parameter als solchen zu nutzen, können sie einen Leerstring (") benutzen. Durch Verwendung der Funktion [header\(\)](#) mit der "content-type"-Angabe "image/jpg" können sie PHP-Skripte erstellen, welche die JPEG-Ausgabe direkt vornehmen.

Anmerkung: Die JPEG-Unterstützung können sie nur verwenden, wenn sie PHP mit einer GD-Bibliothek ab Version 1.8 kompiliert haben.

ImagePNG

(unknown)

ImagePNG -- Ausgabe eines Bildes an den Browser oder in eine Datei

Beschreibung:

`int imagepng (int im [, string filename])`

ImagePNG() erzeugt eine Datei im Format PNG aus *im*. Falls sie [filename] angegeben haben, wird die Ausgabe in diese Datei geschrieben, ansonsten wird die Ausgabe direkt an den Browser geleitet. *Im* ist das Ergebniss der [imagecreate\(\)](#)-Funktion.

```
<?php
$im = ImageCreateFromPNG("test.png");
ImagePNG($im);
?>
```

Die Angabe des Dateinamens ist optional. Fehlt er, werden die Bilddaten direkt ausgegeben. Durch Verwendung der Funktion [header\(\)](#) mit der "content-type"-Angabe "image/png" können sie PHP-Skripte erstellen, welche die PNG-Ausgabe direkt vornehmen.

Anmerkung: ACHTUNG: Die PNG-Unterstützung ersetzt die GIF-Funktionen ab der GD-Bibliothek Version 1.6. Diese Funktion ist also nur verfügbar, wenn sie eine Version ab 1.6 verwenden.

ImageInterlace

(unknown)

ImageInterlace -- Schaltet die Interlaced-Darstellung eines Bildes an oder aus

Beschreibung:

`int imageinterlace (int im [, int interlace])`

Mit der Funktion **imageInterlace()** setzen sie das Bit für die interlaced-Darstellung eines Bildes auf An (1) oder Aus (0).

Als Rückgabewert erhalten sie die aktuelle Einstellung des Interlaced-Modus.

ImageLine

(unknown)

ImageLine -- Zeichnen einer Linie

Beschreibung:

`int imageline (int im, int x1, int y1, int x2, int y2, int col)`

ImageLine() zeichnet eine Linie von der *x1* / *y1* Koordinate bis zur Kordinate *x2* / *y2* (die Koordinate 0 / 0 ist die obere linke Ecke von *im*). Die Linienfarbe wird durch *col* festgelegt.

Siehe auch **ImageCreate()** und **ImageColorAllocate()**.

ImageLoadFont

(unknown)

ImageLoadFont -- Lädt einen neuen Font

Beschreibung:

`int imageloadfont (string file)`

ImageLoadFont() lädt einen vom Benutzer definierten Bitmap-Font und gibt einen Zeiger auf diesen zurück. Dessen Wert ist immer grösser als 5, um Konflikte mit den eingebauten Fonts zu vermeiden.

Das Font-Datei-Format unterliegt binären Abhängigkeiten und ist von Rechner-Typ zu Rechner-Typ unterschiedlich. Das bedeutet, dass sie die Font-Dateien immer auf dem Rechner-Typ erzeugen müssen, auf dem diese Verwendung finden.

Tabelle 1 Font-Datei-Format

Byte-Position	C Daten-Typ	Beschreibung
Byte 0-3	int	Anzahl der Zeichen des Font
Byte 4-7	int	Wert des ersten Zeichens des Fonts (meist 32 für Leerzeichen)
Byte 8-11	int	Pixel-Breite jedes Zeichens
Byte 12-15	int	Pixel-Höhe jedes Zeichens
Byte 16-	Zeichen	Zeichen-Daten als Array, ein Byte je Pixel für jedes Zeichen; in der Summe also (Anzahl-Zeichen * Breite * Höhe)-Bytes.

Siehe auch **ImageFontWidth()** und **ImageFontHeight()**.

ImagePolygon

(unknown)

ImagePolygon -- Zeichnen eines Vielecks (Polygon)

Beschreibung:

`int imagepolygon (int im, array points, int num_points, int col)`

ImagePolygon() erzeugt ein Vieleck innerhalb von *im*. *Points* ist ein PHP-Array, dass die Werte für die Eckpunkte des Polygons enthält, z.B. Points[0] = x0, Points[1] = y0, Points[2] = x1, Points[3] = y1 usw. *Num_points* enthält die Gesamtzahl an Points (Points[n]).

Siehe auch [imagecreate\(\)](#) und [imagefilledpolygon\(\)](#).

ImagePSBox

(unknown)

ImagePSBox -- Ermittelt die Ausmaße des Rechtecks, das für die Ausgabe eines Textes unter Verwendung eines PostScript-Fonts (Typ 1) notwendig ist

Beschreibung:

```
array imagepsbbox (string text, int font, int size [, int space [, int tightness [, float angle]]])
```

Size ist die Größe in Pixel.

Space ermöglicht ihnen, den vorgegebenen Wert für den Zeichen-Zwischenraum eines Fonts anzugeben. Dieser Wert wird zum Normal-Wert hinzu gezählt und kann auch negativ sein.

Tightness ermöglicht ihnen die Kontrolle über die Breite eines Leerzeichens. Der angegebene Wert wird zur normalen Zeichenbreite hinzu gezählt und kann auch negativ sein.

Angle (Winkel) wird in Grad angegeben und beinhaltet den Drehungswinkel des auszugebenden Textes.

Die Parameter *space* und *tightness* haben als Einheit die Zeichenbreite, wobei der Wert 1 gleich 1/1000stel der Breite des Zeichens 'm' entspricht (Maß 'em').

Die Parameter *space*, *tightness* und *angle* sind optional.

Das umgebende Rechteck wird berechnet unter Verwendung der für die Zeichen-Abmessungen verfügbaren Informationen und unterscheidet sich immer etwas von den aktuellen Raster-Werten des Textes. Ist der Winkel (angle) mit 0 angegeben, können sie davon ausgehen, dass der Text in jede Richtung immer 1 Pixel mehr braucht als sonst.

Diese Funktion gibt ein Array zurück, welches die folgenden Elemente enthält:

0	untere linke x-Koordinate
1	untere linke y-Koordinate
2	obere rechte x-Koordinate
3	obere rechte y-Koordinate

Siehe auch [imagepstext\(\)](#).

ImagePSEncodeFont

(unknown)

ImagePSEncodeFont -- Ändert die Vektor-Beschreibung eines Fonts

Beschreibung:

`int imagepsencodefont (string encodingfile)`

Lädt einen Zeichen-Beschreibungs-Vektor aus einer Datei und verändert die Vektor-Beschreibung des Fonts. Da ein PS-Font meist Lücken oberhalb des 127. Zeichens aufweist, werden sie dies oft anpassen wollen, sofern sie eine andere Sprache als Englisch benutzen. Das genaue Dateiformat ist in der Dokumentation zu T1Lib beschrieben. T1Lib wird mit zwei gebrauchsfertigen Beschreibungsdateien (IsoLatin1.enc und IsoLatin2.enc) ausgeliefert.

Sollten sie diese Funktion ständig benutzen, ist es besser, diese Beschreibungs-Datei innerhalb der [Konfigurations-Datei](#) bei ps.default_encoding anzugeben. Dadurch wird immer die richtige Datei geladen und die Zeichen haben damit die richtige Beschreibung.

ImagePsExtendFont

(unknown)

ImagePsExtendFont -- Vergrößert oder komprimiert einen Font

Beschreibung:

```
bool imagepsextendfont (int font_index, double extend)
```

Der mit *font_index* angegebene Font wird gedehnt (*extend* ist größer als der Font) oder verkleinert (*extend* ist kleiner als der Font).

ImagePSFreeFont

(unknown)

ImagePSFreeFont -- Gibt den durch einen Typ 1 PostScript-Font belegten Speicher wieder frei

Beschreibung:

```
void imagepsfreefont (int fontindex)
```

Siehe auch **ImagePSLoadFont()**.

ImagePSLoadFont

(unknown)

ImagePSLoadFont -- Lädt einen Typ 1 PostScript-Font aus einer Datei

Beschreibung:

```
int imagepsloadfont (string filename)
```

Wenn alles in Ordnung ist, wird ein gültiger Zeiger (Index) zurück gegeben, auf den sie immer wieder zurück greifen können; ansonsten wird false zurück und eine Fehlermeldung ausgegeben.

Siehe auch **ImagePSFreeFont()**.

ImagePsSlantFont

(unknown)

ImagePsSlantFont -- Setzt einen Font schräg

Beschreibung:

bool **imagepsslantfont** (int font_index, double slant)

Der mit *font_index* angegebene Font wird um den Faktor *slant* "verschrägt" (italic!).

ImagePSText

(unknown)

ImagePSText -- Ausgabe eines Textes auf einem Bild unter Verwendung von Typ 1 PostScript-Fonts

Beschreibung:

```
array imagepstext (int 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 wird in Pixel ausgedrückt.

Foreground ist die Text-Ausgabefarbe (Vordergrund). *Background* ist die Farbe, zu der ein Farbübergang versucht werden soll. Dabei wird kein Pixel mit der Farbe *Background* ausgegeben, so dass das Bild im Hintergrund keine festen Farben benötigt.

Die Koordinaten *x* und *y* definieren den Ursprung (bzw. den Referenz-Punkt) des ersten Zeichens (ungefähr die linke untere Ecke). Dies ist anders als bei der **ImageString()**-Funktion, wo *x* und *y* die obere linke Ecke des ersten Zeichens angeben. Lesen sie in der Dokumentation zu PostScript und seines Maßsystems nach, falls sie die Funktionsweise nicht verstehen.

Space ermöglicht ihnen, die Vorgabe des Wertes für den Zeichen-Zwischenraum zu verändern. Der angegebene Wert wird addiert und kann auch negativ sein.

Tightness ermöglicht die Kontrolle über die Breite des Leerzeichens. Der Wert wird addiert und kann auch negativ sein.

Angle wird in Grad angegeben und hat die entsprechende Drehung des Ausgabetextes zur Folge.

Antialias_steps erlaubt die Kontrolle über die Anzahl der Farben, mit der der Farbübergang vom Text zum Hintergrundbild vorgenommen werden soll. Möglich sind Werte von 4 bis 16. Der höhere Wert wird empfohlen bei Textgrößen kleiner als 20, wo der Effekt bei der Textqualität deutlich sichtbar ist. Bei grösseren Werten als 20 werden 4 Farben empfohlen. Das ist weniger rechenintensiv.

Die Parameter *space* und *tightness* werden in 1/1000-stel der Breite des Buchstabens "m" (Einheit 'em') angegeben.

Die Parameter *space*, *tightness*, *angle* und *antialias* sind optional.

Diese Funktion gibt ein Array mit den folgenden Werten zurück:

0	untere linke X-Koordinate

1	untere linke Y-Koordinate
2	obere rechte X-Koordinate
3	obere rechte Y-Koordinate

Siehe auch [imagepsbbox\(\)](#).

[Zurück](#)[ImagePsSlantFont](#)[Anfang](#)[Hoch](#)[Vor](#)[ImageRectangle](#)

ImageRectangle

(unknown)

ImageRectangle -- Zeichnet ein Rechteck

Beschreibung:

`int imagerectangle (int im, int x1, int y1, int x2, int y2, int col)`

ImageRectangle() erzeugt ein Rechteck mit der Linienfarbe *col*, beginnend bei der oberen linken Koordinate X1 und Y1. Das Ende liegt bei der unteren rechten Koordinate X2 und Y2. Dabei ist 0, 0 die linke obere Ecke des Bildes *im*.

ImageSetPixel

(unknown)

ImageSetPixel -- Setzt ein einzelnes Pixel

Beschreibung:

```
int imagesetpixel (int im, int x, int y, int col)
```

ImageSetPixel() zeichnet ein Pixel an der Koordinate *x* / *y*. Dabei ist die Koordinate 0, 0 die linke obere Ecke des Bildes *im*. Die Pixelfarbe wird durch *col* bestimmt.

Siehe auch **ImageCreate()** und **ImageColorAllocate()**.

ImageString

(unknown)

ImageString -- Zeichnet einen horizontalen String

Beschreibung:

int **imagestring** (int *im*, int *font*, int *x*, int *y*, string *s*, int *col*)

ImageString() gibt den String *s* in dem durch den Parameter *im* bezeichneten Bild an den Koordinaten *x* und *y* aus. Die Koordinaten 0, 0 geben die linke obere Ecke des Bildes *im* an. Der Font erscheint in der Farbe *col*. Hat *font* den Wert 1, 2, 3, 4 oder 5 wird ein interner Font benutzt.

Siehe auch **ImageLoadFont()**.

ImageStringUp

(unknown)

ImageStringUp -- Zeichnet einen vertikalen String

Beschreibung:

`int imagestringup (int im, int font, int x, int y, string s, int col)`

ImageStringUp() gibt den String *s* vertikal innerhalb des Bildes *im* an den Koordinaten *x* und *y* aus. Die linke obere Ecke von *im* ist dabei 0, 0. Der Text wird in der Farbe *col* ausgegeben. Hat *font* den Wert 1, 2, 3, 4 oder 5 wird ein interner Font benutzt.

Siehe auch **ImageLoadFont()**.

ImageSX

(unknown)

ImageSX -- Ermittelt die Bild-Breite

Beschreibung:

```
int imagesx (int im)
```

ImageSX() ermittelt die Breite des Bildes, das durch *im* referenziert wird.

Siehe auch **ImageCreate()** und **ImageSY()**.

ImageSY

(unknown)

ImageSY -- Ermittelt die Bild-Höhe

Beschreibung:

```
int imagesy (int im)
```

ImageSY() ermittelt die Höhe des Bildes, das durch *im* referenziert wird.

Siehe auch **ImageCreate()** und **ImageSX()**.

ImageTTFBBox

(unknown)

ImageTTFBBox -- Ermittelt die Rahmenmaße für die Ausgabe eines Textes im True-Type-Format

Beschreibung:

array **imagettfbbox** (int size, int angle, string fontfile, string text)

Diese Funktion errechnet die Grösse des Bereichs für eine True-Type-Textausgabe.

text

Der zu messende Text

size

Die Font-Größe

fontfile

Der Name der TTF-Datei (kann auch eine URL sein)

angle

Der Winkel, in dem *text* ausgegeben wird

ImageTTFBBox() gibt ein Array mit acht Werten zurück. Diese enthalten die vier Punkte, durch die der Ausgabebereich markiert wird:

0	untere linke Ecke, X-Position
1	untere linke Ecke, Y-Position
2	untere rechte Ecke, X-Position
3	untere rechte Ecke, Y-Position
4	obere rechte Ecke, X-Position
5	obere rechte Ecke, Y-Position
6	obere linke Ecke, X-Position
7	obere linke Ecke, Y-Position

Die Punkte sind relativ zum *text*. Ohne Rücksicht auf den Winkel ist oben-links also die obere linke Ecke, wobei die Textausgabe horizontal gesehen wird.

Diese Funktion erfordert sowohl die GD- als auch die [FreeType](#)-Bibliothek.

Siehe auch **ImageTTFText()**.

[Zurück](#)

ImageSY

[Anfang](#)

[Hoch](#)

[Vor](#)

ImageTTFText

ImageTTFText

(unknown)

ImageTTFText -- Erzeugt TTF-Text im Bild

Beschreibung:

array **imageTTFtext** (int *im*, int *size*, int *angle*, int *x*, int *y*, int *col*, string *fontfile*, string *text*)

ImageTTFText() zeichnet den Text *text* in das per *im* angegebene Bild, beginnend bei den Koordinaten *x* und *y* (oben links ist 0, 0) mit einem Ausgabewinkel von *angle* in der Farbe *col* unter Verwendung der TTF-Datei, die mittels *fontfile* benannt wurde.

Die per *x* und *y* angegebenen Koordinaten definieren den Startpunkt des ersten Zeichens (in etwa die linke untere Ecke). Das ist anders als bei der **ImageString()**-Funktion, bei der *x* und *y* die obere rechte Ecke des ersten Zeichens darstellen.

Angle ist in Grad anzugeben. 0 ist eine normale Ausrichtung von links nach rechts (von 9 nach 3 Uhr). Höhere Werte entsprechen einer immer weiteren Drehung entgegen dem Uhrzeiger-Sinn. Eine Drehung von 90 Grad entspricht also einer Text-Ausrichtung von unten nach oben (von 6 nach 12 Uhr).

Fontfile ist der Pfad zur TTF-Datei, die sie verwenden wollen.

Text ist der auszugebende String, der auch UTF-8-Zeichenfolgen enthalten darf (z.B. {)

Col ist der Farbindex. Verwenden sie den negativen Index-Wert, führt das zum Aufheben des Farbüberlaufs (Antialiasing).

ImageTTFText() gibt ein Array aus acht Elementen zurück, das die vier Ecken des Textausgabe-Bereichs enthält (wie **ImageTTFBBox()**). Die Reihenfolge ist dabei oben links, oben rechts, unten rechts und unten links. Die Punkte sind dabei relativ zum Text. Deshalb gibt oben links die linke Ecke bei horizontaler Sichtweise an.

Das Beispiel-Skript erzeugt eine schwarze GIF-Grafik von 400x30 Pixel mit dem weissen Text "Teste..." im TT-Font Arial.

Beispiel 1 ImageTTFText

```
<?php
Header ("Content-type: image/gif");
$im = imagecreate (400, 30);
$black = ImageColorAllocate ($im, 0, 0, 0);
$white = ImageColorAllocate ($im, 255, 255, 255);
ImageTTFText ($im, 20, 0, 10, 20, $white, "/path/arial.ttf",
              "Teste... Omega: &#937;");
ImageGif ($im);
ImageDestroy ($im);
?>
```

Diese Funktion erfordert sowohl die GD- als auch die [FreeType](#)-Bibliothek.

Siehe auch **ImageTTFBBox()**.

[Zurück](#)[ImageTTFBBox](#)[Anfang](#)[Hoch](#)[Vor](#)[ImageTypes](#)

ImageTypes

(unknown)

ImageTypes -- Gibt die von der aktuell verwendeten PHP-Version unterstützten Grafik-Formate zurück

Beschreibung:

```
int imageTypes(void);
```

Der von dieser Funktion zurück gegebene Bit-Wert stellt die von der in der aktuellen PHP-Kompilation enthaltenen GD-Version unterstützten Grafik-Formate dar. Die folgenden Bit-Werte sind möglich: IMG_GIF, IMG_JPG, IMG_PNG oder IMG_WBMP. Um das Vorhanden sein der GD-Unterstützung zu testen, können sie folgendes machen:

Beispiel 1 ImageTypes

```
<?php
if (ImageTypes() & IMG_PNG) {
    echo "PNG-Unterstützung ist vorhanden.";
}
?>
```


XXVI. IMAP, POP3 und NNTP Funktionen

Dieses Paket baut auf auf der c-client Bibliothek auf, die Sie unter der URL <ftp://ftp.cac.washington.edu/imap/> erhalten. Kompilieren Sie diese und kopieren Sie `c-client/c-client.a` nach `/usr/local/lib/libc-client.a` sowie `c-client/rfc822.h`, `mail.h` und `linkage.h` nach `/usr/local/include` oder in ein anderes Verzeichnis in Ihrem Link- bzw. Include-Pfad. Anschließend konfigurieren und kompilieren Sie PHP mit der Option `--with-imap`.

Beachten Sie, daß diese Funktionen nicht auf das IMAP -Protokol beschränkt sind, auch wenn der Name dies vermuten läßt. Die zugrundeliegende c-client Bibliothek unterstützt auch POP3, NNTP und lokale Mailbox-Zugriffe.

Diese Dokumentation kann nicht alle Bereiche beschreiben, die von den enthaltenen Funktionen berührt werden. Weitergehende Informationen erhalten Sie in der Dokumentation der c-client Bibliothek (Datei `docs/internal.txt` im Quellpaket der Bibliothek) sowie in folgenden RFC-Dokumenten:

- [RFC821](#) : Simple Mail Transfer Protocol (SMTP)
- [RFC822](#) : 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)

Eine ausführliche Übersicht bietet auch das Buch [Programming Internet Email](#) von David Wood.

Inhaltsverzeichnis

[imap_append](#) Fügt eine String-Nachricht an das angegebene Postfach an.

[imap_base64](#) Dekodiert BASE64-codierten Text

[imap_body](#) Liest den Körper einer Nachricht

[imap_check](#) Prüft den Status des aktuelle Postfachs

[imap_close](#) Schließt eine IMAP-Verbindung

[imap_createmailbox](#) Erzeugt ein neues Postfach

- [imap_delete](#) Merkt eine Nachricht des aktuellen Postfachs zum Löschen vor.
- [imap_deletemailbox](#) Löscht ein Postfach
- [imap_expunge](#) Löscht alle zum Löschen vorgemerkten Nachrichten
- [imap_fetchbody](#) Liefert einen bestimmten Abschnitt aus dem Körper einer Nachricht.
- [imap_fetchstructure](#) Liefert die Struktur der angegebenen Nachricht
- [imap_header](#) Liefert den Kopf einer Nachricht
- [imap_headers](#) Liefert eine Zusammenfassung aller Nachrichtenköpfe eines Postfachs
- [imap_listmailbox](#) Liefert eine Liste der Postfach-Namen
- [imap_getmailboxes](#) Liefert detaillierte Informationen über eine Auswahl von Postfächern
- [imap_listsubscribed](#) Liefert eine Liste aller abonnierten Postfächer
- [imap_getsubscribed](#) Liefert eine Auswahl aller abonnierten Postfächer
- [imap_mail_copy](#) Kopiert Nachrichten in ein Postfach
- [imap_mail_move](#) Verschiebt Nachrichten in ein anderes Postfach
- [imap_num_msg](#) Liefert die Anzahl der Nachrichten im aktuellen Postfach
- [imap_num_recent](#) Liefert die Anzahl der neu hinzugekommenen Nachrichten im aktuellen Postfach
- [imap_open](#) Öffnet eine Verbindung zu einem Postfach auf einem POP-, IMAP- oder NNTP-Server
- [imap_ping](#) Prüft, ob die aktuelle Verbindung noch verfügbar ist
- [imap_renamemailbox](#) Ändert den Namen eines Postfachs
- [imap_reopen](#) Wechselt das aktuelle Postfach der Verbindung
- [imap_subscribe](#) Abbonieren eines Postfachs
- [imap_undelete](#) Nimmt eine bereits gesetzte Löschmarkierung einer Nachricht zurück
- [imap_unsubscribe](#) Abbonement eines Postfachs beenden
- [imap_qprint](#) Konvertiert einen quoted-printable kodierten String in einen 8bit-String
- [imap_8bit](#) Konvertiert einen String in einen quoted-printable kodierten String.
- [imap_binary](#) Konvertiert 8Bit-Text in einen BASE64-kodierten String.
- [imap_scanmailbox](#) Durchsucht Postfächer nach einen String
- [imap_mailboxmsginfo](#) Liefert Informationen über das aktuelle Postfach
- [imap_rfc822_write_address](#) Bildet aus Realnamen, Postfach und Server eine korrekt formatierte Mail-Adresse
- [imap_rfc822_parse_adrlist](#) Parsen eines Adress-Strings
- [imap_setflag_full](#) Setzt Flags einer Nachricht
- [imap_clearflag_full](#) Löscht Flags einer Nachricht
- [imap_sort](#) Sortiert Nachrichten eines Postfachs

- [imap_fetchheader](#) Liefert den Kopf einer Nachricht
- [imap_uid](#) Liefert die UID zu einem gegebenen Nachrichten-Index
- [imap_msgno](#) Liefert den aktuellen Nachrichten-Index zu einer UID
- [imap_search](#) Sucht Nachrichten, die den übergebenen Suchkriterien entsprechen
- [imap_last_error](#) Diese Funktion liefert die zuletzt während dieses Seitenzugriffs aufgetretene Fehlermeldung (falls vorhanden)
- [imap_errors](#) Diese Funktion liefert alle bisher aufgetretenen Fehlermeldungen
- [imap_alerts](#) Diese Funktion liefert alle alert-Meldungen zurück, die während der Ausführung dieses Requests bzw. seit dem letzten Reset des alert-Stacks aufgetreten sind.
- [imap_status](#) Liefert ausgewählte Statusinformationen zum angegebenen Postfach
- [imap_utf7_decode](#) Dekodiert einen String im modifizierten UTF-7 Format.
- [imap_utf7_encode](#) Kodiert Text im modifizierten UTF-7 Format
- [imap_utf8](#) Konvertiert Text zu UTF8
- [imap_fetch_overview](#) Liefert einen Auszug aus den Header-Feldern von Nachrichten
- [imap_mime_header_decode](#) Decode MIME header elements
- [imap_mail_compose](#) Erzeugt eine MIME-Nachricht aus gegebenen Header- und Body-Teilen
- [imap_mail](#) Versendet eine Email

imap_append

(PHP3 , PHP4)

imap_append -- Fügt eine String-Nachricht an das angegebene Postfach an.

Beschreibung

```
int imap_append (int imap_stream, string mbox, string message [, string flags])
```

Liefert true bei Erfolg und false bei Fehlern.

imap_append() fügt den String aus *message* als neue Nachricht an die durch *mbox* spezifizierte Mailbox an. Wenn der optionale *flags* spezifiziert wird, so werden die Flags der Nachricht in der Mailbox entsprechend gesetzt.

Beispiel 1 imap_append() Beispiel

```
$stream = imap_open ("{your.imap.host}INBOX.Drafts", "username", "password");

$check = imap_check ($stream);
print "Msg Count before append: " . $check->Nmsgs . "<br>\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 . "<br>\n";

imap_close ($stream);
```

Warnung

Der Cyrus IMAP Server erwartet zwingend "\r\n" als Zeilenende-Kennung, ein einfaches "\n" wird nicht richtig verarbeitet.

imap_base64

(PHP3 , PHP4)

imap_base64 -- Dekodiert BASE64-codierten Text

Beschreibung

string **imap_base64** (string text)

Die **imap_base64()** Funktion decodiert BASE64 codierten Text (gemäß [RFC2045](#), Abschnitt 6.8). Der decodierte Text wird als String zurückgegeben.

Siehe auch [imap_binary\(\)](#).

imap_body

(PHP3 , PHP4)

imap_body -- Liest den Körper einer Nachricht

Beschreibung

string **imap_body** (int imap_stream, int msg_number [, int flags])

imap_body() liefert den Körper der Nachricht mit der Nummer *msg_number* im aktuellen Postfach. Die optionalen *flags* können eine Bitmaske enthalten, die sich aus folgenden Werten zusammensetzt:

- FT_UID - *msg_number* ist eine UID
- FT_PEEK - Das \Seen-Flag wird nicht beeinflußt.
- FT_INTERNAL - Die Nachricht wird im internen Format zurückgegeben eine CRLF-Umwandlung findet nicht statt.

imap_check

(PHP3 , PHP4)

imap_check -- Prüft den Status des aktuelle Postfachs

Beschreibung

object **imap_check** (int imap_stream)

Die **imap_check()** Funktion prüft den Status des aktuellen Postfach und liefert die entsprechenden Felder in einem Objekt mit folgenden Attributen zurück:

- Date - Datum der letzten Änderung des Postfach-Inhalts
- Driver - Zugriffsprotokoll: POP, IMAP, NNTP
- Mailbox - Name des Postfachs
- Nmsgs - Anzahl der Nachrichten im Postfach
- Recent - Anzahl der neuen Nachrichten im Postfach

Beispiel 1 imap_check() Beispiel

```
$mbox = imap_open ("{your.imap.host}INBOX", "username", "password")
|| die ("can't connect: " . imap_last_error());

$check = imap_check ($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" ;
} else {
    print "imap_check() failed: " . imap_last_error() . "<br>\n";
}

imap_close ($mbox);
```

[Zurück](#)

imap_body

[Anfang](#)

[Hoch](#)

[Vor](#)

imap_close

imap_close

(PHP3 , PHP4)

imap_close -- Schließt eine IMAP-Verbindung

Beschreibung

```
int imap_close (int imap_stream [, int flags])
```

Schließt eine IMAP-Verbindung. Mit dem optionalen *flag* CL_EXPUNGE kann ein Bereinigen (physikalisches Löschen aller zum löschen markierten Nachrichten) ausgelöst werden.

imap_createmailbox

(PHP3 , PHP4)

imap_createmailbox -- Erzeugt ein neues Postfach

Beschreibung

int **imap_createmailbox** (int imap_stream, string mbox)

imap_createmailbox() legt ein über den Parameter *mbox* spezifiziertes neues Postfach an.

Liefert true bei Erfolg und false bei Fehlern.

Siehe auch [imap_renamemailbox\(\)](#) and [imap_deletemailbox\(\)](#) sowie [imap_open\(\)](#) für das Format des Parameters *mbox*.

Beispiel 1 imap_createmailbox() Beispiel

```
$mbox = imap_open ("{your.imap.host}", "username", "password", OP_HALFOPEN)
|| die ("can't connect: ".imap_last_error());

$name1 = "phpnewbox";
$name2 = imap_utf7_encode ("phpnewböx");

$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 finally 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";
        }
    }
}
```

imap_createmailbox

```
    } 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);
```

[Zurück](#)

imap_close

[Anfang](#)

[Hoch](#)

[Vor](#)

imap_delete

imap_delete

(PHP3 , PHP4)

imap_delete -- Merkt eine Nachricht des aktuellen Postfachs zum Löschen vor.

Beschreibung

`int imap_delete (int imap_stream, int msg_number [, int flags])`

Diese Funktion liefert immer true, eventuelle Fehler können nicht erkannt werden!

Die **imap_delete()** Funktion merkt die mit *msg_number* angegebene Nachricht des aktuellen Postfachs zum Löschen vor. Enthält der optionale Parameter *flags* den Wert *FT_UID*, so wird der Parameter *msg_number* als *UID* interpretiert. Die Nachricht verbleibt zunächst noch im Postfach, endgültig gelöscht wird sie (zusammen mit allen anderen entsprechend markierten Nachrichten) durch Aufruf der Funktion [imap_expunge\(\)](#) oder durch [imap_close\(\)](#) mit dem optionalen Flag *CL_EXPUNGE*.

Beispiel 1 Imap_delete() Beispiel

```
$mbox = imap_open ("{your.imap.host}INBOX", "username", "password")
|| 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

(PHP3 , PHP4)

imap_deletemailbox -- Löscht ein Postfach

Beschreibung

int **imap_deletemailbox** (int imap_stream, string mbox)

Imap_deletemailbox() löscht das angegebene Postfach inclusive aller darin enthaltener Nachrichten und Unter-Postfächer.

Liefert true bei Erfolg und false bei Fehlern.

Siehe auch [imap_createmailbox\(\)](#) und [imap_renamemailbox\(\)](#) sowie [imap_open\(\)](#) für das Format des *mbox* Parameters.

imap_expunge

(PHP3 , PHP4)

imap_expunge -- Löscht alle zum Löschen vorgemerkten Nachrichten

Beschreibung

int **imap_expunge** (int imap_stream)

Imap_expunge() löscht endgültig alle von den Funktionen [imap_delete\(\)](#), [imap_mail_move\(\)](#) oder [imap_setflag_full\(\)](#) zum Löschen vorgemerkten Nachrichten.

Diese Funktion liefert immer true, eventuelle Fehler können nicht erkannt werden!

imap_fetchbody

(PHP3 , PHP4)

imap_fetchbody -- Liefert einen bestimmten Abschnitt aus dem Körper einer Nachricht.

Beschreibung

string **imap_fetchbody** (int imap_stream, int msg_number, string part_number [, flags flags])

Diese Funktion liefert einen bestimmten Teil einer MIME-kodierten Nachricht als String. Die Nummer des gewünschten Teils besteht aus von Punkten getrennten Zahlen, die den Index des Teils gemäß der IMAP4-Spezifikationen bezeichnen. Der Inhalt wird dabei von dieser Funktion nicht dekodiert.

Die Bitmaske *flags* kann aus folgenden Werten zusammengesetzt werden:

- FT_UID - *msg_number* enthält eine UID
- FT_PEEK - Das \Seen Flag der Nachricht wird nicht verändert
- FT_INTERNAL - Das Ergebnis wird ohne jeden Versuch einer Konvertierung der Zeilenenden zurückgegeben

imap_fetchstructure

(PHP3 , PHP4)

imap_fetchstructure -- Liefert die Struktur der angegebenen Nachricht

Beschreibung

object **imap_fetchstructure** (int imap_stream, int msg_number [, int flags])

Diese Funktion liefert die gesamten Strukturinformationen zu einer gegebenen Nachricht. Das zurückgegebene Objekt enthält den Umschlag, die internen Daten, Größe und Flags der Nachricht sowie ein (rekursives) Array von ähnlichen Objekten für jeden enthaltenen MIME-Teil der Nachricht. Enthält der optionale Parameter *flags* den Wert *FT_UID*, so wird *msg_number* als UID anstelle eines Index interpretiert.

Das zurückgegebene Objekt verfügt über die folgenden Attribute:

Tabelle 1 Zurückgegebene Objekte von imap_fetchstructure()

type	Primärer MIME-Typ des Nachrichten-Teils
encoding	Transfer-Kodierung
ifsubtype	Wahr, wenn das Feld subtype gesetzt ist
subtype	MIME Subtyp
ifdescription	Wahr, wenn ein Beschreibungsstring existiert
description	Inhalt des Bechreibungsstrings
ifid	Wahr, wenn ein Identifikationsstring existiert
id	Identificationsstring
lines	Anzahl der Zeilen dieses Teils
bytes	Größe des Teils in Byte
ifdisposition	Wahr, wenn der Dispositionsstring gesetzt ist
disposition	Dispositionsstring
ifdparameters	Wahr, wenn Dispositions-Parameterwerte gesetzt sind
dparameters	Array der DispositionsParameter
ifparameters	Wahr, wenn Parameter gesetzt sind
parameters	Array der MIME Parameter

parts	Array von Objekten, die die einzelnen Unterteile der Nachricht beschreiben

1. dparameters ist ein Array von Objekten mit jeweils einem "attribute" und einem "value" Attribut.
2. parameters ist ein Array von Objekten mit jeweils einem "attribute" und einem "value" Attribut.
3. Die Elemente des parts Array entsprechen dem von der Funktion zurückgegebenen Objekt (usf.).

Tabelle 2 Primary body type

0	text
1	multipart
2	message
3	application
4	audio
5	image
6	video
7	other

Tabelle 3 Transfer encodings

0	7BIT
1	8BIT
2	BINARY
3	BASE64
4	QUOTED-PRINTABLE
5	OTHER

[Zurück](#)

imap_fetchbody

[Anfang](#)[Hoch](#)[Vor](#)

imap_header

imap_header

(PHP3 , PHP4)

imap_header -- Liefert den Kopf einer Nachricht

Beschreibung

object **imap_header** (int imap_stream, int msg_number [, int fromlength [, int subjectlength [, string defaulthost]])

Diese Funktion liefert ein Objekt, das aus verschiedenen Header-Elementen gebildet wird.

remail, date, Date, subject, Subject, in_reply_to, message_id,
newsgroups, followup_to, references

message flags:

Recent - 'R' wenn neu und bereits gelesen,
 'N' wenn neu und noch nicht gelesen,
 '' wenn nicht neu
Unseen - 'U' wenn noch nicht gelesen UND nicht neu,
 '' wenn gelesen OR neu und noch nicht gelesen
Answered - 'A' wenn beantwortet,
 '' sonst
Deleted - 'D' wenn zum Löschen vorgemerkt
 '' sonst
Draft - 'X' wenn als Entwurf markiert,
 '' sonst
Flagged - 'F' wenn als 'wichtig' markiert,
 '' sonst

Beachten Sie bitte die etwas seltsame Recent/Unseen-Zuordnung.
Eine Nachricht wurde noch nicht gelesen, wenn folgende Bedingung
erfüllt ist:

Unseen == 'U' || Recent == 'N'

toaddress (Inhalt der to: Zeile, bis zu 1024 Zeichen)

to[] (ein Array aus Objekten aus der to: Zeile, jeweils mit:)

imap_header

personal
adl
mailbox
host

fromaddress (Inhalt der from: Zeile, up to 1024 characters)

from[] (Inhalt der From Zeile, bestehend aus:)

personal
adl
mailbox
host

ccaddress (Inhalt der cc: Zeile, max. 1024 Zeichen)

cc[] (Inhalt der Cc Zeile, bestehend aus:)

personal
adl
mailbox
host

bccaddress (Inhalt der bcc Zeile, max. 1024 Zeichen)

bcc[] (Inhalt der Bcc Zeile, bestehend aus:)

personal
adl
mailbox
host

reply_toaddress (Inhalt der reply_to: Zeile, max. 1024 Zeichen)

reply_to[] (Inhalt der Reply_to Zeile, bestehend aus:)

personal
adl
mailbox
host

senderaddress (Inhalt der sender: Zeile, max. 1024 Zeichen)

sender[] (Inhalt der sender Zeile, bestehend aus:)

personal
adl
mailbox
host

return_path (Inhalt der return-path: Zeile, max. 1024 Zeichen)

return_path[] (Inhalt der return_path Zeile, bestehend aus:)

personal
adl
mailbox

imap_header

host

update (Sende-Zeitpunkt der Nachricht in Sekunden seit 1.1.1970)

fetchfrom (From: Zeile, formatiert auf
max. *fromlength* Zeichen)

fetchsubject (Subject: Zeile, formatiert auf
max. *subjectlength* Zeichen)

[Zurück](#)

imap_fetchstructure

[Anfang](#)

[Hoch](#)

[Vor](#)

imap_headers

imap_headers

(PHP3 , PHP4)

imap_headers -- Liefert eine Zusammenfassung aller Nachrichtenköpfe eines Postfachs

Beschreibung

array **imap_headers** (int imap_stream)

Diese Funktion liefert ein String-Array, das je Nachricht einen String mit vorformatierten Informationen aus dem Nachrichtenheader enthält.

imap_listmailbox

(PHP3 , PHP4)

imap_listmailbox -- Liefert eine Liste der Postfach-Namen

Beschreibung

array **imap_listmailbox** (int imap_stream, string ref, string pat)

Liefert die Namen der gefundenen Postfächer als String-Array. *ref* und *pat* werden in [imap_getmailboxes\(\)](#) beschrieben.

imap_getmailboxes

(PHP3 >= 3.0.12, PHP4 >= 4.0b4)

imap_getmailboxes -- Liefert detaillierte Informationen über eine Auswahl von Postfächern

Beschreibung

array **imap_getmailboxes** (int imap_stream, string ref, string pat)

imap_getmailboxes() liefert ein Array zurück, das für jedes gefundene Postfach ein Objekt mit de Attributen *name*, *delimiter* und *attributes* enthält. *name* enthält dabei den vollständigen Pfadnamen des Postfachs, *delimiter* das verwendete Trennzeichen zwischen Postfach-Ebenen, und *attributes* besteht aus einer Bitmaske, die einzelne Eigenschaften des Postfachs beschreibt:

- LATT_NOINFERIORS - es existieren keine weiteren Postfächer unterhalb dieses Postfachs
- LATT_NOSELECT - Dieses "Postfach" ist nur ein Behälter für weitere Postfächer.
Es kann selbst keine Nachrichten enthalten und somit auch nicht geöffnet werden.
- LATT_MARKED - Dieses Postfach wurde markiert (nur bei UW-IMAPD).
- LATT_UNMARKED - Dieses Postfach wurde nicht markiert (nur bei UW-IMAPD).

Namen von Postfächern, die Zeichen außerhalb des druckbaren ASCII-Bereiches enthalten, können mit Hilfe der Funktion [imap_utf7_decode\(\)](#) dekodiert werden.

ref sollte normalerweise nur die Spezifikation des Servers in der in [imap_open\(\)](#) beschriebenen Form enthalten. Über *pat* kann ein Startpunkt innerhalb der Postfach-Hierarchie vorgegeben werden. Wünschen Sie die Ausgabe aller Postfächer, so übergeben Sie als *pat* '*'.

Die Zeichen '*' und '%' haben innerhalb von *pattern* eine besondere Bedeutung:

- '*' - entspricht beliebig vielen Zeichen (incl. des Trennzeichens zw. Postfach-Namen)
- '%' - entspricht beliebig vielen Zeichen innerhalb eines Postfach-Namens

So liefert "*" alle verfügbaren Postfächer auf dem Server, "INBOX/*" dementsprechend alle persönlichen Postfächer des aktuellen Benutzers. "%" dagegen liefert nur die Toplevel-Postfächer des Servers, "INBOX/%" dementsprechend nur die Toplevel-Postfächer des aktuellen Benutzers. Ein '%' am Ende von *pattern* entspricht zusätzlich auch allen Zeichenfolgen, die mit dem Trennzeichen beginnen.

Beispiel 1 Imap_getmailboxes() Beispiel

```
$mbox = imap_open ("{your.imap.host}", "username", "password", OP_HALFOPEN)
|| 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);
```

[Zurück](#)[imap_listmailbox](#)[Anfang](#)[Hoch](#)[Vor](#)[imap_listsubscribed](#)

imap_listsubscribed

(PHP3 , PHP4)

imap_listsubscribed -- Liefert eine Liste aller abonnierten Postfächer

Beschreibung

array **imap_listsubscribed** (int imap_stream, string ref, string pattern)

Diese Funktion entspricht weitgehend [imap_listmailbox\(\)](#), allerdings werden nur Postfächer zurückgegeben, die der aktuelle Benutzer auch abonniert hat.

imap_getsubscribed

(PHP3 >= 3.0.12, PHP4 >= 4.0b4)

imap_getsubscribed -- Liefert eine Auswahl aller abonnierten Postfächer

Beschreibung

array **imap_getsubscribed** (int imap_stream, string ref, string pattern)

Diese Funktion entspricht weitgehend [imap_getmailboxes\(\)](#), allerdings werden nur Postfächer zurückgegeben, die der aktuelle Benutzer auch abonniert hat.

imap_mail_copy

(PHP3 , PHP4)

imap_mail_copy -- Kopiert Nachrichten in ein Postfach

Beschreibung

`int imap_mail_copy (int imap_stream, string msglist, string mbox [, int flags])`

Diese Funktion kopiert die in *msglist* angegebenen Nachrichten in das Postfach *mbox*. *msglist* kann eine einzelne Nachrichtennummer oder Aufzählungen und Bereiche gemäß [RFC2060](#) enthalten.

Die Bitmaske *flags* kann aus den folgenden Werten zusammengesetzt werden.

- CP_UID - Die Nummern in *msglist* entsprechen den UIDs der gewünschten Nachrichten
- CP_MOVE - Die ursprünglichen Nachrichten werden nach dem Kopieren gelöscht. copying

Liefert true bei Erfolg und false bei Fehlern.

imap_mail_move

(PHP3 , PHP4)

imap_mail_move -- Verschiebt Nachrichten in ein anderes Postfach

Beschreibung

`int` **imap_mail_move** (`int` imap_stream, `string` msglist, `string` mbox [, `int` flags])

Diese Funktion verschiebt die in *msglist* angegebenen Nachrichten in das Postfach *mbox*. *msglist* kann eine einzelne Nachrichtennummer oder Aufzählungen und Bereiche gemäß [RFC2060](#) enthalten.

Die Bitmaske *flags* kann den folgenden Wert enthalten:

- CP_UID - Die Nummern in *msglist* entsprechen den UIDs der gewünschten Nachrichten

Liefert true bei Erfolg und false bei Fehlern.

imap_num_msg

(PHP3 , PHP4)

imap_num_msg -- Liefert die Anzahl der Nachrichten im aktuellen Postfach

Beschreibung

```
int imap_num_msg (int imap_stream)
```

Liefert die Anzahl der Nachrichten im aktuellen Postfach

[Zurück](#)[Anfang](#)[Vor](#)

imap_mail_move

[Hoch](#)

imap_num_recent

imap_num_recent

(PHP3 , PHP4)

imap_num_recent -- Liefert die Anzahl der neu hinzugekommenen Nachrichten im aktuellen Postfach

Beschreibung

```
int imap_num_recent (int imap_stream)
```

Liefert die Anzahl der neu hinzugekommenen Nachrichten im aktuellen Postfach

imap_open

(PHP3 , PHP4)

imap_open -- Öffnet eine Verbindung zu einem Postfach auf einem POP-, IMAP- oder NNTP-Server

Beschreibung

int **imap_open** (string mailbox, string username, string password [, int flags])

Diese Funktion liefert ein Verbindungs-Handle bzw. false bei evtl. auftretenden Fehlern. Verbindungen können nicht nur zu IMAP-, sondern auch zu POP- oder NNTP-Servern aufgebaut werden, wobei dann allerdings nicht der gesamte Funktionsumfang zur Verfügung steht. Liefert ein IMAP-Streamhandle oder false bei Fehlern.

Der Name eines Postfachs setzt sich zusammen aus der Bezeichnung des Servers und dem Pfad des Postfachs auf diesem Server. Der spezielle Name INBOX steht dabei als Synonym für das persönliche Postfach des jeweiligen Benutzers. Die Bezeichnung des Servers setzt sich zusammen aus dem eigentlichen Servernamen oder der entsprechenden IP-Adresse sowie das gewünschte Protokoll (getrennt durch '/') und optional der zu benutzende Port (getrennt durch ':'). Die gesamte Serverbezeichnung wird in '{' und '}' eingeschlossen und muß immer mit angegeben werden. Enthält der Name eines Postfachs Zeichen außerhalb des druckbaren ASCII-Bereiches wie z.B. Umlaute oder das 'ß', so muß er mit Hilfe der Funktion [imap_utf7_encode\(\)](#) codiert werden.

Der optionale Parameter *flags* kann sich aus den folgenden Bitmasken zusammensetzen:

- OP_READONLY - Über diese Verbindung darf nur lesend zugegriffen werden
- OP_ANONYMOUS - Bei NNTP-Verbindungen wird die Datei .newsrc nicht aktualisiert
- OP_HALFOPEN - es wird eine IMAP- oder NNTP-Verbindung aufgebaut, aber noch kein Postfach ausgewählt
- CL_EXPUNGE - beim Schließen der Verbindung werden alle zum Löschen vorgemerkte Nachrichten endgültig gelöscht (siehe auch [imap_delete\(\)](#) und [imap_expunge\(\)](#))

Das folgende Beispiel stellt eine Verbindung zu einem lokalen IMAP-Server (IMAP = Port 143) her:

```
$mbox = imap_open ("{localhost:143}INBOX", "user_id", "password");
```

Eine Verbindung zu einem lokalen POP3-Server (POP3 = Port 110) erhalten sie mit:

```
$mbox = imap_open ("{localhost/pop3:110}INBOX", "user_id", "password");
```

Eine Verbindung zu einem lokalen News-Server (NNTP = Port 119) erhalten Sie mit:

```
$nntp = imap_open ("{localhost/nntp:119}comp.test", "", "");
```

Eine Verbindung zu einem nicht lokalen Server können Sie herstellen, indem Sie in den oben genannten Beispielen "localhost" durch den Namen oder die IP-Adresse des Servers ersetzen.

Beispiel 1 Imap_open() Beispiel

```
$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);
```

[Zurück](#)
[imap_num_recent](#)
[Anfang](#)
[Hoch](#)
[Vor](#)
[imap_ping](#)

imap_ping

(PHP3 , PHP4)

imap_ping -- Prüft, ob die aktuelle Verbindung noch verfügbar ist

Beschreibung

int **imap_ping** (int imap_stream)

Liefert true, wenn die aktuelle Verbindung noch verfügbar ist, sonst false.

Die **imap_ping()** Funktion schickt ein Ping-Signal an den Mailserver und prüft so die Verfügbarkeit der Verbindung. Auf diese Weise kann die Verbindung zu einem Server aufrechterhalten werden, der Verbindungen nach einer gewissen Zeit ohne Aktivität automatisch beendet (da PHP-Skripte i.A. nicht sehr lange laufen, kann ich mir kaum vorstellen, das jemand diese Funktion wirklich benötigt).

imap_renamemailbox

(PHP3 , PHP4)

imap_renamemailbox -- Ändert den Namen eines Postfachs

Beschreibung

int **imap_renamemailbox** (int imap_stream, string old_mbox, string new_mbox)

Die Funktion **imap_renamemailbox()** ändert den Namen eines Postfachs.

Liefert true bei Erfolg und false bei Fehlern.

Siehe auch [imap_createmailbox\(\)](#) und [imap_deletemailbox\(\)](#) sowie [imap_open\(\)](#) für das Format der Parameter *old_mbox* und *new_mbox*.

imap_reopen

(PHP3 , PHP4)

imap_reopen -- Wechselt das aktuelle Postfach der Verbindung

Beschreibung

int imap_reopen (string imap_stream, string mailbox [, string flags])

Mit Hilfe dieser Funktion kann auf IMAP- und NNTP-Servern ein anderes Postfach als das bisher aktuelle ausgewählt werden.

Der optionale Parameter *flags* kann sich aus den folgenden Bitmasken zusammensetzen:

- OP_READONLY - Auf das Postfach soll nur lesend zugegriffen werden
- OP_ANONYMOUS - Bei NNTP-Verbindungen wird die persönliche .newsrc Datei nicht aktualisiert
- OP_HALFOPEN - bei IMAP- oder NNTP-Servern wird in den allgemeinen Verbindungsstatus ohne aktuelles Postfach zurückgeschaltet
- CL_EXPUNGE - beim Schließen der Verbindung werden alle zum Löschen vorgemerkte Nachrichten endgültig gelöscht (siehe auch [imap_delete\(\)](#) und [imap_expunge\(\)](#))

Liefert true bei Erfolg und false bei Fehlern.

imap_subscribe

(PHP3 , PHP4)

imap_subscribe -- Abbonieren eines Postfachs

Beschreibung

int **imap_subscribe** (int imap_stream, string mbox)

Abbonieren eines Postfachs.

Liefert true bei Erfolg und false bei Fehlern.

imap_undelete

(PHP3 , PHP4)

imap_undelete -- Nimmt eine bereits gesetzte Löschmarkierung einer Nachricht zurück

Beschreibung

int **imap_undelete** (int imap_stream, int msg_number)

Diese Funktion nimmt eine evtl. durch die Funktionen [imap_delete\(\)](#) oder [imap_mail_move\(\)](#) gesetzte Löschmarkierung für die durch *msg_number* spezifizierte Nachricht zurück.

Liefert true bei Erfolg und false bei Fehlern.

imap_unsubscribe

(PHP3 , PHP4)

imap_unsubscribe -- Abbonement eines Postfachs beenden

Beschreibung

int **imap_unsubscribe** (int imap_stream, string mbox)

Abbonement eines Postfachs beenden.

Liefert true bei Erfolg und false bei Fehlern.

imap_qprint

(PHP3 , PHP4)

imap_qprint -- Konvertiert einen quoted-printable kodierten String in einen 8bit-String

Beschreibung

string **imap_qprint** (string string)

Konvertiert einen quoted-printable kodierten String (gemäß [RFC2045](#), Abschnitt 6.7) in einen 8bit-String.

Siehe auch [imap_8bit\(\)](#).

imap_8bit

(PHP3 , PHP4)

imap_8bit -- Konvertiert einen String in einen quoted-printable kodierten String.

Beschreibung

string **imap_8bit** (string string)

Konvertiert einen String in einen quoted-printable kodierten String (gemäß [RFC2045](#), Abschnitt 6.7).

Siehe auch [imap_qprint\(\)](#).

imap_binary

(PHP3 >= 3.0.2, PHP4)

imap_binary -- Konvertiert 8Bit-Text in einen BASE64-kodierten String.

Beschreibung

string **imap_binary** (string string)

Konvertiert 8Bit-Text in einen BASE64-kodierten String (gemäß [RFC2045](#), Abschnitt 6.8).

Siehe auch [imap_base64\(\)](#).

imap_scanmailbox

(PHP3 , PHP4)

imap_scanmailbox -- Druchsucht Postfächer nach einen String

Beschreibung

array **imap_scanmailbox** (int imap_stream, string ref, string pat, string content)

Diese Funktion arbeitet ähnlich wie [imap_listmailbox\(\)](#). Sie giebt aber nur die Namen der Postfächer zurück, in denen irgendwo der Text *content* vorkommt.

imap_mailboxmsginfo

(PHP3 >= 3.0.2, PHP4)

imap_mailboxmsginfo -- Liefert Informationen über das aktuelle Postfach

Beschreibung

object **imap_mailboxmsginfo** (int imap_stream)

Diese Funktion liefert dieselben Informationen wie [imap_status\(\)](#) sowie zusätzlich die Größe des Postfachs in Bytes. Diese wird aus der Summe der Größe der einzelnen Nachrichten gebildet, deshalb benötigt diese Funktion mehr Zeit zur Ausführung als [imap_status\(\)](#).

Tabelle 1 Felder des Rückgabe-Objekts

Date	Datum der letzten Änderung
Driver	Driver
Mailbox	Name des Postfachs
Nmsgs	Anzahl der Nachrichten
Recent	Anzahl neuer Nachrichten
Unread	Anzahl ungelesener Nachrichten
Size	Größe des Postfachs in Byte

Beispiel 1 imap_mailboxmsginfo() Beispiel

```
$mbox = imap_open ("{your.imap.host}INBOX", "username", "password")
|| 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 "Size: "        . $check->Size       . "<br>\n";
} else {
```

```
    print "imap_check() failed: " . imap_lasterror() . "<br>\n";  
}  
imap_close ($mbox);
```

[Zurück](#)

imap_scanmailbox

[Anfang](#)

[Hoch](#)

[Vor](#)

imap_rfc822_write_address

imap_rfc822_write_address

(PHP3 >= 3.0.2, PHP4)

imap_rfc822_write_address -- Bildet aus Realnamen, Postfach und Server eine korekt formatierte Mail-Adresse

Beschreibung

string **imap_rfc822_write_address** (string mailbox, string host, string personal)

Bildet aus Realnamen, Postfach und Server eine korekt formatierte Mail-Adresse gemäß [RFC822](#).

Beispiel 1 Imap_rfc822_write_address() Beispiel

```
print imap_rfc822_write_address ("hartmut", "cvs.php.net",  
                                "Hartmut Holzgraefe");
```

imap_rfc822_parse_adrlist

(PHP3 >= 3.0.2, PHP4)

imap_rfc822_parse_adrlist -- Parsen eines Adress-Strings

Beschreibung

array **imap_rfc822_parse_adrlist** (string address, string default_host)

Diese Funktion zerlegt einen Adress-String gemäß [RFC822](#) und liefert ein Array von Objekten mit einem Eintrag je erkannter Adresse. Die Address-Objekte bestehen dabei jeweils aus den folgenden vier String-Elementen:

- mailbox - Name des Postfachs (Username)
- host - Hostname
- personal - Realer Name des Empfängers
- adl - at domain source route

Beispiel 1 Imap_rfc822_parse_adrlist() Beispiel

```
$address_string = "Hartmut Holzgraefe <hartmut@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_setflag_full

(PHP3 >= 3.0.3, PHP4)

imap_setflag_full -- Setzt Flags einer Nachricht

Beschreibung

string **imap_setflag_full** (int stream, string sequence, string flag, string options)

Diese Funktion setzt die übergebenen Flags für eine oder mehrere Nachrichten.

Mögliche Flags sind "\\Seen", "\\Answered", "\\Flagged", "\\Deleted", "\\Draft", und "\\Recent" (siehe auch [RFC2060](#))

Enthält *options* den Wert ST_UID, so werden die übergebenen Nachrichtennummern als UIDs anstelle von Indizes interpretiert.

Beispiel 1 imap_setflag_full() Beispiel

```
$mbox = imap_open ("{your.imap.host:143}", "username", "password")
|| die ("can't connect: " . imap_last_error());

$status = imap_setflag_full ($mbox, "2,5", "\\Seen \\Flagged");
if (!$status) echo imap_last_error();

imap_close ($mbox);
```

imap_clearflag_full

(PHP3 >= 3.0.3, PHP4)

imap_clearflag_full -- Löscht Flags einer Nachricht

Beschreibung

string **imap_clearflag_full** (int stream, string sequence, string flag, string options)

Diese Funktion löscht die übergebenen Flags für eine oder mehrere Nachrichten.

Mögliche Flags sind "\\Seen", "\\Answered", "\\Flagged", "\\Deleted", "\\Draft", und "\\Recent" (siehe auch [RFC2060](#))

Enthält *options* den Wert ST_UID, so werden die übergebenen Nachrichtennummern als UIDs anstelle von Indizes interpretiert.

imap_sort

(PHP3 >= 3.0.3, PHP4)

imap_sort -- Sortiert Nachrichten eines Postfachs

Beschreibung

array **imap_sort** (int stream, int criteria, int reverse, int options)

Diese Funktion liefert die Indizes der Nachrichten eines Postfachs als sortiertes Array zurück. Als Sortier-Kriterium kann im Parameter *criteria* genau einer der folgenden Werte angegeben werden

- SORTDATE Versanddatum der Nachricht
- SORTARRIVAL Empfangsdatum der Nachricht
- SORTFROM Benutzername des ersten From: Eintrags
- SORTSUBJECT Betreff der Nachricht
- SORTTO Benutzername des ersten To: Eintrags
- SORTCC Benutzername des ersten CC: Eintrags
- SORTSIZE Größe der Nachricht in Bytes

Ist *reverse* ungleich 0, so werden die Nachrichten in umgekehrter Reihenfolge übergeben.

Die Bitmaske *options* kann aus folgenden werten zusammengesetzt werden:

- SE_UID Es werden UIDs anstelle von Indizes zurückgegeben
- SE_NOPREFETCH Es werden keine Nachrichten im Cache gehalten

imap_fetchheader

(PHP3 >= 3.0.3, PHP4)

imap_fetchheader -- Liefert den Kopf einer Nachricht

Beschreibung

string**imap_fetchheader** (int imap_stream, int msgno, int flags)

Diese Funktion liefert den vollständigen, ungefilterten und gemäß [RFC 822](#) formatierten Kopf einer Nachricht als String zurück.

Die Bitmaske *flags* kann aus folgenden Werten zusammengesetzt werden:

- FT_UID *msgno* enthält eine UID
- FT_INTERNAL Der Kopf wird im "internen" Format, d.h. ohne jegliche Konvertierung von Zeilenenden zurückgegeben
- FT_PREFETCHTEXT Der Körper der Nachricht wird ebenfalls angefordert, hierdurch wird ein zus. Zugriff vermieden, wenn z.B. für eine Save-Operation der gesamte Text einer Nachricht benötigt wird

imap_uid

(PHP3 >= 3.0.3, PHP4)

imap_uid -- Liefert die UID zu einem gegebenen Nachrichten-Index

Beschreibung

int **imap_uid** (int imap_stream, int msgno)

Diese Funktion ermittelt die UID zu einem gegebenen Nachrichten-Index. Während die Nachrichten-Indices einfach von eins an aufsteigend durchnummeriert sind und sich z.B. bei Löschoperationen ändern können, ist die UID einer Nachricht innerhalb eines Postfachs unveränderbar eindeutig. (Siehe auch [imap_msgno\(\)](#)).

imap_msgno

(PHP3 >= 3.0.3, PHP4)

imap_msgno -- Liefert den aktuellen Nachrichten-Index zu einer UID

Beschreibung

int **imap_msgno** (int imap_stream, int uid)

Diese Funktion liefert den aktuellen Index einer Nachricht innerhalb eines Postfachs aus der eindeutigen UID. Sie ist somit die Umkehrung der Funktion [imap_uid\(\)](#).

imap_search

(PHP3 >= 3.0.12, PHP4 >= 4.0b4)

imap_search -- Sucht Nachrichten, die den übergebenen Suchkriterien entsprechen

Beschreibung

array **imap_search** (int imap_stream, string criteria, int flags)

Diese Funktion durchsucht die Nachrichten des aktuellen Postfachs und liefert ein Array mit den Nummern der Nachrichten, die mit den angegebenen Kriterien übereinstimmen. Der Parameter *criteria* enthält einen String, der sich aus den folgenden Suchbegriffen zusammensetzen kann

- ALL - liefert alle Nachrichten, die den restlichen Kriterien entsprechen
- ANSWERED - das \ANSWERED Flag der Nachricht ist gesetzt
- BCC "string" - Das Bcc: Feld der Nachricht enthält "string"
- BEFORE "date" - Das Date: Feld der Nachricht enthält ein Datum, das vor dem in "date" angegebenen liegt
- BODY "string" - Im Inhalt der Nachricht ist "string" enthalten
- CC "string" - Das Cc: Feld der Nachricht enthält "string"
- DELETED - Die Nachricht ist zum Löschen vorgemerkt
- FLAGGED - Das \FLAGGED (wichtig bzw. dringend) Flag der Nachricht ist gesetzt
- FROM "string" - Das From: Feld der Nachricht enthält "string"
- KEYWORD "string" - Das Feld Keyword: enthält "string"
- NEW - die Nachricht ist neu
- OLD - die Nachricht ist nicht neu
- ON "date" - Das Datum in Date: entspricht "date"
- RECENT - Das \RECENT Flag ist gesetzt
- SEEN - Das \SEEN Flag ist gesetzt, die Nachricht wurde bereits einmal abgerufen
- SINCE "date" - Das Datum "date" liegt vor dem im Date: Feld der Nachricht

- SUBJECT "string" - "string" ist Teil des Betreffs der Nachricht
- TEXT "string" - "string" ist Teil der Nachricht (Kopf oder Inhalt)
- TO "string" - Das To: Feld der Nachricht enthält "string"
- UNANSWERED - Die Nachricht wurde nicht beantwortet, das \Answered Flag ist nicht gesetzt
- UNDELETED - Die Nachricht ist nicht zum Löschen vorgemerkt
- UNFLAGGED - Die Nachricht ist nicht als 'wichtig' markiert
- UNKEYWORD "string" - "string" ist nicht Schlüsselwort der Nachricht
- UNSEEN - Die Nachricht wurde bisher nicht gelesen

Argumente, die Leerzeichen enthalten, müssen dabei in Anführungszeichen eingebettet werden.

Um zum Beispiel alle unbeantworteten Nachrichten von Mama zu finden übergeben Sie als *criteria*: "UNANSWERED FROM Mama". Die Suche scheint nicht zwischen Groß- und Kleinschreibung zu unterscheiden. Die Liste der Suchbegriffe entstand nach Inspektion der UW c-client Sourcen und ist vermutlich nicht vollständig (siehe auch [RFC 2060](#), Abschnitt 6.4.4).

Enthält der optionale Parameter *flags* den Wert SE_UID, so werden UIDs anstelle von laufenden Nachrichtennummern zurückgegeben.

[Zurück](#)[Anfang](#)[Vor](#)

imap_msgno

[Hoch](#)

imap_last_error

imap_last_error

(PHP3 >= 3.0.12, PHP4 >= 4.0b4)

imap_last_error -- Diese Funktion liefert die zuletzt während dieses Seitenzugriffs aufgetretene Fehlermeldung (falls vorhanden)

Beschreibung

string **imap_last_error** (void)

Diese Funktion liefert den vollständigen Text der zuletzt aufgetretenen Fehlermeldung dieses Requests. Der Zustand des Fehlerstacks wird durch diese Funktion nicht verändert, weitere Aufrufe von **imap_last_error()** liefern somit weiter dasselbe Ergebnis, solange keine weiteren Fehler aufgetreten sind.

imap_errors

(PHP3 >= 3.0.12, PHP4 >= 4.0b4)

imap_errors -- Diese Funktion liefert alle bisher aufgetretenen Fehlermeldungen

Beschreibung

array **imap_errors** (void)

Diese Funktion liefert in Form eines String-Arrays alle seit Beginn des Requests bzw. seit dem letzten Aufruf von **imap_errors()** aufgetretenen Fehlermeldung. Der Fehler-Stack wird dabei gelöscht, so daß folgende Aufrufe von **imap_errors()** bzw. **imap_last_errors()** bis zum nächsten aufgetretenen Fehler kein Ergebnis liefern.

imap_alerts

(PHP3 >= 3.0.12, PHP4 >= 4.0b4)

`imap_alerts` -- Diese Funktion liefert alle alert-Meldungen zurück, die während der Ausführung dieses Requests bzw. seit dem letzten Reset des alert-Stacks aufgetreten sind.

Beschreibung

array `imap_alerts` (void)

Diese Funktion liefert in Form eines String-Arrays alle IMAP alert-Medlungen, die seit Beginn dieses Requests bzw. dem letzten Aufruf der Funktion aufgetreten sind. Durch den Aufruf der Funktion wird der alert-Stack gelöscht, so daß weitere Aufrufe zunächst bis zum Auftreten eines neuen alerts kein Ergebnis liefern.

Die IMAP-Spezifikationen verlangen, daß diese Meldungen an den Benutzer weitergegeben werden.

imap_status

(PHP3 >= 3.0.4, PHP4)

imap_status -- Liefert ausgewählte Statusinformationen zum angegebenen Postfach

Beschreibung

object **imap_status** (int imap_stream, string mailbox, int options)

Diese Funktion liefert Statusinformationen zum mit *mailbox* angegebenen Postfach. Im der Bitmaske *options* können dabei folgende Funktionen ausgewählt werden, die den Inhalt des zurückgegebenen Objekts beeinflussen:

- SA_MESSAGES - das Feld messages wird mit der Anzahl der enthaltenen Nachrichten gefüllt
- SA_RECENT - das Feld recent wird mit der Anzahl der neuen Nachrichten gefüllt
- SA_UNSEEN - das Feld unseen wird mit der Anzahl der noch nicht abgerufenen Nachrichten gefüllt
- SA_UIDNEXT - das Feld uidnext wird mit der UID gefüllt, die die nächste in diesem Postfach abgelegte Nachricht erhalten wird
- SA_UIDVALIDITY - das Feld uidvalidity wird mit einer Konstanten gefüllt, die sich immer dann verändern wird, wenn die UIDs der Nachrichten im Postfach sich vermutlich verändert haben
- SA_ALL - es werden alle beschriebenen Felder gefüllt

Status->flags wird ebenfalls gesetzt und enthält eine Bitmaske aus den oben genannten Werten, die die tatsächlich zurückgegebenen Informationen beschreibt.

Beispiel 1 Imap_status() Beispiel

```
$mbox = imap_open ("{your.imap.host}", "username", "password", OP_HALFOPEN)
|| 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:    " . $status->uidvalidity        ) . "<br>\n";
} else {
    print "imap_status failed: " . imap_lasterror() . "\n";
}
imap_close ($mbox);
```

[Zurück](#)

imap_alerts

[Anfang](#)

[Hoch](#)

[Vor](#)

imap_utf7_decode

imap_utf7_decode

(PHP3 >= 3.0.15, PHP4 >= 4.0b4)

imap_utf7_decode -- Dekodiert einem String im modifizierten UTF-7 Format.

Beschreibung

string **imap_utf7_decode** (string text)

Dekodiert den String *text* aus dem modifizierten UTF-7 Format in 8Bit-Text.

Liefert den dekodierten Text oder false, wenn die Eingabe nicht gemäß modifiziertem UTF-7 formatiert war. Diese Funktion wird benötigt, um Postfachnamen zu decodieren, die internationale Sonderzeichen, wie z.B. unsere Umlaute, enthalten.

(siehe [RFC2152](#) für UTF-7 und [RFC2060](#), Abschnitt 5.1.3 für modifiziertes UTF-7)

imap_utf7_encode

(PHP3 >= 3.0.15, PHP4 >= 4.0b4)

imap_utf7_encode -- Kodiert Text im modifizieren UTF-7 Format

Beschreibung

string **imap_utf7_encode** (string data)

Konvertiert den in *data* übergebenen Text in das modifizierte UTF-7 Format. Diese Funktion wird benötigt, um Postfachnamen darzustellen, die internationale Sonderzeichen, wie z.B. unsere Umlaute, enthalten.

(siehe [RFC2152](#) für UTF-7 und [RFC2060](#), Abschnitt 5.1.3 für modifiziertes UTF-7)

imap_utf8

(PHP3 >= 3.0.13, PHP4 >= 4.0RC1)

imap_utf8 -- Konvertiert Text zu UTF8

Beschreibung

string **imap_utf8** (string text)

Konvertiert den übergebenen String in das UTF8-Format gemäß [RFC2044](#).

imap_fetch_overview

(PHP3 >= 3.0.4, PHP4)

imap_fetch_overview -- Liefert einen Auszug aus den Header-Feldern von Nachrichten

Beschreibung

array **imap_fetch_overview** (int imap_stream, string sequence [, int flags])

Diese Funktion liefert ausgewählte Headerfelder der in *sequence* übergebenen Nachrichten und liefert sie in einem Array mit je einem Objekt pro Nachricht zurück. *sequence* enthält eine Sequenz von Nachrichten-Indizes oder, bei gesetztem FT_UID in *flags*, von UIDs. Die einzelnen Objekte des zurückgegeben Arrays enthalten dabei folgende Informationen:

- subject - Betreff
- from - Absender
- date - Sendedatum
- message_id - Message-ID
- references - bezieht sich auf Message-ID
- size - Größe in Byte
- uid - UID der Nachricht im Postfach
- msgno - Index der Nachricht im Postfach
- recent - Flag gesetzt
- flagged - Flag gesetzt
- answered - Flag gesetzt
- deleted - Flag gesetzt
- seen - Flag gesetzt
- draft - Flag gesetzt

Beispiel 1 Imap_fetch_overview() Beispiel

```
$mbox = imap_open ("{your.imap.host:143}", "username", "password")
|| 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);
```

[Zurück](#)[imap_utf8](#)[Anfang](#)[Hoch](#)[Vor](#)[imap_mime_header_decode](#)

imap_mime_header_decode

(PHP3 CVS only, PHP4 >= 4.0RC1)

imap_mime_header_decode -- Decode MIME header elements

Beschreibung

array **imap_header_decode** (string text)

imap_mime_header_decode() decodiert MIME Headerfelder, die mehr als nur reinen ASCII-Text enthalten und gemäß [RFC2047](#) codiert wurden. Die decodierten Teilstücke werden in einem Array von Objekten zurückgegeben, wobei jedes Objekt die Felder "text" für den decodierten Teilttext und "charset" für den zugehörigen Zeichensatz enthält. Nichtcodierte Teile des Headerfeldes werden dabei durch den Wert "default" beschrieben.

Im folgenden Beispiel besteht das Headerfeld aus zwei Teilstücken, wobei das erste nach ISO-8859-1 codiert ist:

Beispiel 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";
}
```

imap_mail_compose

(PHP3 >= 3.0.5, PHP4)

imap_mail_compose -- Erzeugt eine MIME-Nachricht aus gegebenen Header- und Body-Teilen

Beschreibung

string **imap_mail_compose** (array envelope, array body)

[Zurück](#)[Anfang](#)[Vor](#)

imap_mime_header_decode

[Hoch](#)

imap_mail

imap_mail

(PHP3 >= 3.0.14, PHP4 >= 4.0b4)

imap_mail -- Versendet eine Email

Beschreibung

string **imap_mail** (string to, string subject, string message [, string additional_headers [, string cc [, string bcc [, string rpath]]]])

This function is currently not available for PHP4 on WIN32.

XXVII. Informix functions

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.

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.

The configure script will autodetect the libraries and include directories, if you run "configure --with_informix=yes". You can override this detection by specifying "IFX_LIBDIR", "IFX_LIBS" and "IFX_INCDIR" in the environment. The configure script will also try to detect your Informix server version. It will set the "HAVE_IFX_IUS" conditional compilation variable if your Informix version ≥ 9.00 .

Runtime considerations: 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.

Inhaltsverzeichnis

- [ifx_connect](#) Open Informix server connection
- [ifx_pconnect](#) Open persistent Informix connection
- [ifx_close](#) Close Informix connection
- [ifx_query](#) Send Informix query
- [ifx_prepare](#) Prepare an SQL-statement for execution
- [ifx_do](#) Execute a previously prepared SQL-statement
- [ifx_error](#) Returns error code of last Informix call
- [ifx_errormsg](#) Returns error message of last Informix call
- [ifx_affected_rows](#) Get number of rows affected by a query
- [ifx_getsqlca](#) Get the contents of sqlca.sqlerrd[0..5] after a query
- [ifx_fetch_row](#) Get row as enumerated array
- [ifx_htmltbl_result](#) Formats all rows of a query into a HTML table
- [ifx_fieldtypes](#) List of Informix SQL fields
- [ifx_fieldproperties](#) List of SQL fieldproperties
- [ifx_num_fields](#) Returns the number of columns in the query
- [ifx_num_rows](#) Count the rows already fetched a query
- [ifx_free_result](#) Releases resources for the query
- [ifx_create_char](#) Creates an char object
- [ifx_free_char](#) Deletes the char object
- [ifx_update_char](#) Updates the content of the char object
- [ifx_get_char](#) Return the content of the char object
- [ifx_create_blob](#) Creates an blob object
- [ifx_copy_blob](#) Duplicates the given blob object
- [ifx_free_blob](#) Deletes the blob object
- [ifx_get_blob](#) Return the content of a blob object
- [ifx_update_blob](#) Updates the content of the blob object
- [ifx_blobinfile_mode](#) Set the default blob mode for all select queries
- [ifx_textasvarchar](#) Set the default text mode
- [ifx_byteasvarchar](#) Set the default byte mode
- [ifx_nullformat](#) Sets the default return value on a fetch row
- [ifxus_create_slob](#) Creates an slob object and opens it
- [ifx_free_slob](#) Deletes the slob object

ifxus_close_slob	Deletes the slob object
ifxus_open_slob	Opens an slob object
ifxus_tell_slob	Returns the current file or seek position
ifxus_seek_slob	Sets the current file or seek position
ifxus_read_slob	Reads nbytes of the slob object
ifxus_write_slob	Writes a string into the slob object

Zurück	Anfang	Vor
imap_mail	Hoch	ifx_connect

ifx_connect

(PHP3 >= 3.0.3, PHP4)

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\(\)](#).

Beispiel 1 Connect to a Informix database

```
$conn_id = ifx_pconnect ("mydb@ol_srv1", "imyself", "mypassword");
```


ifx_pconnect

(PHP3 >= 3.0.3, PHP4)

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_close

(PHP3 >= 3.0.3, PHP4)

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\(\)](#).

Beispiel 1 Closing a Informix connection

```
$conn_id = ifx_connect ("mydb@ol_srv", "itsme", "mypassword");  
... some queries and stuff ...  
ifx_close($conn_id);
```

ifx_query

(PHP3 >= 3.0.3, PHP4)

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. 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 [ifx_connect\(\)](#) was called, and use it.

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. If 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\(\)](#).

Beispiel 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);
```

Beispiel 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,'HR0',?,?)";
$res_id = ifx_query($query, $conn_id, $blobidarray);
if (! $res_id) {
    ... error ...
}
                // free result id
ifx_free_result($res_id);
```

[Zurück](#)

ifx_close

[Anfang](#)[Hoch](#)[Vor](#)

ifx_prepare

ifx_prepare

(PHP3 >= 3.0.4, PHP4)

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_do

(PHP3 >= 3.0.4, PHP4)

ifx_do -- Execute a previously prepared SQL-statement

Description

int **ifx_do** (int result_id)

Returns TRUE on success, FALSE on error.

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\(\)](#). There is a example.

ifx_error

(PHP3 >= 3.0.3, PHP4)

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

(PHP3 >= 3.0.4, PHP4)

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<br>", ifx_errormsg(-201));
```


ifx_affected_rows

(PHP3 >= 3.0.3, PHP4)

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\(\)](#)

Beispiel 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_getsqlca

(PHP3 >= 3.0.8, PHP4)

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.

Beispiel 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_fetch_row

(PHP3 >= 3.0.3, PHP4)

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.

Beispiel 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);
```

[Zurück](#)[ifx_getsqlca](#)[Anfang](#)[Hoch](#)[Vor](#)[ifx_htmltbl_result](#)

ifx_htmltbl_result

(PHP3 >= 3.0.3, PHP4)

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

Beispiel 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_fieldtypes

(PHP3 >= 3.0.3, PHP4)

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.

Beispiel 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_fieldproperties

(PHP3 >= 3.0.3, PHP4)

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".

Beispiel 1 Informix SQL fieldproperties

```
$properties = ifx_fielddtypes ($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_num_fields

(PHP3 >= 3.0.3, PHP4)

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

(PHP3 >= 3.0.3, PHP4)

ifx_num_rows -- Count the rows already fetched 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_free_result

(PHP3 >= 3.0.3, PHP4)

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_create_char

(PHP3 >= 3.0.6, PHP4)

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_free_char

(PHP3 >= 3.0.6, PHP4)

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 FALSE on error otherwise TRUE.

[Zurück](#)[Anfang](#)[Vor](#)

ifx_create_char

[Hoch](#)

ifx_update_char

ifx_update_char

(PHP3 >= 3.0.6, PHP4)

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 FALSE on error otherwise TRUE.

ifx_get_char

(PHP3 >= 3.0.6, PHP4)

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_create_blob

(PHP3 >= 3.0.4, PHP4)

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_copy_blob

(PHP3 >= 3.0.4, PHP4)

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_free_blob

(PHP3 >= 3.0.4, PHP4)

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 FALSE on error otherwise TRUE.

[Zurück](#)[Anfang](#)[Vor](#)

ifx_copy_blob

[Hoch](#)

ifx_get_blob

ifx_get_blob

(PHP3 >= 3.0.4, PHP4)

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*.

[Zurück](#)[Anfang](#)[Vor](#)

ifx_free_blob

[Hoch](#)

ifx_update_blob

ifx_update_blob

(PHP3 >= 3.0.4, PHP4)

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 FALSE on error otherwise TRUE.

ifx_blobinfile_mode

(PHP3 >= 3.0.4, PHP4)

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_textasvarchar

(PHP3 >= 3.0.4, PHP4)

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_byteasvarchar

(PHP3 >= 3.0.4, PHP4)

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_nullformat

(PHP3 >= 3.0.4, PHP4)

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".

ifxus_create_slob

(PHP3 >= 3.0.4, PHP4)

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.

ifx_free_slob

(unknown)

ifx_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 FALSE on error otherwise TRUE.

ifxus_close_slob

(PHP3 >= 3.0.4, PHP4)

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*. Return FALSE on error otherwise TRUE.

[Zurück](#)[Anfang](#)[Vor](#)

ifx_free_slob

[Hoch](#)

ifxus_open_slob

ifxus_open_slob

(PHP3 >= 3.0.4, PHP4)

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.

[Zurück](#)[Anfang](#)[Vor](#)[ifxus_close_slob](#)[Hoch](#)[ifxus_tell_slob](#)

ifxus_tell_slob

(PHP3 >= 3.0.4, PHP4)

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_seek_slob

(PHP3 >= 3.0.4, PHP4)

ifxus_seek_slob -- Sets the current file or seek position

Description

int **ifxus_seek_blob** (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.

[Zurück](#)[Anfang](#)[Vor](#)

ifxus_tell_slob

[Hoch](#)

ifxus_read_slob

ifxus_read_slob

(PHP3 >= 3.0.4, PHP4)

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 zu read.
Return FALSE on error otherwise the string.

ifxus_write_slob

(PHP3 >= 3.0.4, PHP4)

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.

XXVIII. InterBase-Funktionen

InterBase ist eine bekannte Datenbank, die von Borland/Inprise stammt. Weitere Informationen zu InterBase finden sie unter <http://www.interbase.com/>. Wichtig: InterBase befindet sich gerade auf dem Weg zum Open Source!

Anmerkung: Umfassende Unterstützung für InterBase 6 wurde in PHP 4.0 eingeführt.

Inhaltsverzeichnis

[ibase_connect](#) Öffnet eine Verbindung zu einer InterBase-Datenbank

[ibase_pconnect](#) Erzeugt eine permanente Verbindung zu einer InterBase-Datenbank

[ibase_close](#) Schließt die Verbindung zu einer InterBase-Datenbank

[ibase_query](#) Führt eine Abfrage (Query) auf eine InterBase-DB aus

[ibase_fetch_row](#) Liest einen Datensatz aus einer InterBase-Datenbank

[ibase_fetch_object](#) Liest einen Datensatz einer InterBase-Datenbank als Objekt ein

[ibase_free_result](#) Gibt den Speicher eines Abfrage-Ergebnisses frei

[ibase_prepare](#) Vorbereitung einer Abfrage für den folgenden Gebrauch von Parameter-Platzhaltern und für die eigentliche Ausführung.

[ibase_execute](#) Ausführen einer vorbereiteten Abfrage

[ibase_free_query](#) Gibt den Speicher einer vorbereiteten Abfrage wieder frei

[ibase_timefmt](#) Bestimmt das Format von Zeitstempel-, Datums- und Zeit-Feldern, die von einer Abfrage zurück gegeben werden

[ibase_num_fields](#) Ermittelt die Anzahl der Felder einer Ergebnis-Liste

ibase_connect

(PHP3 >= 3.0.6, PHP4)

ibase_connect -- Öffnet eine Verbindung zu einer InterBase-Datenbank

Beschreibung:

```
int ibase_connect (string database [, string username [, string  
password [, string charset [, int buffers [, int dialect [, string  
role]]]]])
```

Stellt eine Verbindung zu einem InterBase-Server her. Das *database*-Argument muss ein gültiger Pfad zu einer bestehenden Datenbank des Servers sein. Ist der DB-Server nicht lokal, muss ihm - in Abhängigkeit vom benutzten Protokoll - ein 'hostname:' (bei TCP/IP), '//hostname/' (bei NetBEUI) oder 'hostname@' (bei IPX/SPX) vorangestellt werden. Die Parameter *username* und *password* können auch über die PHP-Konfigurations-Anweisungen `ibase.default_user` und `ibase.default_password` gesetzt werden. *Charset* ist der reguläre Zeichensatz der Datenbank. Mit *buffers* geben sie die Anzahl der Datenbank-Cache-Buffer (serverseitig) an. Ist der Wert 0 oder nicht angegeben, benutzt der Server seine Default-Werte. Mittels *dialect* bestimmen sie den Default-Wert des anzuwendenden SQL-Dialekts für alle Anweisungen während einer Datenbank-Verbindung (default ist der Client-seitig höchst verfügbare Dialekt).

Wenn **ibase_connect()** mit den selben Argumenten erneut aufgerufen wird, wird keine neue Verbindung hergestellt, sondern die ID der bereits geöffneten Verbindung zurück gegeben. Die Verbindung zum InterBase-Server wird am Ende des jeweiligen Skripts automatisch beendet, es sei denn, es wird vorher die Funktion [ibase_close\(\)](#) aufgerufen.

Beispiel 1 Ibase_connect()-Beispiel:

```
<?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);
?>
```

Anmerkung: *Buffers* existiert seit PHP4-RC2.

Anmerkung: *Dialect* wurde in PHP4-RC2 hinzugefügt. Er funktioniert ab der InterBase-Version 6.

Anmerkung: *Role* existiert seit PHP4-RC2. Er funktioniert ab der InterBase-Version 5.

Siehe auch: [ibase_pconnect\(\)](#).

[Zurück](#)[InterBase-Funktionen](#)[Anfang](#)[Hoch](#)[Vor](#)[ibase_pconnect](#)

ibase_pconnect

(PHP3 >= 3.0.6, PHP4)

ibase_pconnect -- Erzeugt eine permanente Verbindung zu einer InterBase-Datenbank

Beschreibung:

```
int ibase_connect (string database [, string username [, string  
password [, string charset [, string role]]]])
```

Ibase_pconnect() agiert ähnlich der Funktion [ibase_connect\(\)](#) (mit zwei Ausnahmen). Erstens wird beim Verbindungs-Aufbau versucht, eine bereits bestehende permanente Verbindung zu finden, die mit den selben Parametern aufgebaut worden ist. Wird eine solche gefunden, wird statt eines neuen Verbindungs-Aufbaus die ID der bestehenden Verbindung zurück gegeben. Zweitens wird beim Beenden des Skripts die Verbindung nicht beendet. Statt dessen wird die Verbindung für den weiteren Gebrauch beibehalten ([ibase_close\(\)](#) beendet keine Verbindung, die mittels **ibase_pconnect()** erzeugt wurde). Diese Art der Verbindung wird deshalb auch 'persistent' (permanent) genannt.

Siehe auch [ibase_connect\(\)](#) hinsichtlich der Bedeutung der dieser Funktion übergebenen Parameter. Es sind die gleichen.

ibase_close

(PHP3 >= 3.0.6, PHP4)

ibase_close -- Schließt die Verbindung zu einer InterBase-Datenbank

Beschreibung:

int **ibase_close** ([int connection_id])

Beendet die Verbindung zu einer InterBase-DB, deren Verbindungs-ID mit der von [ibase_connect\(\)](#) zurück gegebenen ID überein stimmt. Wird die Verbindungs-ID weggelassen, wird die zuletzt hergestellte Verbindung beendet. Vorgenommene Transaktionen der Verbindung werden übertragen, andere werden zurückgenommen.

ibase_query

(PHP3 >= 3.0.6, PHP4)

ibase_query -- Führt eine Abfrage (Query) auf eine InterBase-DB aus

Beschreibung:

```
int ibase_query ([int link_identifizier, string query [, int  
bind_args]])
```

Erzeugt eine Anfrage auf eine InterBase-DB, deren Ergebnis mit [ibase_fetch_row\(\)](#), [ibase_fetch_object\(\)](#), [ibase_free_result\(\)](#) und [ibase_free_query\(\)](#) weiter verarbeitet werden kann.

Anmerkung: Obwohl diese Funktion auch die Verwendung von Platzhaltern unterstützt, ist ihr keine besondere Bedeutung beizumessen. Für reale Abfragen und Beispiele sehen sie unter [ibase_prepare\(\)](#) und [ibase_execute\(\)](#) nach.

ibase_fetch_row

(PHP3 >= 3.0.6, PHP4)

ibase_fetch_row -- Liest einen Datensatz aus einer InterBase-Datenbank

Beschreibung:

array **ibase_fetch_row** (int result_identifizier)

Gibt den nächsten Datensatz des per [ibase_query\(\)](#) gewonnenen Zeigers in einem Array zurück.

ibase_fetch_object

(PHP3 >= 3.0.7, PHP4 >= 4.0RC1)

`ibase_fetch_object` -- Liest einen Datensatz einer InterBase-Datenbank als Objekt ein

Beschreibung:

object **ibase_fetch_object** (int *result_id*)

Liest die Daten der per [ibase_query\(\)](#) oder [ibase_execute\(\)](#) gewonnenen *result_id* in ein Pseudo-Objekt ein.

```
<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);
?>
```

Siehe auch [ibase_fetch_row\(\)](#).

ibase_free_result

(PHP3 >= 3.0.6, PHP4)

ibase_free_result -- Gibt den Speicher eines Abfrage-Ergebnisses frei

Beschreibung:

int **ibase_free_result** (int result_identifizier)

Gibt den durch den Aufruf von [ibase_query\(\)](#) belegten Speicher wieder frei.

ibase_prepare

(PHP3 >= 3.0.6, PHP4)

ibase_prepare -- Vorbereitung einer Abfrage für den folgenden Gebrauch von Parameter-Platzhaltern und für die eigentliche Ausführung.

Beschreibung:

```
int ibase_prepare ([int link_identifizier, string query])
```

Vorbereitung einer Abfrage für den folgenden Gebrauch von Parameter-Platzhaltern und für die eigentliche Ausführung (per [ibase_execute\(\)](#)).

ibase_execute

(PHP3 >= 3.0.6, PHP4)

ibase_execute -- Ausführen einer vorbereiteten Abfrage

Beschreibung:

int **ibase_execute** (int query [, int bind_args])

Führt eine vorher per [ibase_prepare\(\)](#) vorbereitete Abfrage aus. Dies ist effizienter als der Gebrauch von [ibase_query\(\)](#), sofern sie mehrere ähnliche Abfragen durchführen wollen, bei denen sich immer nur einige Parameter ändern.

```
<?php
    $updates = array(
        1 => 'Erik',
        5 => 'Phillip',
        7 => 'Ludwig'
    );

    $query = ibase_prepare("UPDATE FOO SET BAR = ? WHERE BAZ = ?");

    while (list($baz, $bar) = each($updates)) {
        ibase_execute($query, $bar, $baz);
    }
?>
```

ibase_free_query

(PHP3 >= 3.0.6, PHP4)

ibase_free_query -- Gibt den Speicher einer vorbereiteten Abfrage wieder frei

Beschreibung:

```
int ibase_free_query (int query)
```

Gibt den Speicher einer vorbereiteten Abfrage (siehe [ibase_prepare\(\)](#)) wieder frei.

ibase_timefmt

(PHP3 >= 3.0.6, PHP4)

`ibase_timefmt` -- Bestimmt das Format von Zeitstempel-, Datums- und Zeit-Feldern, die von einer Abfrage zurück gegeben werden

Beschreibung:

`int ibase_timefmt (string format [, int columntype])`

Bestimmt das Format von Zeitstempel-, Datums- und Zeit-Feldern, die durch eine Abfrage zurück gegeben werden. Intern werden diese Felder durch die C-Funktion `strftime()` formatiert. Deshalb finden sie in der dazu bestehenden Dokumentation genaue Hinweise über das entsprechende String-Format. Der optionale Parameter *columntype* kann eine der Konstanten `IBASE_TIMESTAMP`, `IBASE_DATE` oder `IBASE_TIME` sein. Fehlt er, wird per default von `IBASE_TIMESTAMP` ausgegangen (wegen der Abwärts-Kompatibilität).

```
<?php
// InterBase 6 Zeit-Typ-Felder werden im
// Format '05 Stunden 37 Minuten' zurück gegeben.
ibase_timefmt("%H Stunden %M Minuten", IBASE_TIME);
?>
```

Sie können über die PHP-Konfigurations-Anweisungen `ibase.timestampformat`, `ibase.dateformat` und `ibase.timeformat` den default-Wert anpassen.

Anmerkung: *Columntype* wurde in PHP 4.0 hinzu gefügt. Er hat erst ab InterBase-Version 6 eine Bedeutung.

Anmerkung: Eine Abwärts-Inkompatibilität wurde in PHP 4.0 behoben, indem die Konfigurations-Anweisung `ibase.timeformat` umbenannt wurde in `ibase.timestampformat` und die Anweisungen `ibase.dateformat` und `ibase.timeformat` hinzu gefügt wurden. Dadurch entsprechen die Namen eher ihrer eigentlichen Funktionalität.

ibase_num_fields

(PHP3 >= 3.0.7, PHP4 >= 4.0RC1)

ibase_num_fields -- Ermittelt die Anzahl der Felder einer Ergebnis-Liste

Beschreibung:

`int ibase_num_fields (int result_id)`

Der zurück gegebene Integer-Wert stellt die Anzahl der Felder einer Ergebnis-Liste dar.

```
<?php
$dbh = ibase_connect ($host, $username, $password);
$stmt = 'SELECT * FROM tblname';
$stmt = ibase_query ($dbh, $stmt);

if (ibase_num_fields($sth) > 0) {
    while ($row = ibase_fetch_object ($sth)) {
        print $row->email . "\n";
    }
} else {
    die ("Keine Ergebnisse für ihre Abfrage");
}

ibase_close ($dbh);
?>
```

Siehe auch: **ibase_field_info()**.

Anmerkung: Ibase_num_fields() funktioniert z.Zt. nicht bei PHP 4.

XXIX. LDAP functions

Introduction to LDAP

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.

Complete code example

Retrieve information for all entries where the surname starts with "S" from a directory server, displaying an extract with name and email address.

Beispiel 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

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. You will also need to recompile PHP with LDAP support enabled before PHP's LDAP calls will work.

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"
```

More Information

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.

Inhaltsverzeichnis

[ldap_add](#) Add entries to LDAP directory

[ldap_mod_add](#) Add attribute values to current attributes

[ldap_mod_del](#) Delete attribute values from current attributes

[ldap_mod_replace](#) Replace attribute values with new ones

[ldap_bind](#) Bind to LDAP directory

[ldap_close](#) Close link to LDAP server

[ldap_connect](#) Connect to an LDAP server

[ldap_count_entries](#) Count the number of entries in a search

[ldap_delete](#) Delete an entry from a directory

[ldap_dn2ufn](#) Convert DN to User Friendly Naming format

[ldap_explode_dn](#) Splits DN into its component parts

[ldap_first_attribute](#) Return first attribute

[ldap_first_entry](#) Return first result id

[ldap_free_result](#) Free result memory

[ldap_get_attributes](#) Get attributes from a search result entry

[ldap_get_dn](#) Get the DN of a result entry

[ldap_get_entries](#) Get all result entries

[ldap_get_values](#) Get all values from a result entry

[ldap_get_values_len](#) Get all binary values from a result entry

[ldap_list](#) Single-level search

[ldap_modify](#) Modify an LDAP entry

[ldap_next_attribute](#) Get the next attribute in result

[ldap_next_entry](#) Get next result entry

[ldap_read](#) Read an entry

[ldap_search](#) Search LDAP tree

[ldap_unbind](#) Unbind from LDAP directory

[ldap_err2str](#) Convert LDAP error number into string error message

[ldap_errno](#) Return the LDAP error number of the last LDAP command

[ldap_error](#) Return the LDAP error message of the last LDAP command

[Zurück](#)

[Anfang](#)

[Vor](#)

[ibase_num_fields](#)

[Hoch](#)

[ldap_add](#)

ldap_add

(PHP3 , PHP4)

ldap_add -- Add entries to LDAP directory

Description

int ldap_add (int link_identifier, string dn, array entry)

returns true on success and false on error.

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
```

Beispiel 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_mod_add

(PHP3 >= 3.0.8, PHP4)

ldap_mod_add -- Add attribute values to current attributes

Description

```
int ldap_mod_add (int link_identifier, string dn, array entry)
```

returns true on success and false on error.

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

(PHP3 >= 3.0.8, PHP4)

ldap_mod_del -- Delete attribute values from current attributes

Description

int **ldap_mod_del** (int link_identifier, string dn, array entry)

returns true on success and false on error.

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_del()** function.

ldap_mod_replace

(PHP3 >= 3.0.8, PHP4)

ldap_mod_replace -- Replace attribute values with new ones

Description

`int ldap_mod_replace (int link_identifier, string dn, array entry)`

returns true on success and false on error.

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_bind

(PHP3 , PHP4)

ldap_bind -- Bind to LDAP directory

Description

```
int ldap_bind (int link_identifier [, string bind_rdn [, string  
bind_password]])
```

Binds to the LDAP directory with specified RDN and password. Returns true on success and false on error.

ldap_bind() does a bind operation on the directory. bind_rdn and bind_password are optional. If not specified, anonymous bind is attempted.

ldap_close

(PHP3 , PHP4)

ldap_close -- Close link to LDAP server

Description

```
int ldap_close (int link_identifier)
```

Returns true on success, false on error.

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()**.

[Zurück](#)[Anfang](#)[Vor](#)

ldap_bind

[Hoch](#)

ldap_connect

ldap_connect

(PHP3 , PHP4)

ldap_connect -- Connect to an LDAP server

Description

```
int 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.

ldap_count_entries

(PHP3 , PHP4)

ldap_count_entries -- Count the number of entries in a search

Description

```
int ldap_count_entries (int link_identifier, int 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

(PHP3 , PHP4)

ldap_delete -- Delete an entry from a directory

Description

```
int ldap_delete (int link_identifier, string dn)
```

Returns true on success and false on error.

ldap_delete() function delete a particular entry in LDAP directory specified by dn.

ldap_dn2ufn

(PHP3 , PHP4)

ldap_dn2ufn -- Convert DN to User Friendly Naming format

Description

string **ldap_dn2ufn** (string dn)

ldap_dn2ufn() function is used to turn a DN into a more user-friendly form, stripping off type names.

ldap_explode_dn

(PHP3 , PHP4)

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 a 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

(PHP3 , PHP4)

ldap_first_attribute -- Return first attribute

Description

string **ldap_first_attribute** (int link_identifier, int 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 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

(PHP3 , PHP4)

ldap_first_entry -- Return first result id

Description

int ldap_first_entry (int link_identifier, int 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 **lap_next_entry()** routine to get successive entries from the result.

see also [ldap_get_entries\(\)](#).

ldap_free_result

(PHP3 , PHP4)

ldap_free_result -- Free result memory

Description

```
int ldap_free_result (int result_identifier)
```

Returns true on success and false on error.

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

(PHP3 , PHP4)

ldap_get_attributes -- Get attributes from a search result entry

Description

array **ldap_get_attributes** (int link_identifier, int 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] = ith value of the attribute
```

Beispiel 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\(\)](#)

[Zurück](#)[ldap_free_result](#)[Anfang](#)[Hoch](#)[Vor](#)[ldap_get_dn](#)

ldap_get_dn

(PHP3 , PHP4)

ldap_get_dn -- Get the DN of a result entry

Description

string **ldap_get_dn** (int link_identifier, int 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

(PHP3 , PHP4)

ldap_get_entries -- Get all result entries

Description

array **ldap_get_entries** (int link_identifier, int result_identifier)

Returns a complete result information in a multi-dimentional array on success and false on error.

ldap_get_entries() function is used to simplify reading multiple entries from the result 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_values

(PHP3 , PHP4)

ldap_get_values -- Get all values from a result entry

Description

array **ldap_get_values** (int link_identifier, int 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

Beispiel 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>";
```

[Zurück](#)[ldap_get_entries](#)[Anfang](#)[Hoch](#)[Vor](#)[ldap_get_values_len](#)

ldap_get_values_len

(PHP3 >= 3.0.13, PHP4 >= 4.0RC2)

ldap_get_values_len -- Get all binary values from a result entry

Description

array **ldap_get_values_len** (int link_identifier, int 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.

ldap_list

(PHP3 , PHP4)

ldap_list -- Single-level search

Description

```
int ldap_list (int link_identifier, string base_dn, string filter [,  
array attributes])
```

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 an optional fourth parameter which is an array of the attributes required. See [ldap_search\(\)](#) notes.

Beispiel 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] ;
```


ldap_modify

(PHP3 , PHP4)

ldap_modify -- Modify an LDAP entry

Description

```
int ldap_modify (int link_identifier, string dn, array entry)
```

Returns true on success and false on error.

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

(PHP3 , PHP4)

ldap_next_attribute -- Get the next attribute in result

Description

```
string ldap_next_attribute (int link_identifier, int  
result_entry_identifier, int 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

(PHP3 , PHP4)

ldap_next_entry -- Get next result entry

Description

int ldap_next_entry (int link_identifier, int 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_identifier as returned from the [ldap_first_entry\(\)](#).

see also [ldap_get_entries\(\)](#)

ldap_read

(PHP3 , PHP4)

ldap_read -- Read an entry

Description

```
int ldap_read (int link_identifier, string base_dn, string filter [,  
array attributes])
```

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 an optional fourth parameter which is an array of the attributes required. See [ldap_search\(\)](#) notes.

ldap_search

(PHP3 , PHP4)

ldap_search -- Search LDAP tree

Description

```
int ldap_search (int link_identifier, string base_dn, string filter [,  
array attributes])
```

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.

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.

Beispiel 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>";
```

[Zurück](#)[ldap_read](#)[Anfang](#)[Hoch](#)[Vor](#)[ldap_unbind](#)

ldap_unbind

(PHP3 , PHP4)

ldap_unbind -- Unbind from LDAP directory

Description

```
int ldap_unbind (int link_identifier)
```

Returns true on success and false on error.

ldap_unbind() function unbinds from the LDAP directory.

ldap_err2str

(PHP3 >= 3.0.13, PHP4 >= 4.0RC2)

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\(\)](#).

Beispiel 1 Enumerating all LDAP error messages

```
<?php
    for($i=0; $i<100; $i++) {
        printf("Error $i: %s<br>\n", ldap_str2err($i));
    }
?>
```


ldap_errno

(PHP3 >= 3.0.12, PHP4 >= 4.0RC2)

ldap_errno -- Return the LDAP error number of the last LDAP command

Description

```
int ldap_errno (int link_id)
```

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 php3.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.

Beispiel 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

(PHP3 >= 3.0.12, PHP4 >= 4.0RC2)

ldap_error -- Return the LDAP error message of the last LDAP command

Description

string **ldap_error** (int link_id)

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 `php3.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\(\)](#).

XXX. Mail Funktionen

Die **Mail()** Funktionen erlauben das Versenden von Mail.

Inhaltsverzeichnis

[mail](#) Sende Mail

[Zurück](#)

[Anfang](#)

[Vor](#)

ldap_error

[Hoch](#)

mail

mail

(PHP3 , PHP4)

mail -- Sende Mail

Beschreibung

bool **mail** (string *to*, string *subject*, string *message* [, string *additional_headers*])

Mail() verschickt die Nachricht in *message* an den Empfänger in *to*. Mehrere Empfänger können angegeben werden, indem ein Komma zwischen die Adressen in *to* gesetzt wird.

Beispiel 1 Verschicken von Mail.

```
mail("rasmus@lerdorf.on.ca", "My Subject", "Line 1\nLine 2\nLine 3");
```

Wenn ein vierter String-Parameter übergeben wird, dann wird dieser an das Ende des Mail-Headers gehängt. Mehrfache Header werden durch einen Zeilenumbruch getrennt.

Beispiel 2 Verschicke Mail mit zusätzlichen Headers.

```
mail("nobody@aol.com", "the subject", $message,  
    "From: webmaster@$SERVER_NAME\nReply-To: webmaster@$SERVER_NAME\nX-Mailer: PHP/"  
    . phpversion());
```

XXXI. Mathematische Funktionen

Introduction

Die mathematischen Funktionen behandeln nur Werte im Bereich der Datentypen long und double der jeweiligen Architektur. Wenn Sie größere Zahlen oder Werte mit höherer Genauigkeit benötigen, sollten Sie sich die [mathematische Funktionen mit beliebiger Genauigkeit](#) ansehen.

Mathematische Konstanten

Die folgenden Werte werden von diesem Packet definiert:

Tabelle 1 Mathematische Konstanten

Konstante	Wert	Beschreibung
M_PI	3.14159265358979323846	Der Wert π (Pi)
M_E	2.7182818284590452354	e
M_LOG2E	1.4426950408889634074	$\log_2 e$
M_LOG10E	0.43429448190325182765	$\log_{10} e$
M_LN2	0.69314718055994530942	$\log_e 2$
M_LN10	2.30258509299404568402	$\log_e 10$
M_PI_2	1.57079632679489661923	$\pi/2$
M_PI_4	0.78539816339744830962	$\pi/4$
M_1_PI	0.31830988618379067154	$1/\pi$
M_2_PI	0.63661977236758134308	$2/\pi$
M_2_SQRTPI	1.12837916709551257390	$2/\sqrt{\pi}$
M_SQRT2	1.41421356237309504880	$\sqrt{2}$
M_SQRT1_2	0.70710678118654752440	$1/\sqrt{2}$

Bis auf M_PI sind diese Konstanten erst ab PHP4.0 verfügbar.

Inhaltsverzeichnis

[Abs](#) Absolutwert

[Acos](#) Arcuscosinus

[Asin](#) Arcussinus

[Atan](#) Arcustangens

[Atan2](#) Arcustangens aus zwei Werten

[base_convert](#) Konvertiert Werte zwischen Zahlensystemen unterschiedlicher Basis

[BinDec](#) Binär zu dezimal Konvertierung

[Ceil](#) Aufrunden

[Cos](#) Cosinus

[DecBin](#) Dezimal zu binär Konvertierung

[DecHex](#) Dezimal zu hexadezimal Konvertierung

[DecOct](#) Dezimal zu oktal Konvertierung

[deg2rad](#) Grad in Bogenmaß wandeln

[Exp](#) e hoch ...

[Floor](#) Abrunden

[getrandmax](#) Größtmöglicher Zufallswert

[HexDec](#) Hexadezimal zu dezimal Konvertierung

[Log](#) Natürlicher Logarithmus

[Log10](#) Zehner-Logarithmus

[max](#) Maximalwert bestimmen

[min](#) Minimalwert bestimmen

[mt_rand](#) Erzeugt 'bessere' Zufallszahlen

[mt_srand](#) seed the better random number generator

[mt_getrandmax](#) show largest possible random value

[number_format](#) Formatiert eine Zahl mit Tausendergruppen

[OctDec](#) Oktal zu dezimal Konvertierung

[pi](#) Wert von PI

[pow](#) Exponentialfunktion

[rad2deg](#) Grad in Bogenmaß wandeln

[rand](#) (Pseudo-) Zufallszahl generieren

[round](#) Runden eines Wertes

[Sin](#) Sinus

[Sqrt](#) Quadratwurzel

[srand](#) Startwert für den Zufallsgenerator festlegen

[Tan](#) Tangens

mail

[Hoch](#)

Abs

Abs

(unknown)

Abs -- Absolutwert

Beschreibung

mixed **abs** (mixed number)

Liefert den Absolutwert einer Zahl. Ist *number* vom Typ float, so ist auch das Ergebnis vom Typ float, ansonsten ist es vom Typ int.

Acos

(unknown)

Acos -- Arcuscosinus

Beschreibung

float **acos** (float arg)

Liefert den Arcuscosinus zum Wert *arg* im Bogenmaß.

Siehe auch [asin\(\)](#), [atan\(\)](#) und [cos\(\)](#).

Asin

(unknown)

Asin -- Arcussinus

Beschreibung

float **asin** (float arg)

Liefert den Arcussinus zum Wert *arg* im Bogenmaß.

Siehe auch [acos\(\)](#), [atan\(\)](#) und [sin\(\)](#).

Atan

(unknown)

Atan -- Arcustangens

Beschreibung

float **atan** (float arg)

Liefert den Arcustangens zum Wert *arg* im Bogenmaß im Bereich $(-\pi/2 ; \pi/2]$.

See also [asin\(\)](#), [acos\(\)](#), [atan2\(\)](#) und [tan\(\)](#).

Atan2

(unknown)

Atan2 -- Arcustangens aus zwei Werten

Beschreibung

float **atan2** (float *y*, float *x*)

Diese Funktion berechnet den Arcustangens aus den Parametern *x* und *y* im Bereich $(-\pi; \pi]$. Dieser entspricht weitgehend dem Arcustangens aus x / y , allerdings kann hier das Vorzeichen beider Parameter ausgewertet und so der Quadrant des Ergebnisses bestimmt werden.

Siehe auch [atan\(\)](#), [acos\(\)](#), [asin\(\)](#), und [tan\(\)](#).

base_convert

(PHP3 >= 3.0.6, PHP4)

base_convert -- Konvertiert Werte zwischen Zahlensystemen unterschiedlicher Basis

Beschreibung

string **base_convert** (string number, int frombase, int tobase)

Diese Funktion liefert einen String, der *number* zur Basis *tobase* enthält. Der Parameter *frombase* gibt die Basis des Eingabewertes *number* an. *tobase* und *frombase* können Werte von 2 bis einschließlich 36 enthalten. Für die Darstellung von Werten mit einer Basis größer als 10 werden neben den Ziffern '0' bis '9' auch die Buchstaben von 'a' bis 'z' eingesetzt, wobei 'a' einem Wert von 10 entspricht, 'b' steht dementsprechend für den Wert 11, bis hin zum 'z', das den Wert 36 darstellt.

Beispiel 1 base_convert()

```
$binary = base_convert($hexadecimal, 16, 2);
```

BinDec

(unknown)

BinDec -- Binär zu dezimal Konvertierung

Beschreibung

`int bindec (string binary_string)`

Liefert die dezimale Entsprechung der in *binary_string* übergebenen binären Zahl. Der größte konvertierbare Wert ist ein String aus 31 Einsen, das entspricht 2147483647 dezimal (2 hoch 32 minus 1).

Siehe auch [decbin\(\)](#), [octdec\(\)](#), und [hexdec\(\)](#).

Ceil

(unknown)

Ceil -- Aufrunden

Beschreibung

int **ceil** (float number)

Diese Funktion liefert diejenige ganze Zahl, die größer oder gleich dem Parameter *number* ist.

Siehe auch [floor\(\)](#) und [round\(\)](#).

Cos

(unknown)

Cos -- Cosinus

Beschreibung

float **cos** (float arg)

Liefert den Cosinus des Winkels *arg* (*arg* in Bogenmaß).

Siehe auch [sin\(\)](#), [tan\(\)](#) und [acos\(\)](#).

DecBin

(unknown)

DecBin -- Dezimal zu binär Konvertierung

Beschreibung

string **decbin** (int number)

Diese Funktion liefert eine binäre Representation von *number* als String zurück. Der größte konvertierbare Wert für *number* ist dezimal 2147483647, dies entspricht (2 hoch 32)-1 und liefert einen String aus 31 Einsen.

Siehe auch [bindec\(\)](#) sowie [dechex\(\)](#) und [decoct\(\)](#).

DecHex

(unknown)

DecHex -- Dezimal zu hexadezimal Konvertierung

Beschreibung

string **dechex** (int number)

Diese Funktion liefert eine hexadezimale Representation von *number* als String zurück. Der größte konvertierbare Wert für *number* ist dezimal 2147483647, dies entspricht (2 hoch 32)-1 und liefert als Ergebnis den String "7fffffff" zurück.

Siehe auch [hexdec\(\)](#) sowie [decbin\(\)](#) und [decoct\(\)](#).

DecOct

(unknown)

DecOct -- Dezimal zu oktal Konvertierung

Beschreibung

string **decoct** (int number)

Diese Funktion liefert eine oktale (zur Basis 8) Representation von *number* als String zurück. Der größte konvertierbare Wert für *number* ist dezimal 2147483647, dies entspricht $(2^{\text{hoch } 32})-1$ und liefert als Ergebnis den String "17777777777" zurück.

Siehe auch [octdec\(\)](#) sowie [decbin\(\)](#) und [dechex\(\)](#).

deg2rad

(PHP3 >= 3.0.4, PHP4)

deg2rad -- Grad in Bogenmaß wandeln

Beschreibung

double **deg2rad** (double number)

Diese Funktion wandelt den übergebenen Winkel von Grad in Bogenmaß.

Siehe auch [rad2deg\(\)](#).

Exp

(unknown)

Exp -- e hoch ...

Beschreibung

float **exp** (float arg)

Liefert e potenziert mit *arg*.

Siehe auch [pow\(\)](#).

Floor

(unknown)

Floor -- Abrunden

Beschreibung

int **floor** (float number)

Diese Funktion liefert die ganze Zahl, die kleiner oder gleich dem Parameter *number* ist.

Siehe auch [ceil\(\)](#) und [round\(\)](#).

getrandmax

(PHP3 , PHP4)

getrandmax -- Größtmöglicher Zufallswert

Beschreibung

int **getrandmax** (void)

Diese Funktion liefert den höchsten Wert, den ein Aufruf der Funktion [rand\(\)](#) auf dieser Plattform zurückgeben kann.

Siehe auch [rand\(\)](#), [srand\(\)](#) [mt_rand\(\)](#), [mt_srand\(\)](#) und [mt_getrandmax\(\)](#).

HexDec

(unknown)

HexDec -- Hexadezimal zu dezimal Konvertierung

Beschreibung

`int hexdec (string hex_string)`

Liefert die dezimale Entsprechung der in *hex_string* übergebenen hexadezimalen Zahl. Der größte konvertierbare Wert ist "7fffffff", das entspricht 2147483647 dezimal (2 hoch 32 minus 1).

Siehe auch [dechex\(\)](#) sowie [bindec\(\)](#) und [octdec\(\)](#).

Log

(unknown)

Log -- Natürlicher Logarithmus

Beschreibung

float **log** (float arg)

Diese Funktion liefert den natürlichen Logarithmus (zur Basis e) des Parameters *arg*.

Siehe auch [exp\(\)](#) und [log10\(\)](#).

Log10

(unknown)

Log10 -- Zehner-Logarithmus

Beschreibung

float **log10** (float arg)

Diese Funktion liefert den Logarithmus des Parameters *arg* zur Basis zehn.

Siehe auch [log\(\)](#).

max

(PHP3 , PHP4)

max -- Maximalwert bestimmen

Beschreibung

mixed **max** (mixed arg1, mixed arg2, mixed argn)

max() liefert den numerisch größten Eingabewert zurück.

Ist der erste Parameter ein Array, so wird der größte Wert innerhalb dieses Array zurückgegeben. Anderenfalls müssen mindestens zwei Parameter übergeben werden, und **max()** liefert den Wert des größten Parameters zurück. Die Anzahl der Eingabe-Paramter ist nur durch den verfügbaren Hauptspeicher begrenzt.

Ist mindestens einer der Eigabewerte vom Typ double, so werden alle Werte als double behandelt und der Rückgabewert ist ebenfalls vom Typ double. Anderenfalls werden alle Eingabewerte als int behandelt und auch ein int zurückgegeben.

Siehe auch [min\(\)](#).

min

(PHP3 , PHP4)

min -- Minimalwert bestimmen

Beschreibung

mixed **min** (mixed arg1, mixed arg2, mixed argn)

min() liefert den numerisch kleinsten Eingabewert zurück.

Ist der erste Parameter ein Array, so wird der kleinste Wert innerhalb dieses Array zurückgegeben. Anderenfalls müssen mindestens zwei Parameter übergeben werden, und **min()** liefert den Wert des kleinsten Parameters zurück. Die Anzahl der Eingabe-Paramter ist nur durch den verfügbaren Hauptspeicher begrenzt.

Ist mindestens einer der Eigabewerte vom Typ double, so werden alle Werte als double behandelt und der Rückgabewert ist ebenfalls vom Typ double. Anderenfalls werden alle Eingabewerte als int behandelt und auch ein int zurückgegeben.

Siehe auch [max\(\)](#).

mt_rand

(PHP3 >= 3.0.6, PHP4)

mt_rand -- Erzeugt 'bessere' Zufallszahlen

Beschreibung

```
int mt_rand ([int min [, int max]])
```

Die PHP [rand\(\)](#) Funktion benutzt den in der jeweiligen libc enthaltenen Zufallsgenerator. Gerade in älteren libc-Versionen ist dieser aber oft langsam und die Qualität der erzeugten Pseudozufalls-Sequenzen ist unbefriedigend bis unbekannt.

Die Funktion **mt_rand()** bietet einen Drop-In Ersatz für die von der libc bereitgestellten Funktionen. Sie benutzt einen Zufallsgenerator mit bekannter Charakteristik, den sogenannten "Mersenne Twister", der Pseudozufalls-Sequenzen generiert, die auch kryptografischen Ansprüchen genügen sollten und viermal so schnell arbeitet wie der typische libc-Algorithmus.

Weitere Informationen zum Mersenne Twister und zur optimierten Implementation finden sich unter <http://www.math.keio.ac.jp/~matumoto/emt.html> und <http://www.scp.syr.edu/~marc/hawk/twister.html>.

Ohne die optionalen Parameter *min* und *max* liefert **mt_rand()** eine Pseudo-Zufallszahl zwischen 0 und einem plattformabhängigen Maximalwert, der mit [mt_getrandmax\(\)](#) ermittelt werden kann.

Benutzen Sie unbedingt [mt_srand\(\)](#), um vor dem ersten Aufruf von **mt_rand()** einen neuen Startwert festzulegen, ansonsten erhalten Sie bei jedem Programmlauf dieselbe Sequenz von Werten.

Anmerkung: In PHP-Versionen vor 3.0.7 hatte der zweite Parameter *max* nicht den Maximalwert sondern die Größe des Wertintervalls an, d.h. in diesen älteren Version muß statt `mt_rand(min, max)` `mt_rand(min, max-min)` benutzt werden, um dasselbe Ergebnis zu erhalten.

Siehe auch [mt_srand\(\)](#), [mt_getrandmax\(\)](#), [srand\(\)](#), [rand\(\)](#) und [getrandmax\(\)](#).

mt_srand

(PHP3 >= 3.0.6, PHP4)

mt_srand -- seed the better random number generator

Beschreibung

`void mt_srand (int seed)`

Legt einen internen Startwert für den Zufallsgenerator in [mt_rand\(\)](#) fest. Gleiche Startwerte führen auch jeweils zu derselben Pseudo-Zufallssequenzen daher ist es üblich, diese über die aktuelle Uhrzeit festzulegen:

```
// seed with microseconds since last "whole" second
mt_srand((double)microtime()*1000000);
$randval = mt_rand();
```

Siehe auch [mt_rand\(\)](#), [mt_getrandmax\(\)](#), [srand\(\)](#), [rand\(\)](#) und [getrandmax\(\)](#).

mt_getrandmax

(PHP3 >= 3.0.6, PHP4)

mt_getrandmax -- show largest possible random value

Beschreibung

`int mt_getrandmax (void)`

Diese Funktion liefert den höchsten Wert, den ein Aufruf der Funktion [mt_rand\(\)](#) auf dieser Plattform zurückgeben kann.

Siehe auch [mt_rand\(\)](#), [mt_srand\(\)](#), [rand\(\)](#), [srand\(\)](#) and [getrandmax\(\)](#).

number_format

(PHP3 , PHP4)

number_format -- Formatiert eine Zahl mit Tausendergruppen

Beschreibung

string **number_format** (float number [, int decimals [, string dec_point [, string thousands_sep]])

number_format() erzeugt aus *number* einen formatierten String. Die Funktion akzeptiert ein, zwei oder vier Parameter (nicht drei).

Wird nur *number* übergeben, so wird diese Zahl ohne Nachkommastellen, aber mit durch Komma getrennten Tausendergruppen ausgegeben. (1234567 -> 1,234,567)

Der optionale Parameter *decimals* gibt die Anzahl der auszugebenden Nachkommastellen an, diese werden durch einen Punkt getrennt ausgegeben.

Mit Hilfe der weiteren Parameter *dec_point* und *thousands_sep* können andere Trennzeichen für Tausendergruppen und Nachkommastellen festgelegt werden. Diese beiden Parameter können nur zusammen angegeben werden.

Im deutschen Sprachraum werden die Trennzeichen i.A. genau umgekehrt benutzt, d.h. Nachkommastellen werden durch ein Komma und Tausender durch einen Punkt getrennt.

OctDec

(unknown)

OctDec -- Oktal zu dezimal Konvertierung

Beschreibung

`int octdec (string octal_string)`

Liefert die dezimale Entsprechung der in *octal_string* übergebenen oktalen Zahl. Der größte konvertierbare Wert ist "17777777777", das entspricht 2147483647 dezimal (2 hoch 32 minus 1).

Siehe auch [decoct\(\)](#) sowie [bindec\(\)](#) und [hexdec\(\)](#).

pi

(PHP3 , PHP4)

pi -- Wert von PI

Beschreibung

double **pi** (void)

Diese Funktionen liefert (im Rahmen der möglichen Genauigkeit) den Wert der Kreiszahl Pi. Die Konstante M_PI erfüllt denselben Zweck allerdings wesentlich effektiver.

pow

(PHP3 , PHP4)

pow -- Exponentialfunktion

Beschreibung

float **pow** (float base, float exp)

Diese Funktion liefert *base* potenziert mit *exp*, also *base* hoch *exp*.

Siehe auch [exp\(\)](#).

rad2deg

(PHP3 >= 3.0.4, PHP4)

rad2deg -- Grad in Bogenmaß wandeln

Beschreibung

double **rad2deg** (double number)

Diese Funktion wandelt den übergebenen Winkel von Bogenmaß in Grad.

Siehe auch [deg2rad\(\)](#).

rand

(PHP3 , PHP4)

rand -- (Pseudo-) Zufallszahl generieren

Beschreibung

```
int rand ([int min [, int max]])
```

Diese Funktion liefert eine Pseudo-Zufallszahl zwischen *min* und *max* (inclusive). Ohne die optionalen Parameter *min* und *max* liefert **rand()** eine Pseudo-Zufallszahl zwischen 0 und einem plattformabhängigen Maximalwert, der mit [getrandmax\(\)](#) ermittelt werden kann.

Benutzen Sie unbedingt [srand\(\)](#), um vor dem ersten Aufruf von **rand()** einen neuen Startwert festzulegen, ansonsten erhalten Sie bei jedem Programmlauf dieselbe Sequenz von Werten.

Anmerkung: In PHP-Versionen vor 3.0.7 hatte der zweite Parameter *max* nicht den Maximalwert sondern die Größe des Wertebereichs an, d.h. in diesen älteren Version muß statt `rand (min, max)` `rand (min, max-min)` benutzt werden, um dasselbe Ergebnis zu erhalten.

Siehe auch [srand\(\)](#), [getrandmax\(\)](#), [mt_rand\(\)](#), [mt_srand\(\)](#) und [mt_getrandmax\(\)](#).

round

(PHP3 , PHP4)

round -- Runden eines Wertes

Beschreibung

double **round** (double *val* [, int *precision*])

Diese Funktion rundet den Eingabewert *val* auf *precision* Stellen nach dem Komma auf oder ab.

```
$foo = round (3.4);    // $foo == 3.0
$foo = round (3.5);    // $foo == 4.0
$foo = round (3.6);    // $foo == 4.0

$foo = round (1.95583,2);    // $foo == 1.96
```

Siehe auch [ceil\(\)](#) und [floor\(\)](#).

Sin

(unknown)

Sin -- Sinus

Beschreibung

float **sin** (float arg)

Liefert den Sinus des Eingabewertes *arg*. *arg* wird im Bogenmaß angegeben.

Siehe auch [cos\(\)](#), [tan\(\)](#) und [asin\(\)](#).

Sqrt

(unknown)

Sqrt -- Quadratwurzel

Beschreibung

float **sqrt** (float arg)

Diese Funktion liefert die Quadratwurzel des Wertes *arg*.

[Zurück](#)

Sin

[Anfang](#)[Hoch](#)[Vor](#)

strand

srand

(PHP3 , PHP4)

srand -- Startwert für den Zufallsgenerator festlegen

Beschreibung

void **srand** (int seed)

Legt einen internen Startwert für den Zufallsgenerator in [rand\(\)](#) fest. Gleiche Startwerte führen auch jeweils zu derselben Pseudo-Zufallssequenzen daher ist es üblich, diese über die aktuelle Uhrzeit festzulegen:

```
// seed with microseconds since last "whole" second
srand ((double)microtime()*1000000);
$randval = rand();
```

Siehe auch [rand\(\)](#), [getrandmax\(\)](#), [mt_rand\(\)](#), [mt_srand\(\)](#) und [mt_getrandmax\(\)](#).

Tan

(unknown)

Tan -- Tangens

Beschreibung

float **tan** (float arg)

Liefert den Tangens des Wertes *arg*. *arg* wird dabei im Bogenmaß angegeben.

Siehe auch [sin\(\)](#), [cos\(\)](#), [atan\(\)](#) und [atan2\(\)](#).

XXXII. MCAL functions

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 reoccurring events.

With libmcal, central calendar servers can be accessed and used, removing the need for any specific database or local file programming.

To get these functions to work, you have to compile PHP with `--with-mcal`. That requires the mcal library to be installed. Grab the latest version from <http://mcal.chek.com/> and compile and install it.

The following constants are defined when using the MCAL module: MCAL_SUNDAY, MCAL_MONDAY, MCAL_TUESDAY, MCAL_WEDNESDAY, MCAL_THURSDAY, MCAL_FRIDAY, MCAL_SATURDAY, MCAL_RECUR_NONE, MCAL_RECUR_DAILY, MCAL_RECUR_WEEKLY, MCAL_RECUR_MONTHLY_MDAY, MCAL_RECUR_MONTHLY_WDAY, MCAL_RECUR_YEARLY, MCAL_JANUARY, MCAL_FEBRUARY, MCAL_MARCH, MCAL_APRIL, MCAL_MAY, MCAL_JUNE, MCAL_JULY, MCAL_AUGUST, MCAL_SEPTEMBER, MCAL_OCTOBER, MCAL_NOVEMBER, and MCAL_DECEMBER. 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.

Inhaltsverzeichnis

[mcal_open](#) Opens up an MCAL connection

[mcal_close](#) Close an MCAL stream

[mcal_fetch_event](#) Fetches an event from the calendar stream

[mcal_list_events](#) Return a list of events between two given datetimes

[mcal_append_event](#) Store a new event into an MCAL calendar

[mcal_store_event](#) Modify an existing event in an MCAL calendar

[mcal_delete_event](#) Delete an event from an MCAL calendar

[mcal_snooze](#) Turn off an alarm for an event

[mcal_list_alarms](#) Return a list of events that has an alarm triggered at the given datetime

[mcal_event_init](#) Initializes a streams global event structure

[mcal_event_set_category](#) Sets the category of the streams global event structure

[mcal_event_set_title](#) Sets the title of the streams global event structure

[mcal_event_set_description](#) Sets the description of the streams global event structure

[mcal_event_set_start](#) Sets the start date and time of the streams global event structure

[mcal_event_set_end](#) Sets the end date and time of the streams global event structure

[mcal_event_set_alarm](#) Sets the alarm of the streams global event structure

[mcal_event_set_class](#) Sets the class of the streams global event structure

[mcal_is_leap_year](#) Returns if the given year is a leap year or not

[mcal_days_in_month](#) Returns the number of days in the given month

[mcal_date_valid](#) Returns true if the given year, month, day is a valid date

[mcal_time_valid](#) Returns true if the given year, month, day is a valid time

[mcal_day_of_week](#) Returns the day of the week of the given date

[mcal_day_of_year](#) Returns the day of the year of the given date

[mcal_date_compare](#) Compares two dates

[mcal_next_recurrence](#) Returns the next recurrence of the event

[mcal_event_set_recur_none](#) Sets the recurrence of the streams global event structure

[mcal_event_set_recur_daily](#) Sets the recurrence of the streams global event structure

[mcal_event_set_recur_weekly](#) Sets the recurrence of the streams global event structure

[mcal_event_set_recur_monthly_mday](#) Sets the recurrence of the streams global event structure

[mcal_event_set_recur_monthly_wday](#) Sets the recurrence of the streams global event structure

[mcal_event_set_recur_yearly](#) Sets the recurrence of the streams global event structure

[mcal_fetch_current_stream_event](#) Returns an object containing the current streams event structure

[Zurück](#)[Anfang](#)[Vor](#)

Tan

[Hoch](#)[mcal_open](#)

mcal_open

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

mcal_open -- Opens up an MCAL connection

Description

`int mcal_open (string calendar, string username, string password, string 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_close

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

mcal_close -- Close an MCAL stream

Description

```
int mcal_close (int mcal_stream, int flags)
```

Closes the given mcal stream.

[Zurück](#)[Anfang](#)[Vor](#)

mcal_open

[Hoch](#)

mcal_fetch_event

mcal_fetch_event

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

`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 is 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

[Zurück](#)

mcal_close

[Anfang](#)

[Hoch](#)

[Vor](#)

mcal_list_events

mcal_list_events

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

mcal_list_events -- Return a list of events between two given datetimes

Description

```
array mcal_list_events (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 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 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_append_event

(PHP4 >= 4.0RC1)

mcal_append_event -- Store a new event into an MCAL calendar

Description

`int mcalf_append_event (int mcalf_stream)`

mcal_append_event() Stores the global event into an MCAL calendar for the given stream.

Returns the uid of the newly inserted event.

mcal_store_event

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

mcal_store_event -- Modify an existing event in an MCAL calendar

Description

`int mcalf_store_event (int mcalf_stream)`

mcalf_store_event() Stores the modifications to the current global event for the given stream.

Returns true on success and false on error.

mcal_delete_event

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

mcal_delete_event -- Delete an event from an MCAL calendar

Description

`int mcald_delete_event (int uid)`

mcald_delete_event() deletes the calendar event specified by the uid.

Returns true.

mcal_snooze

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

mcal_snooze -- Turn off an alarm for an event

Description

`int mcalsnooze (int uid)`

mcalsnooze() turns off an alarm for a calendar event specified by the uid.

Returns true.

mcal_list_alarms

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

mcal_list_alarms -- Return a list of events that has an alarm triggered at the given datetime

Description

```
array mcal_list_events (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_event_init

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

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 0, or the default settings.

Returns true.

mcal_event_set_category

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

mcal_event_set_category -- Sets the category of the streams global event structure

Description

```
int mcalf_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_title

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

`mcal_event_set_title` -- Sets the title of the streams global event structure

Description

`int mcald_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_event_set_description

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

mcal_event_set_description -- Sets the description of the streams global event structure

Description

int mcald_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_start

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

`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_end

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

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.

[Zurück](#)[Anfang](#)[Vor](#)[mcal_event_set_start](#)[Hoch](#)[mcal_event_set_alarm](#)

mcal_event_set_alarm

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

mcal_event_set_alarm -- Sets the alarm of the streams global event structure

Description

`int mcalf_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_class

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

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_is_leap_year

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

mcal_is_leap_year -- Returns if the given year is a leap year or not

Description

int mc**al_is_leap_year** (int year)

mcal_is_leap_year() returns 1 if the given year is a leap year, 1 if not.

mcal_days_in_month

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

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_date_valid

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

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_time_valid

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

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)`

mc~~al~~_time_valid() Returns true if the given hour, minutes and seconds is a valid time, false if not.

[Zurück](#)[Anfang](#)[Vor](#)

mcal_date_valid

[Hoch](#)

mcal_day_of_week

mcal_day_of_week

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

mcal_day_of_week -- Returns the day of the week of the given date

Description

int **mcal_** (int year, int month, int day)

mcal_day_of_week() returns the day of the week of the given date

[Zurück](#)[Anfang](#)[Vor](#)

mcal_time_valid

[Hoch](#)

mcal_day_of_year

mcal_day_of_year

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

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_date_compare

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

mcal_date_compare -- Compares two dates

Description

```
int mcald_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 <0, 0, >0 if a<b, a==b, a>b respectively

[Zurück](#)[Anfang](#)[Vor](#)[mcal_day_of_year](#)[Hoch](#)[mcal_next_recurrence](#)

mcal_next_recurrence

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

mcal_next_recurrence -- Returns the next recurrence of the event

Description

int mcald_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.

[Zurück](#)[Anfang](#)[Vor](#)[mcal_date_compare](#)[Hoch](#)[mcal_event_set_recur_none](#)

mcal_event_set_recur_none

(PHP3 >= 3.0.15, PHP4 >= 4.0RC1)

mcal_event_set_recur_none -- Sets the recurrence of the streams global event structure

Description

int mcald_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).

[Zurück](#)[Anfang](#)[Vor](#)[mcal_next_recurrence](#)[Hoch](#)[mcal_event_set_recur_daily](#)

mcal_event_set_recur_daily

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

mcal_event_set_recur_daily -- Sets the recurrence of the streams global event structure

Description

`int mcalf_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 reoccurring on a daily basis, ending at the given date.

mcal_event_set_recur_weekly

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

mcal_event_set_recur_weekly -- Sets the recurrence of the streams global event structure

Description

int mcald_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 reoccurring on a weekly basis, ending at the given date.

mcal_event_set_recur_monthly_mday

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

mcal_event_set_recur_monthly_mday -- Sets the recurrence of the streams global event structure

Description

int mcald_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 reoccurring on a monthly by month day basis, ending at the given date.

mcal_event_set_recur_monthly_wday

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

mcal_event_set_recur_monthly_wday -- Sets the recurrence of the streams global event structure

Description

`int mcald_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 reoccurring on a monthly by week basis, ending at the given date.

[Zurück](#)[Anfang](#)[Vor](#)[mcal_event_set_recur_monthly_mday](#)[Hoch](#)[mcal_event_set_recur_yearly](#)

mcal_event_set_recur_yearly

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

mcal_event_set_recur_yearly -- Sets the recurrence of the streams global event structure

Description

`int mcalf_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 reoccurring on a yearly basis, ending at the given date .

mcal_fetch_current_stream_event

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

`mcal_fetch_current_stream_event` -- Returns an object containing the current streams event structure

Description

`int mcalfetchcurrentstreamevent (int stream)`

`mcalfetchcurrentstreamevent()` returns the current stream's event structure as an object containing:

- `int id` - ID of that event.
- `int public` - TRUE if the event is 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

[Zurück](#)

[Anfang](#)

[Vor](#)

[mcal_event_set_recur_yearly](#)

[Hoch](#)

[Encryption functions](#)

XXXIII. Encryption functions

These functions work using [mcrypt](#).

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".

To use it, download libmcrypt-x.x.tar.gz from [here](#) and follow the included installation instructions. You need to compile PHP with the `--with-mcrypt` parameter to enable this extension.

mcrypt can be used to encrypt and decrypt using the above mentioned ciphers. 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.

Beispiel 1 Encrypt an input value with TripleDES in ECB mode

```
<?php
$key = "this is a very secret key";
$input = "Let us meet at 9 o'clock at the secret place.";

$encrypted_data = mcrypt_ecb(MCRYPT_TripleDES, $key, $input, MCRYPT_ENCRYPT);
?>
```

This example will give you the encrypted data as a string in `$encrypted_data`.

Mcrypt can operate in four cipher modes (CBC, OFB, CFB, and ECB). We will outline the normal use for each of these modes. For a more complete reference and discussion see Applied Cryptography by Schneier (ISBN 0-471-11709-9).

- 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.
- CBC (cipher block chaining) is especially suitable for encrypting files where the security is increased over ECB significantly.
- CFB (cipher feedback) is the best mode for encrypting byte streams where single bytes must be encrypted.
- OFB (output feedback) is comparable to CFB, but can be used in applications where error propagation cannot be tolerated.

PHP does not support encrypting/decrypting bit streams currently. As of now, PHP only supports handling of strings.

For a complete list of supported ciphers, see the defines at the end of `mcrypt.h`. The general rule is that you can access the cipher from PHP with `MCRYPT_ciphername`.

Here is a short list of ciphers which are currently supported by the mcrypt extension. If a cipher is not listed here,

but is listed by mcrypt as supported, you can safely assume that this documentation is outdated.

- MCRYPT_BLOWFISH
- MCRYPT_DES
- MCRYPT_TripleDES
- MCRYPT_ThreeWAY
- MCRYPT_GOST
- MCRYPT_CRYPT
- MCRYPT_DES_COMPAT
- MCRYPT_SAFER64
- MCRYPT_SAFER128
- MCRYPT_CAST128
- MCRYPT_TEAN
- MCRYPT_RC2
- MCRYPT_TWOFISH (for older mcrypt 2.x versions)
- MCRYPT_TWOFISH128 (TWOFISHxxx are available in newer 2.x versions)
- MCRYPT_TWOFISH192
- MCRYPT_TWOFISH256
- MCRYPT_RC6
- MCRYPT_IDEA

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).

Inhaltsverzeichnis

[mcrypt_get_cipher_name](#) Get the name of the specified cipher

[mcrypt_get_block_size](#) Get the block size of the specified cipher

[mcrypt_get_key_size](#) Get the key size of the specified cipher

[mcrypt_create_iv](#) Create an initialization vector (IV) from a random source

[mcrypt_cbc](#) Encrypt/decrypt data in CBC mode

[mcrypt_cfb](#) Encrypt/decrypt data in CFB mode

[mcrypt_ecb](#) Encrypt/decrypt data in ECB mode

[mcrypt_ofb](#) Encrypt/decrypt data in OFB mode

mcalfetch_current_stream_event

[Hoch](#)

mcrypt_get_cipher_name

mcrypt_get_cipher_name

(PHP3 >= 3.0.8, PHP4)

mcrypt_get_cipher_name -- Get the name of the specified cipher

Description

string **mcrypt_get_cipher_name** (int 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 and returns the name of the cipher or false, if the cipher does not exist.

Beispiel 1 Mcrypt_get_cipher_name() example

```
<?php
$cipher = MCRYPT_TripleDES;

print mcrypt_get_cipher_name ($cipher);
?>
```

The above example will produce:

```
TripleDES
```

mcrypt_get_block_size

(PHP3 >= 3.0.8, PHP4)

mcrypt_get_block_size -- Get the block size of the specified cipher

Description

`int mcrypt_get_block_size (int cipher)`

Mcrypt_get_block_size() is used to get the size of a block of the specified *cipher*.

Mcrypt_get_block_size() takes one argument, the *cipher* and returns the size in bytes.

See also: [mcrypt_get_key_size\(\)](#).

mcrypt_get_key_size

(PHP3 >= 3.0.8, PHP4)

mcrypt_get_key_size -- Get the key size of the specified cipher

Description

```
int mcrypt_get_key_size (int cipher)
```

Mcrypt_get_key_size() is used to get the size of a key of the specified *cipher*.

mcrypt_get_key_size() takes one argument, the *cipher* and returns the size in bytes.

See also: [mcrypt_get_block_size\(\)](#).

mdecrypt_create_iv

(PHP3 >= 3.0.8, PHP4)

mdecrypt_create_iv -- Create an initialization vector (IV) from a random source

Description

string **mdecrypt_create_iv** (int size, int source)

Mdecrypt_create_iv() is used to create an IV.

mdecrypt_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.

Beispiel 1 Mdecrypt_create_iv() example

```
<?php
$cipher = MCRYPT_TripleDES;
$block_size = mdecrypt_get_block_size ($cipher);
$iv = mdecrypt_create_iv ($block_size, MCRYPT_DEV_RANDOM);
?>
```

mdecrypt_cbc

(PHP3 >= 3.0.8, PHP4)

mdecrypt_cbc -- Encrypt/decrypt data in CBC mode

Description

string **mdecrypt_cbc** (int cipher, string key, string data, int mode [, string iv])

Mdecrypt_cbc() encrypts or decrypts (depending on *mode*) the *data* with *cipher* and *key* in CBC cipher mode and returns the resulting string.

Cipher is one of the MCRYPT_ciphrname constants.

Key is the key supplied to the algorithm. It must be kept secret.

Data is the data which shall be encrypted/decrypted.

Mode is MCRYPT_ENCRYPT or MCRYPT_DECRYPT.

IV is the optional initialization vector.

See also: [mdecrypt_cfb\(\)](#), [mdecrypt_ecb\(\)](#), and [mdecrypt_ofb\(\)](#).

mdecrypt_cfb

(PHP3 >= 3.0.8, PHP4)

mdecrypt_cfb -- Encrypt/decrypt data in CFB mode

Description

string **mdecrypt_cfb** (int cipher, string key, string data, int mode, string iv)

Mdecrypt_cfb() encrypts or decrypts (depending on *mode*) the *data* with *cipher* and *key* in CFB cipher mode and returns the resulting string.

Cipher is one of the MCRYPT_ciphrname constants.

Key is the key supplied to the algorithm. It must be kept secret.

Data is the data which shall be encrypted/decrypted.

Mode is MCRYPT_ENCRYPT or MCRYPT_DECRYPT.

IV is the initialization vector.

See also: [mdecrypt_cbc\(\)](#), [mdecrypt_ecb\(\)](#), and [mdecrypt_ofb\(\)](#).

mcrypt_ecb

(PHP3 >= 3.0.8, PHP4)

mcrypt_ecb -- Encrypt/decrypt data in ECB mode

Description

string **mcrypt_ecb** (int cipher, string key, string data, int mode)

Mcrypt_ecb() encrypts or decrypts (depending on *mode*) the *data* with *cipher* and *key* in ECB cipher mode and returns the resulting string.

Cipher is one of the MCRYPT_ciphername constants.

Key is the key supplied to the algorithm. It must be kept secret.

Data is the data which shall be encrypted/decrypted.

Mode is MCRYPT_ENCRYPT or MCRYPT_DECRYPT.

See also: [mcrypt_cbc\(\)](#), [mcrypt_cfb\(\)](#), and [mcrypt_ofb\(\)](#).

mdecrypt_ofb

(PHP3 >= 3.0.8, PHP4)

mdecrypt_ofb -- Encrypt/decrypt data in OFB mode

Description

string **mdecrypt_ofb** (int cipher, string key, string data, int mode, string iv)

Mdecrypt_ofb() encrypts or decrypts (depending on *mode*) the *data* with *cipher* and *key* in OFB cipher mode and returns the resulting string.

Cipher is one of the MCRYPT_ciphrname constants.

Key is the key supplied to the algorithm. It must be kept secret.

Data is the data which shall be encrypted/decrypted.

Mode is MCRYPT_ENCRYPT or MCRYPT_DECRYPT.

IV is the initialization vector.

See also: [mdecrypt_cbc\(\)](#), [mdecrypt_cfb\(\)](#), and [mdecrypt_ecb\(\)](#).

XXXIV. Hash functions

These functions are intended to work with [mhash](#).

This is an interface to the mhash library. mhash supports a wide variety of hash algorithms such as MD5, SHA1, GOST, and many others.

To use it, download the mhash distribution from [its web site](#) and follow the included installation instructions. You need to compile PHP with the `--with-mhash` parameter to enable this extension.

mhash can be used to create checksums, message digests, and more.

Beispiel 1 Compute the SHA1 key and print it out as hex

```
<?php
$input = "Let us meet at 9 o' clock at the secret place.";
$hash = mhash(MHASH_SHA1, $input);

print "The hash is ".bin2hex($hash)."\n";

?>
```

This will produce:

```
The hash is d3b85d710d8f6e4e5efd4d5e67d041f9cecedafe
```

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 HAVAL you use the PHP constant MHASH_HAVAL.

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_HAVAL
- MHASH_RIPEMD160
- MHASH_RIPEMD128

- MHASH_SNEFRU
- MHASH_TIGER
- MHASH_GOST
- MHASH_CRC32
- MHASH_CRC32B

Inhaltsverzeichnis

[mhash_get_hash_name](#) Get the name of the specified hash

[mhash_get_block_size](#) Get the block size of the specified hash

[mhash_count](#) Get the highest available hash id

[mhash](#) Compute hash

[Zurück](#)

mcrypt_ofb

[Anfang](#)

[Hoch](#)

[Vor](#)

mhash_get_hash_name

mhash_get_hash_name

(PHP3 >= 3.0.9, PHP4)

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.

Beispiel 1 mhash_get_hash_name example

```
<?php
$hash = MHASH_MD5;

print mhash_get_hash_name($hash);
?>
```

The above example will print out:

```
MD5
```

mhash_get_block_size

(PHP3 >= 3.0.9, PHP4)

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_count

(PHP3 >= 3.0.9, PHP4)

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 0 to this hash id.

Beispiel 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

(PHP3 >= 3.0.9, PHP4)

mhash -- Compute hash

Description

string **mhash** (int hash, string data)

mhash() applies a hash function specified by *hash* to the *data* and returns the resulting hash (also called digest).

XXXV. Microsoft SQL Server functions

Inhaltsverzeichnis

- [mssql_close](#) Close MS SQL Server connection
- [mssql_connect](#) Open MS SQL server connection
- [mssql_data_seek](#) Move internal row pointer
- [mssql_fetch_array](#) Fetch row as array
- [mssql_fetch_field](#) Get field information
- [mssql_fetch_object](#) Fetch row as object
- [mssql_fetch_row](#) Get row as enumerated array
- [mssql_field_length](#) Get the length of a field
- [mssql_field_name](#) Get the name of a field
- [mssql_field_seek](#) Set field offset
- [mssql_field_type](#) Get the type of a field
- [mssql_free_result](#) Free result memory
- [mssql_get_last_message](#) Returns the last message from server (over min_message_severity?)
- [mssql_min_error_severity](#) Sets the lower error severity
- [mssql_min_message_severity](#) Sets the lower message severity
- [mssql_num_fields](#) Get number of fields in result
- [mssql_num_rows](#) Get number of rows in result
- [mssql_pconnect](#) Open persistent MS SQL connection
- [mssql_query](#) Send MS SQL query
- [mssql_result](#) Get result data
- [mssql_select_db](#) Select MS SQL database

mssql_close

(PHP3 , PHP4)

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

(PHP3 , PHP4)

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

(PHP3 , PHP4)

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 pointer to the specified row number. The next call to [mssql_fetch_row\(\)](#) would return that row.

See also: **mssql_data_seek()**.

mssql_fetch_array

(PHP3 , PHP4)

mssql_fetch_array -- Fetch row as array

Description

int 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_field

(PHP3 , PHP4)

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

(PHP3 , PHP4)

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

(PHP3 , PHP4)

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

(PHP3 >= 3.0.3, PHP4 >= 4.0b4)

mssql_field_length -- Get the length of a field

Description

```
int mssql_field_length (int result [, int offset])
```

mssql_field_name

(PHP3 >= 3.0.3, PHP4 >= 4.0b4)

mssql_field_name -- Get the name of a field

Description

```
int mssql_field_name (int result [, int offset])
```

[Zurück](#)[Anfang](#)[Vor](#)[mssql_field_length](#)[Hoch](#)[mssql_field_seek](#)

mssql_field_seek

(PHP3 , PHP4)

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

(PHP3 >= 3.0.3, PHP4 >= 4.0b4)

mssql_field_type -- Get the type of a field

Description

string **mssql_field_type** (int result [, int offset])

[Zurück](#)[Anfang](#)[Vor](#)

mssql_field_seek

[Hoch](#)

mssql_free_result

mysql_free_result

(PHP3 , PHP4)

mysql_free_result -- Free result memory

Description

```
int mysql_free_result (int result)
```

mysql_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, you may call **mysql_free_result()** with the result identifier as an argument and the associated result memory will be freed.

mssql_get_last_message

(PHP3 , PHP4)

mssql_get_last_message -- Returns the last message from server (over min_message_severity?)

Description

string **mssql_get_last_message** (void)

[Zurück](#)[Anfang](#)[Vor](#)

mssql_free_result

[Hoch](#)

mssql_min_error_severity

mssql_min_error_severity

(PHP3 , PHP4)

mssql_min_error_severity -- Sets the lower error severity

Description

```
void mssql_min_error_severity (int severity)
```

[Zurück](#)[Anfang](#)[Vor](#)[mssql_get_last_message](#)[Hoch](#)[mssql_min_message_severity](#)

mssql_min_message_severity

(PHP3 , PHP4)

mssql_min_message_severity -- Sets the lower message severity

Description

```
void mssql_min_message_severity (int severity)
```

mssql_num_fields

(PHP3 , PHP4)

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

(PHP3 , PHP4)

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

(PHP3 , PHP4)

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

(PHP3 , PHP4)

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

(PHP3 , PHP4)

mssql_result -- Get result data

Description

`int mssql_result (int result, int i, mixed field)`

Returns: The contents of the cell at the row and offset in the specified MS SQL result set.

Mssql_result() returns the contents of one cell from a MS SQL 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 **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_select_db

(PHP3 , PHP4)

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\(\)](#)

XXXVI. Sonstige Funktionen

Diese Funktionen stehen hier, weil sie in keine andere Kategorie passen.

Inhaltsverzeichnis

[connection_aborted](#) Gibt true, wenn die Client-Verbindung beendet wurde

[connection_status](#) Gibt den Verbindungs-Status als Bit-Feld

[connection_timeout](#) Gibt true, wenn das Skript seinen Time-Out erreicht hat

[define](#) Definiert eine Konstante

[defined](#) Prüft, ob eine angegebene Konstante existiert

[die](#) Gibt eine Nachricht aus und beendet das aktuelle Skript

[eval](#) Wertet einen String aus, als wäre er PHP-Code

[exit](#) Beenden des aktuellen Skripts

[func_get_arg](#) Gibt einen Eintrag aus einer Liste von Argumenten zurück

[func_get_args](#) Gibt ein Array zurück, das die Liste der einer Funktion übergebenen Argumente enthält

[func_num_args](#) Gibt die Anzahl der einer Funktion übergebenen Argumente zurück

[function_exists](#) Falls die angegebene Funktion definiert ist, wird true zurück gegeben

[get_browser](#) Ermittelt die Möglichkeiten des Browsers eines Benutzers

[ignore_user_abort](#) Stellt ein, ob der Verbindungsabbruch eines Clients die Skript-Ausführung abbrechen soll

[iptcparse](#) Übersetzt einen binären IPTC-<http://www.xe.net/iptc/>-Block in einzelne Tags.

[leak](#) Speicher-Verbrauch

[pack](#) Packt Daten in einen Binär-String

[register_shutdown_function](#) Registriert eine Funktion zur Ausführung beim Skript-Abschluss

[serialize](#) Erzeugt ein speicherbares Datenformat

[sleep](#) Programmverzögerung

[uniqid](#) Erzeugt eine eindeutige ID

[unpack](#) Entpackt die Daten eines Binär-Strings

[unserialize](#) Erzeugt aus einem gespeicherten Datenformat einen Wert in PHP

[usleep](#) Programm-Verzögerung in Mikrosekunden

[highlight_string](#) Hervorhebung der Syntax eines Strings

[highlight_file](#) Syntax-Hervorhebung für eine Datei

[show -source](#) Syntax-Highlighting einer Datei

[Zurück](#)

mssql_select_db

[Anfang](#)

[Hoch](#)

[Vor](#)

connection_aborted

connection_aborted

(PHP3 >= 3.0.7, PHP4 >= 4.0b4)

connection_aborted -- Ergibt true, wenn die Client-Verbindung beendet wurde

Beschreibung:

```
int connection_aborted (void )
```

Gibt true zurück, wenn der Client nicht mehr verbunden ist. Siehe auch die [Connection Handling](#)-Beschreibung im [Features](#)-Kapitel für weitere Erläuterungen.

connection_status

(PHP3 >= 3.0.7, PHP4 >= 4.0b4)

connection_status -- Ergibt den Verbindungs-Status als Bit-Feld

Beschreibung:

```
int connection_status (void )
```

Gibt den Verbindungs-Status als Bit-Feld zurück. Siehe auch die [Connection Handling](#)-Beschreibung im [Features](#)-Kapitel für weitere Erläuterungen.

connection_timeout

(PHP3 >= 3.0.7, PHP4 >= 4.0b4)

connection_timeout -- Gibt true, wenn das Skript seinen Time-Out erreicht hat

Beschreibung:

```
int connection_timeout (void )
```

Gibt true zurück, wenn der Time-Out des Skripts erreicht ist. Siehe auch die [Connection Handling](#)-Beschreibung im [Features](#)-Kapitel für weitere Erläuterungen.

define

(PHP3 , PHP4)

define -- Definiert eine Konstante

Beschreibung:

`int define (string name, mixed value [, int case_insensitive])`

Definiert eine Konstante, ähnlich wie bei Variablen, ausser das:

- Eine Konstante nicht mit einem '\$' beginnt;
- Auf eine Konstante überall zugegriffen werden kann, ohne allerdings den Auswertungsregeln von Variablen zu unterliegen;
- Eine Konstante nicht umdefiniert oder zurück gesetzt werden kann, wenn sie einmal gesetzt wurde und
- eine Konstante nur eindimensionale Werte (also keine Arrays) enthalten kann.

Der Konstanten-Name wird durch *name* angegeben. Der Wert wird durch *value* gesetzt.

Wenn der optional mögliche dritte Parameter *case_insensitive* den Wert *1* hat, wird die Konstante zwischen Groß- und Kleinschreibung unterschieden. Das ist auch die Voreinstellung (es wird dann also zwischen KONSTANTE und Konstante unterschieden).

Die Funktionalität des dritten Parameters scheint z.Zt. noch nicht richtig umgesetzt worden zu sein (Stand 03/2000).

Beispiel 1 Definition von Konstanten:

```
<?php
define ("KONSTANTE", "Hallo Welt.");
echo KONSTANTE; // outputs "Hallo Welt.";
?>
```

Define() gibt bei erfolgreicher Zuweisung true, sonst false zurück.

Siehe auch unter [defined\(\)](#) und im Abschnitt [Konstanten](#).

define

connection_timeout

[Hoch](#)

defined

defined

(PHP3 , PHP4)

defined -- Prüft, ob eine angegebene Konstante existiert

Beschreibung:

int **defined** (string name)

Wenn die angegebene Konstante *name* existiert, wird true zurück gegeben, ansonsten false.

Siehe auch [define\(\)](#) und den Abschnitt [Konstanten](#).

die

(unknown)

die -- Gibt eine Nachricht aus und beendet das aktuelle Skript

Beschreibung:

void **die** (string message)

Dieser Sprach-Konstrukt erzeugt eine Meldung und bricht das Parsing / die Abarbeitung des Skriptes ab (ohne Rückkehrmöglichkeit).

Beispiel 1 "Die"-Beispiel:

```
<?php
$filename = '/pfad/zu/dateiname';
$file = fopen ($filename, 'r')
    or die ("Kann die Datei ($filename) nicht öffnen.");
?>
```

Beachten sie auch die Funktion [exit\(\)](#).

eval

(unknown)

eval -- Wertet einen String aus, als wäre er PHP-Code

Beschreibung:

void **eval** (string code_str)

Eval() versucht, den in *code_str* enthaltenen String als PHP-Code auszuführen. Abgesehen von anderen Zwecken kann dies nützlich sein, um Code auszuführen, der aus einer Datenbank ausgelesen wird.

Beim Gebrauch von **eval()** gilt es einige Faktoren zu beachten. Die Zeichenkette *code_str* muss gültigen PHP-Code enthalten, einschließlich syntaktisch richtiger Begrenzungszeichen (also incl. ";"-Abschluss-Zeichen für einzelne Anweisungen, sonst bricht der PHP-Parser in der folgenden Zeile mit einem Fehler ab.

Beachten sie, dass die innerhalb von **eval()** vorgenommenen Variablen-Zuweisungen nach Abarbeitung auch im Hauptskript ihre Gültigkeit behalten.

Beispiel 1 Eval()-Beispiel (einfache Texteingfügung):

```
<?php
$string = 'Bierglas';
$name = 'Binding-Lager';
$str = 'Das ist mein $string, voll mit $name.<br>';
echo $str;
eval ("\$str = \"\$str\";");
echo $str;
?>
```

Das obige Beispiel gibt aus:

```
Das ist mein $string, voll mit $name.
Das ist mein Bierglas, voll mit Binding-Lager.
```

Beachten sie bitte das Setzen des Begrenzungs-Zeichens ";" in *code_str* sowie den Gebrauch des "\"-Zeichens.

[Zurück](#)
die

[Anfang](#)
[Hoch](#)

[Vor](#)
exit

exit

(unknown)

exit -- Beenden des aktuellen Skripts

Beschreibung:

```
void exit(void);
```

Mit **exit()** beenden sie - ohne Möglichkeit der Rückkehr - die Skriptausführung.

Beachten sie auch die Funktion [die\(\)](#).

func_get_arg

(PHP4 >= 4.0b4)

`func_get_arg` -- Gibt einen Eintrag aus einer Liste von Argumenten zurück

Beschreibung:

`int func_get_arg (int arg_num)`

Gibt das Argument des an der *arg_num* stehenden Stelle in eine benutzer-definierte Liste zurück. Der Zähler hierfür beginnt bei 0. Der Aufruf von **func_get_arg()** ausserhalb einer Funktions-Definition erzeugt eine Warnung.

Ist *arg_num* grösser als die Anzahl der vorhandenen Argumente, wird eine Warnung aus- und false zurück gegeben.

```
<?php
function foo() {
    $numargs = func_num_args();
    echo "Anzahl der Argumente: $numargs<br>\n";
    if ($numargs >= 2) {
        echo "Das 2. Argument ist: " . func_get_arg (1) . "<br>\n";
    }
}

foo (1, 2, 3);
?>
```

func_get_arg() kann in Verbindung mit [func_num_args\(\)](#) und [func_get_args\(\)](#) gebraucht werden, um benutzer-definierte Funktionen mit variabler Argument-Liste zu erstellen.

Anmerkung: Diese Funktion steht ab PHP 4 zur Verfügung.

func_get_args

(PHP4 >= 4.0b4)

`func_get_args` -- Gibt ein Array zurück, das die Liste der einer Funktion übergebenen Argumente enthält

Beschreibung:

`int func_get_args (void )`

In einem Array wird jedes Element zurück gegeben, das einer benutzer-definierten Funktion übergeben wurde. Sollten sie dies ausserhalb einer eigenen (benutzer-definierten) Funktion anwenden, wird eine Warnung ausgegeben.

```
<?php
function foo() {
    $numargs = func_num_args();
    echo "Anzahl der Argumente: $numargs<br>\n";
    if ($numargs >= 2) {
        echo "Das 2. Argument ist: " . func_get_arg (1) . "<br>\n";
    }
    $arg_list = func_get_args();
    for ($i = 0; $i < $numargs; $i++) {
        echo "Argument $i ist: " . $arg_list[$i] . "<br>\n";
    }
}

foo (1, 2, 3);
?>
```

Func_get_args() kann in Verbindung mit [func_num_args\(\)](#) und [func_get_arg\(\)](#) verwendet werden, um einer benutzer-definierten Funktion eine Argument-Liste mit variabler Länge zu übergeben.

Anmerkung: Diese Funktion steht ab PHP 4 zur Verfügung.

func_num_args

(PHP4 >= 4.0b4)

`func_num_args` -- Gibt die Anzahl der einer Funktion übergebenen Argumente zurück

Beschreibung:

`int func_num_args (void )`

Die einer benutzer-definierten Funktion übergebene Anzahl an Argumenten wird zurück gegeben. Falls **func_num_args()** ausserhalb einer benutzer-definierten Funktion Verwendung findet, wird eine Warnung ausgegeben.

```
<?php
function foo() {
    $numargs = func_num_args();
    echo "Anzahl der Argumente: $numargs\n";
}

foo (1, 2, 3); // Ausgabe: 'Anzahl der Argumente: 3'
?>
```

func_num_args() kann in Verbindung mit [func_get_arg\(\)](#) und [func_get_args\(\)](#) benutzt werden, um einer benutzer-definierten Funktion eine variable Anzahl von Argumenten zu übergeben.

Anmerkung: Diese Funktion wurde in PHP 4 hinzu gefügt.

function_exists

(PHP3 >= 3.0.7, PHP4)

function_exists -- Falls die angegebene Funktion definiert ist, wird true zurück gegeben

Beschreibung:

int **function_exists** (string function_name)

Prüft die Liste der definierten Funktionen auf das Vorhanden sein der Funktion mit dem Namen *function_name* . Sofern vorhanden wird true, sonst false zurück gegeben.

[Zurück](#)[Anfang](#)[Vor](#)

func_num_args

[Hoch](#)

get_browser

get_browser

(PHP3 , PHP4)

get_browser -- Ermittelt die Möglichkeiten des Browsers eines Benutzers

Beschreibung:

object **get_browser** ([string user_agent])

Get_browser() ermittelt die Möglichkeiten des auf das PHP-Skript zugreifenden Browsers. Dies geschieht durch Auslesen der zum Browser vorhandenen Werte aus der Datei `browscap.ini`. Per Voreinstellung wird der Wert von `$HTTP_USER_AGENT` als Browser-Typ benutzt. Dieses Verhalten können sie ändern (so z.B. den Namen der INI-Datei), indem sie den optionalen *user_agent*-Parameter an die **get_browser()**-Funktion übergeben.

Die zurück gegebene Information ist ein Objekt, das verschiedene Datenelemente enthält. Dazu gehören z.B. die Haupt- und Unterversions-Nummer des Browsers sowie true / false-Werte für Browser-*Fähigkeiten* (u.a. Frame-, Java- und Cookies-Unterstützung). Nicht gemeint ist hier die aktuell vorhandene Nutzung. Ist der Browser so konfiguriert, dass er z.Zt. keine Cookies annehmen soll, wird trotzdem true in Bezug auf die Cookie-Fähigkeiten zurückgegeben, wenn in der Datei `browscap.ini` der entsprechende Wert true ist.

Da in `browscap.ini` Informationen vieler Browser stehen, obliegt es dem Skript-Schreiber, diese Datei auf dem Laufenden zu halten. Das Format dieser Datei sollte selbsterklärend sein.

Das folgende Beispiel zeigt, wie man alle verfügbaren Informationen zu einem gerade benutzten Browser anzeigen könnte:

Beispiel 1 Get_browser()-Beispiel:

```

<?php
function list_array ($array) {
    while (list ($key, $value) = each ($array)) {
        $str .= "<b>$key:</b> $value<br>\n";
    }
    return $str;
}
echo "$HTTP_USER_AGENT<hr>\n";
$browser = get_browser();
echo list_array ((array) $browser);
?>

```

Die Ausgabe des obigen Skripts könnte so aussehen:

```

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> Unknown<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>authenticcodeupdate:</b> <br>
<b>msn:</b> <br>

```

Damit dieses Skript richtig funktioniert, muss die [browscap](#)-Einstellung ihrer Konfigurations-Datei auf die richtige `browscap.ini`-Datei zeigen.

Weitere Informationen (incl. Bezugsquellen für `browscap.ini`-Dateien) finden sie in der PHP-FAQ unter <http://www.php.net/FAQ.php>.

get_browser

function_exists

[Hoch](#)

ignore_user_abort

ignore_user_abort

(PHP3 >= 3.0.7, PHP4 >= 4.0b4)

`ignore_user_abort` -- Stellt ein, ob der Verbindungsabbruch eines Clients die Skript-Ausführung abbrechen soll

Beschreibung:

```
int ignore_user_abort ([int setting])
```

Mit dieser Funktion wird der Wert dafür gesetzt, ob der Abbruch einer Client-Verbindung die weitere Abarbeitung eines Skripts beenden soll. Es wird der vorherige Wert zurück gegeben. Die Funktion kann auch ohne Argument aufgerufen werden, um nur die aktuelle Einstellung zu erfahren. Die komplette Beschreibung des Verbindungs-Handling von PHP können sie im Abschnitt [Connection-Handling](#) des Kapitels [Features](#) nachlesen. Dort werden auch die Werte für den optionalen Parameter *setting* näher erläutert.

iptcparse

(PHP3 >= 3.0.6, PHP4)

iptcparse -- Übersetzt einen binären IPTC-<http://www.xe.net/iptc/>-Block in einzelne Tags.

Beschreibung:

array **iptcparse** (string iptcblock)

Mit dieser Funktion übertragen sie einen binären IPTC-Block in seine einzelnen Tags. Es wird ein Array zurück gegeben. In diesem bilden die Tag-Markierungen den Index, die Werte stellen den Inhalt dar. Bei einem Fehler oder wenn keine IPTC-Daten gefunden wurden, wird false zurück gegeben. Unter **GetImageSize()** finden sie ein Beispiel.

leak

(PHP3 , PHP4)

leak -- Speicher-Verbrauch

Beschreibung:

void **leak** (int bytes)

Leak() belegt die angegebene Speichergrösse.

Diese Funktion ist hilfreich beim Debuggen des Speicher-Managers, der "leaked"-Speicher nach Abarbeitung aller Anforderungen wieder automatisch frei gibt.

pack

(PHP3 , PHP4)

pack -- Packt Daten in einen Binär-String

Beschreibung:

string **pack** (string format [, mixed args ...])

Diese Funktion packt ein oder mehrere Argumente - unter Beachtung der Argument-Formate - in einen Binär-String. In dem zurück gegebenen Binär-String stehen anschließend die Daten.

Die Idee entstammt Perl. Alle Formatierungs-Anweisungen funktionieren genau wie dort. In PHP fehlen aber einige Format-Codes von Perl (z.B. "u"). Der Format-String besteht aus Format-Codes, gefolgt von einem optionalen Wiederholungs-Argument. Dieses Argument kann ein Integer-Wert oder ein "*" (für Wiederholung bis ans Ende der Daten) sein. Bei "a", "A", "h" und dem "H" Format-Code gibt das Wiederholungs-Argument an, wie viele gleiche Zeichen folgen. Im Zusammenhang mit "@" gibt das Wiederholungs-Argument die absolute Position an, ab der das nächste Zeichen steht. Bei allen anderen steht der Wiederholungs-Zähler für die Anzahl der benutzten Daten-Argumente, die in den Binär-String gepackt werden sollen. Z. Zt. sind folgende Codes implementiert:

- a mit NUL gefüllte Zeichenkette (String)
- A mit Leerzeichen (SPACE) gefüllte Zeichenkette (String)
- h Hex Zeichenkette (String), unterer Halbwert zuerst
- H Hex Zeichenkette (String), höherer Halbwert zuerst
- c signed char
- C unsigned char
- s Short-Typ mit Vorzeichen (immer 16 Bit, Byte-Folge maschinen-abhängig)
- S Short-Typ ohne Vorzeichen (immer 16 Bit, Byte-Folge maschinen-abhängig)
- n Short-Typ ohne Vorzeichen (immer 16 Bit, big endian byte order)
- v Short-Typ ohne Vorzeichen (immer 16 Bit, little endian byte order)
- i Integer mit Vorzeichen (Größe und Byte-Folge maschinen-abhängig)
- I Integer-Typ ohne Vorzeichen (Größe und Byte-Folge maschinen-abhängig)

- l Long-Typ mit Vorzeichen (immer 32 Bit, Byte-Folge maschinen-abhängig)
- L Long-Typ ohne Vorzeichen (immer 32 Bit, Byte-Folge maschinen-abhängig)
- N Long-Typ ohne Vorzeichen (immer 32 Bit, big endian byte order)
- V Long-Typ ohne Vorzeichen (immer 32 Bit, little endian byte order)
- f Fließkomma-Typ (maschinen-abhängige Größe und Wiedergabe)
- d Double-Typ (maschinen-abhängige Größe und Wiedergabe)
- x NUL Byte
- X Sichern eines Bytes
- @ NUL-Auffüllung bis zur absoluten Position

Beispiel 1 Mit Pack formatierter String

```
$binarydata = pack ("nvc*", 0x1234, 0x5678, 65, 66);
```

Der sich daraus ergebende Binär-String ist 6 Bytes lang und enthält: 0x12, 0x34, 0x78, 0x56, 0x41, 0x42.

Bedenken sie, dass der Unterschied zwischen vorzeichen-losen und vorzeichen-behafteten Werten nur Einfluss auf die Funktion [unpack\(\)](#) hat, wogegen dies bei der Funktion **pack()** egal ist.

Beachten sie auch, dass PHP intern Ganzzahl-Werte als vorzeichen-behaftete Werte speichert, deren Umfang vom Maschinen-Typ abhängig ist. Geben sie PHP einen vorzeichen-losen Ganzzahl-Wert, der zu groß ist für diese Art der Speicherung, wird er ins Double-Format umgesetzt, was oft zu unerwünschten Resultaten führt.

[Zurück](#)[leak](#)[Anfang](#)[Hoch](#)[Vor](#)[register_shutdown_function](#)

register_shutdown_function

(PHP3 >= 3.0.4, PHP4)

register_shutdown_function -- Registriert eine Funktion zur Ausführung beim Skript-Abschluss

Beschreibung:

int register_shutdown_function (string func)

Die mit *func* benannte Funktion wird im Skript zwecks Ausführung an dessen Ende eingetragen.

Bekannte Problemfälle:

Da innerhalb dieser Funktion keinerlei Ausgabe im Browser erzeugt werden, können sie die registrierte Funktion nicht mit print oder echo usw. debuggen.

serialize

(PHP3 >= 3.0.5, PHP4)

serialize -- Erzeugt ein speicherbares Datenformat

Beschreibung:

string **serialize** (mixed value)

Serialize() gibt eine Zeichenkette (String) zurück, die eine Byte-Stream entsprechende Wiedergabe von *value* enthält und beliebig abgespeichert werden kann.

Diese Funktion dient der Speicherung oder Übergabe von PHP-Werten, ohne dass diese ihren Wert oder ihre Struktur verlieren.

Um eine serialisierte Zeichenkette wieder als PHP-Wert verfügbar zu machen steht die [unserialize\(\)](#)-Funktion zur Verfügung. **Serialize()** kann mit den Typen integer, double, string, array (mehrdimensional) und object umgehen. Beim Objekt werden die Eigenschaften serialisiert, die Methoden gehen aber verloren.

Beispiel 1 Serialize()-Beispiel:

```
// $session_data enthält ein mehrdimensionales Array mit
// Session-Informationen des aktuellen Benutzers. Wir benutzen
// serialize() um diese Infos am Ende der Anfrage in eine
// Datenbank zu speichern.

$conn = odbc_connect ("webdb", "php", "moorhuhn");
$stmt = odbc_prepare ($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)) {
        /* Etwas lief schief. Fluche, heule und stöhne. */
    }
}
```

sleep

(PHP3 , PHP4)

sleep -- Programmverzögerung

Beschreibung:

void **sleep** (int seconds)

Die **sleep()**-Funktion unterbricht die weitere Skript-Ausführung für die mit *seconds* angegebene Anzahl in Sekunden.

Siehe auch [usleep\(\)](#).

uniqid

(PHP3 , PHP4)

uniqid -- Erzeugt eine eindeutige ID

Beschreibung:

`int uniqid (string prefix [, boolean lcg])`

Uniqid() gibt eine eindeutige ID mit Präfix zurück. Diese basiert auf der aktuellen Zeit in Mikrosekunden. Das Präfix kann hilfreich sein, wenn sie z. B. auf mehreren Servern gleichzeitig ID's erzeugen, was ja im ungünstigsten Fall zur selben Mikrosekunde geschehen kann. Der *prefix*-Parameter kann bis zu 114 Zeichen lang sein.

Ist der optionale Parameter *lcg* true, wird **uniqid()** am Ende des Rückgabewertes zusätzliche "LCG"-Zustandswerte anfügen, durch die das Ergebnis noch eindeutiger wird.

Bei leerem *prefix*-Parameter ist der zurück gegebene String 13 Zeichen lang. Ist der Parameter *lcg* true, sind es 23 Zeichen.

Anmerkung: Der *lcg*-Parameter ist erst ab PHP 4 bzw. PHP 3.0.13 und später verfügbar.

Wenn sie eindeutige ID's bzw. eindeutige Token benötigen und beabsichtigen, diese Token über das Netzwerk an den Benutzer weiter zu leiten (z.B. als Session-Cookies), ist es ratsam, wenn sie wie folgt vorgehen:

```
$token = md5 (uniqid ("")); // kein Zufalls-Anteil = schlecht  
$better_token = md5 (uniqid (rand())); // besser, da schwer zu erraten
```

Hierdurch wird eine 32-Zeichen-ID (128 Bit-Hex-Wert) erzeugt, die nur sehr schwer vorhersehbar ist.

unpack

(PHP3 , PHP4)

unpack -- Entpackt die Daten eines Binär-Strings

Beschreibung:

array **unpack** (string format, string data)

Unpack() überträgt die Daten eines Binär-Strings in ein Array unter Berücksichtigung des *format*-Parameters. Das Array wird als Ergebnis des Funktionsaufrufs zurück gegeben.

Unpack() funktioniert etwas anders als man es von Perl her kennt, da sich die zurück gegebenen Daten in einem assoziativen Array befinden. Dies erreichen sie nur, wenn sie die verschiedenen Format-Codes benennen und sie mit einem "/" voneinander trennen.

Beispiel 1 Unpack() formatierter String

```
$array = unpack ("c2chars/nint", $binarydata);
```

Das resultierende Array wird die Einträge "chars1", "chars2" und "int" enthalten.

Die Format-Codes werden unter [pack\(\)](#) näher erläutert.

Beachten sie, dass PHP Integer-Werte intern mit Vorzeichen speichert. Wenn sie einen grossen vorzeichen-losen Long-Wert entpacken und er von der selben Grösse ist, die PHP für dessen Speicherung verwendet, wird das Ergebnis ein negativer Wert sein (auch wenn sie dieses als vorzeichen-los zu entpacken angegeben haben).

unserialize

(PHP3 >= 3.0.5, PHP4)

unserialize -- Erzeugt aus einem gespeicherten Datenformat einen Wert in PHP

Beschreibung:

mixed **unserialize** (string str)

Unserialize() nimmt eine einzelne Variable aus einer serialisierten Datenmenge (siehe unter [serialize\(\)](#)) und wandelt sie in einen PHP-Wert um. Dieser Wert wird als Funktionsergebnis zurück gegeben. Er kann vom Typ integer, double, string, array oder object sein. Wurde ein Objekt serialisiert, können dessen Methoden nicht zurück gegeben werden.

Beispiel 1 Unserialize()-Beispiel:

```
// Als Beispiel benutzen wir unserialize() um Daten aus einer
// Datenbank nach $session_data zu laden. Das Beispiel vervollständigt
// das unter serialize() beschriebene.

$conn = odbc_connect ("webdb", "php", "moorhuhn");
$stmt = odbc_prepare ($conn, "SELECT data FROM sessions WHERE id = ?");
$sqldata = array ($PHP_AUTH_USER);
if (!odbc_execute ($stmt, &$sqldata) || !odbc_fetch_into ($stmt, &$tmp)) {
    // wenn execute oder fetch fehlschlägt, initialisiere ein leeres Array
    $session_data = array();
} else {
    // die serialisierten Daten sollten nun in $tmp[0] stehen
    $session_data = unserialize ($tmp[0]);
    if (!is_array ($session_data)) {
        // etwas ging schief, also wieder ein leeres Array initialisieren
        $session_data = array();
    }
}
```

usleep

(PHP3 , PHP4)

usleep -- Programm-Verzögerung in Mikrosekunden

Beschreibung:

void **usleep** (int micro_seconds)

Mittels der Funktion [sleep\(\)](#) verzögern sie die weitere Skript-Ausführung um die gegebene Anzahl an *micro_seconds*.

Beachten sie auch die Funktion [sleep\(\)](#).

Anmerkung: Dies Funktion steht unter Windows-Systemen nicht zur Verfügung.

highlight_string

(PHP4)

highlight_string -- Hervorhebung der Syntax eines Strings

Beschreibung:

void **highlight_string** (string str)

Die Funktion **highlight_string()** gibt den String *str* mit hervorgehobener Syntax (Syntax highlighting) aus. Dabei werden die Farben des in PHP eingebauten Syntax-Highlighter benutzt.

Siehe auch [highlight_file\(\)](#) und [show_source\(\)](#).

highlight_file

(PHP4)

highlight_file -- Syntax-Hervorhebung für eine Datei

Beschreibung:

void **highlight_file** (string filename)

Die Funktion **highlight_file()** erzeugt die Ausgabe des Codes der Datei *filename* mit hervorgehobener Syntax. Dabei werden die Farben des in PHP eingebauten Syntax-Highlighter benutzt.

Beispiel 1 Erzeugung einer hervorgehobenen URL

Zur Erzeugung einer URL, die Code eines ihm übermittelten Skripts hervorgehoben darstellen kann, machen wir Gebrauch von der "ForceType"-Anweisung des Apachen um eine richtige URL zu erzeugen. Zusätzlich benutzen wir die Funktion **highlight_file()**, um das Code-Listing wie gewünscht an zu zeigen.

In ihrer httpd.conf sollten sie folgendes hinzu fügen:

```
<Location /source>
    ForceType application/x-httpd-php
</Location>
```

Jetzt erzeugen sie eine Datei namens "source" und speichern es in dem Root-Verzeichnis ihrer Web-Seite.

```
<HTML>
<HEAD>
<TITLE>Source Display</TITLE>
</HEAD>
<BODY BGCOLOR="white">
<?php
    $script = getenv ("PATH_TRANSLATED");
    if(!$script) {
        echo "<BR><B>Fehler: Skript-Name nötig</B><BR>";
    } else {
        if (ereg("(\\.php|\\.inc)$",$script)) {
            echo "<H1>Source von: $PATH_INFO</H1>\\n<HR>\\n";
            highlight_file($script);
        } else {
            echo "<H1>ERROR: Nur PHP oder Include-Skripte sind erlaubt</H1>";
        }
    }
    echo "<HR>Erzeugt: ".date("Y/M/d H:i:s",time());
?>
```

highlight_file

```
</BODY>  
</HTML>
```

Jetzt können sie die u.a. URL benutzen, um eine farblich hervorgehobene Version des Skriptes "/path/to/script.php" innerhalb ihrer Web-Seite zu zeigen.

```
http://ihr.server.de/source/pfad/zu/script.php
```

Siehe auch [highlight_string\(\)](#) und [show_source\(\)](#).

[Zurück](#)

highlight_string

[Anfang](#)

[Hoch](#)

[Vor](#)

show_-source

show_-source

(unknown)

show_-source -- Syntax-Highlighting einer Datei

Beschreibung:

void **show_source** (string filename)

Die Funktion [show_source\(\)](#) erzeugt die Ausgabe des Codes der Datei *filename* mit hervorgehobener Syntax. Dabei werden die Farben des in PHP eingebauten Syntax-Highlighter benutzt.

Anmerkung: Diese Funktion ist ein Alias der Funktion [highlight_file\(\)](#).

Siehe auch [highlight_string\(\)](#) und [highlight_file\(\)](#).

XXXVII. mSQL functions

Inhaltsverzeichnis

- [mysql](#) Send mSQL query
- [mysql_affected_rows](#) Returns number of affected rows
- [mysql_close](#) Close mSQL connection
- [mysql_connect](#) Open mSQL connection
- [mysql_create_db](#) Create mSQL database
- [mysql_createdb](#) Create mSQL database
- [mysql_data_seek](#) Move internal row pointer
- [mysql_dbname](#) Get current mSQL database name
- [mysql_drop_db](#) Drop (delete) mSQL database
- [mysql_dropdb](#) Drop (delete) mSQL database
- [mysql_error](#) Returns error message of last mysql call
- [mysql_fetch_array](#) Fetch row as array
- [mysql_fetch_field](#) Get field information
- [mysql_fetch_object](#) Fetch row as object
- [mysql_fetch_row](#) Get row as enumerated array
- [mysql_fieldname](#) Get field name
- [mysql_field_seek](#) Set field offset
- [mysql_fieldtable](#) Get table name for field
- [mysql_fieldtype](#) Get field type
- [mysql_fieldflags](#) Get field flags
- [mysql_fieldlen](#) Get field length
- [mysql_free_result](#) Free result memory
- [mysql_freeresult](#) Free result memory
- [mysql_list_fields](#) List result fields
- [mysql_listfields](#) List result fields
- [mysql_list_dbs](#) List mSQL databases on server
- [mysql_listdbs](#) List mSQL databases on server

[mysql_list_tables](#) List tables in an mSQL database
[mysql_listtables](#) List tables in an mSQL database
[mysql_num_fields](#) Get number of fields in result
[mysql_num_rows](#) Get number of rows in result
[mysql_numfields](#) Get number of fields in result
[mysql_numrows](#) Get number of rows in result
[mysql_pconnect](#) Open persistent mSQL connection
[mysql_query](#) Send mSQL query
[mysql_regcase](#) Make regular expression for case insensitive match
[mysql_result](#) Get result data
[mysql_select_db](#) Select mSQL database
[mysql_selectdb](#) Select mSQL database
[mysql_tablename](#) Get table name of field

[Zurück](#)

[Anfang](#)

[Vor](#)

show_-source

[Hoch](#)

mysql

mysql

(PHP3 , PHP4)

mysql -- Send mSQL query

Description

int **mysql** (string database, string query, int link_identifier)

Returns a positive mSQL query identifier to the query result, or false on error.

mysql() 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 [mysql_connect\(\)](#) was called with no arguments (see [mysql_connect\(\)](#)).

mysql_affected_rows

(PHP3 >= 3.0.6, PHP4)

mysql_affected_rows -- Returns number of affected rows

Description

`int mysql_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: [mysql_query\(\)](#).

mysql_close

(PHP3 , PHP4)

mysql_close -- Close mSQL connection

Description

```
int mysql_close (int link_identifier)
```

Returns true on success, false on error.

mysql_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.

mysql_close() will not close persistent links generated by [mysql_pconnect\(\)](#).

See also: [mysql_connect\(\)](#) and [mysql_pconnect\(\)](#).

mysql_connect

(PHP3 , PHP4)

mysql_connect -- Open mSQL connection

Description

`int mysql_connect (string hostname)`

Returns a positive mSQL link identifier on success, or false on error.

mysql_connect() establishes a connection to a mSQL server. The hostname argument is optional, and if it's missing, localhost is assumed.

In case 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 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\(\)](#).

See also [mysql_pconnect\(\)](#), [mysql_close\(\)](#).

mysql_create_db

(PHP3 , PHP4)

mysql_create_db -- Create mSQL database

Description

int mysql_create_db (string database name [, int link_identifier])

mysql_create_db() attempts to create a new database on the server associated with the specified link identifier.

See also: [mysql_drop_db\(\)](#).

mysql_createdb

(PHP3 , PHP4)

mysql_createdb -- Create mSQL database

Description

int **mysql_createdb** (string database name [, int link_identifizier])

Identical to [mysql_create_db\(\)](#).

mysql_data_seek

(PHP3 , PHP4)

mysql_data_seek -- Move internal row pointer

Description

`int mysql_data_seek (int query_identifier, int row_number)`

Returns true on success, false on failure.

mysql_data_seek() moves the internal row pointer of the mSQL result associated with the specified query identifier to pointer to the specified row number. The next call to [mysql_fetch_row\(\)](#) would return that row.

See also: [mysql_fetch_row\(\)](#).

mysql_dbname

(PHP3 , PHP4)

mysql_dbname -- Get current mSQL database name

Description

string **mysql_dbname** (int query_identifier, int i)

Mysql_dbname() returns the database name stored in position *i* of the result pointer returned from the [mysql_listdbs\(\)](#) function. The [mysql_numrows\(\)](#) function can be used to determine how many database names are available.

mysql_drop_db

(PHP3 , PHP4)

mysql_drop_db -- Drop (delete) mSQL database

Description

int mysql_drop_db (string database_name, int link_identifier)

Returns true on success, false on failure.

Mysql_drop_db() attempts to drop (remove) an entire database from the server associated with the specified link identifier.

See also: [mysql_create_db\(\)](#).

mysql_dropdb

(PHP3 , PHP4)

mysql_dropdb -- Drop (delete) mSQL database

Description

See [mysql_drop_db\(\)](#).

[Zurück](#)[Anfang](#)[Vor](#)

mysql_drop_db

[Hoch](#)

mysql_error

mysql_error

(PHP3 , PHP4)

mysql_error -- Returns error message of last mysql call

Description

string **mysql_error** ()

Errors coming back from the mSQL database backend no longer issue warnings. Instead, use these functions to retrieve the error string.

mysql_fetch_array

(PHP3 , PHP4)

mysql_fetch_array -- Fetch row as array

Description

```
int mysql_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.

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.

The second optional argument *result_type* in **mysql_fetch_array()** is a constant and can take the following values: MYSQL_ASSOC, MYSQL_NUM, and MYSQL_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 0 (or an empty string, or NULL).

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.

For further details, also see [mysql_fetch_row\(\)](#).

mysql_fetch_field

(PHP3 , PHP4)

mysql_fetch_field -- Get field information

Description

object **mysql_fetch_field** (int query_identifier, 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
- 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 [mysql_field_seek\(\)](#).

mysql_fetch_object

(PHP3 , PHP4)

mysql_fetch_object -- Fetch row as object

Description

int mysql_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.

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).

The optional second argument *result_type* in [mysql_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 [mysql_fetch_array\(\)](#), and almost as quick as [mysql_fetch_row\(\)](#) (the difference is insignificant).

See also: [mysql_fetch_array\(\)](#) and [mysql_fetch_row\(\)](#).

mysql_fetch_row

(PHP3 , PHP4)

mysql_fetch_row -- Get row as enumerated array

Description

array **mysql_fetch_row** (int query_identifier)

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 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 **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\(\)](#), and [mysql_result\(\)](#).

mysql_fieldname

(PHP3 , PHP4)

mysql_fieldname -- Get field name

Description

string **mysql_fieldname** (int query_identifier, int field)

Mysql_fieldname() returns the name of the specified field. *query_identifier* is the query identifier, and *field* is the field index. `mysql_fieldname($result, 2);` will return the name of the second field in the result associated with the result identifier.

mysql_field_seek

(PHP3 , PHP4)

mysql_field_seek -- Set field offset

Description

int **mysql_field_seek** (int query_identifier, int field_offset)

Seeks to the specified field offset. If the next call to [mysql_fetch_field\(\)](#) won't include a field offset, this field would be returned.

See also: [mysql_fetch_field\(\)](#).

mysql_fieldtable

(PHP3 , PHP4)

mysql_fieldtable -- Get table name for field

Description

int **mysql_fieldtable** (int query_idenfifier, int field)

Returns the name of the table *field* was fetched from.

[Zurück](#)[Anfang](#)[Vor](#)

mysql_field_seek

[Hoch](#)

mysql_fieldtype

mysql_fieldtype

(PHP3 , PHP4)

mysql_fieldtype -- Get field type

Description

string **mysql_fieldtype** (int query_identifier, int i)

Msql_fieldtype() is similar to the [mysql_fieldname\(\)](#) function. The arguments are identical, but the field type is returned. This will be one of "int", "string" or "real".

mysql_fieldflags

(PHP3 , PHP4)

mysql_fieldflags -- Get field flags

Description

string **mysql_fieldflags** (int query_identifizier, int i)

mysql_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).

mysql_fieldlen

(PHP3 , PHP4)

mysql_fieldlen -- Get field length

Description

```
int mysql_fieldlen (int query_identifier, int i)
```

Mysql_fieldlen() returns the length of the specified field.

[Zurück](#)[Anfang](#)[Vor](#)

mysql_fieldflags

[Hoch](#)

mysql_free_result

mysql_free_result

(PHP3 , PHP4)

mysql_free_result -- Free result memory

Description

```
int mysql_free_result (int query_identifier)
```

mysql_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.

mysql_freeresult

(PHP3 , PHP4)

mysql_freeresult -- Free result memory

Description

See [mysql_free_result\(\)](#)

[Zurück](#)[Anfang](#)[Vor](#)

mysql_free_result

[Hoch](#)

mysql_list_fields

mysql_list_fields

(PHP3 , PHP4)

mysql_list_fields -- List result fields

Description

int mysql_list_fields (string database, string tablename)

Mysql_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 [mysql_fieldflags\(\)](#), [mysql_fieldlen\(\)](#), [mysql_fieldname\(\)](#), and [mysql_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 \$phperrormsg, and unless the function was called as @mysql_list_fields() then this error string will also be printed out.

See also [mysql_error\(\)](#).

mysql_listfields

(PHP3 , PHP4)

mysql_listfields -- List result fields

Description

See [mysql_list_fields\(\)](#).

mysql_list_dbs

(PHP3 , PHP4)

mysql_list_dbs -- List mSQL databases on server

Description

```
int mysql_list_dbs(void);
```

mysql_list_dbs() will return a result pointer containing the databases available from the current msql daemon. Use the [mysql_dbname\(\)](#) function to traverse this result pointer.

mysql_listdbs

(PHP3 , PHP4)

mysql_listdbs -- List mSQL databases on server

Description

See [mysql_list_dbs\(\)](#).

[Zurück](#)[Anfang](#)[Vor](#)

mysql_list_dbs

[Hoch](#)

mysql_list_tables

mysql_list_tables

(PHP3 , PHP4)

mysql_list_tables -- List tables in an mSQL database

Description

```
int mysql_list_tables (string database)
```

Msql_list_tables() takes a database name and result pointer much like the [mysql\(\)](#) function. The [mysql_tablename\(\)](#) function should be used to extract the actual table names from the result pointer.

mysql_listtables

(PHP3 , PHP4)

mysql_listtables -- List tables in an mSQL database

Description

See [mysql_list_tables\(\)](#).

[Zurück](#)[Anfang](#)[Vor](#)

mysql_list_tables

[Hoch](#)

mysql_num_fields

mysql_num_fields

(PHP3 , PHP4)

mysql_num_fields -- Get number of fields in result

Description

```
int mysql_num_fields (int query_identifier)
```

Mysql_num_fields() returns the number of fields in a result set.

See also: [mysql\(\)](#), [mysql_query\(\)](#), [mysql_fetch_field\(\)](#), and [mysql_num_rows\(\)](#).

mysql_num_rows

(PHP3 , PHP4)

mysql_num_rows -- Get number of rows in result

Description

```
int mysql_num_rows (int query_identifier)
```

Mysql_num_rows() returns the number of rows in a result set.

See also: [mysql\(\)](#), [mysql_query\(\)](#), and [mysql_fetch_row\(\)](#).

mysql_numfields

(PHP3 , PHP4)

mysql_numfields -- Get number of fields in result

Description

```
int mysql_numfields (int query_identifier)
```

Identical to [mysql_num_fields\(\)](#).

mysql_numrows

(PHP3 , PHP4)

mysql_numrows -- Get number of rows in result

Description

```
int mysql_numrows(void);
```

Identical to [mysql_num_rows\(\)](#).

mysql_pconnect

(PHP3 , PHP4)

mysql_pconnect -- Open persistent mSQL connection

Description

int mysql_pconnect (string hostname)

Returns a positive mSQL persistent link identifier on success, or false on error.

Msql_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. 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()**).

This type of links is therefore called 'persistent'.

mysql_query

(PHP3 , PHP4)

mysql_query -- Send mSQL query

Description

`int mysql_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 [mysql_connect\(\)](#) was called, and use it.

Returns a positive mSQL query identifier on success, or false on error.

See also: [mysql\(\)](#), [mysql_select_db\(\)](#), and [mysql_connect\(\)](#).

mysql_regcase

(PHP3 , PHP4)

mysql_regcase -- Make regular expression for case insensitive match

Description

See [sql_regcase\(\)](#).

mysql_result

(PHP3 , PHP4)

mysql_result -- Get result data

Description

int mysql_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.

Mysql_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 **mysql_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: [mysql_fetch_row\(\)](#), [mysql_fetch_array\(\)](#), and [mysql_fetch_object\(\)](#).

mysql_select_db

(PHP3 , PHP4)

mysql_select_db -- Select mSQL database

Description

int mysql_select_db (string database_name, int link_identifier)

Returns true on success, false on error.

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, 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\(\)](#).

mysql_selectdb

(PHP3 , PHP4)

mysql_selectdb -- Select mSQL database

Description

See [mysql_select_db\(\)](#).

[Zurück](#)[Anfang](#)[Vor](#)

mysql_select_db

[Hoch](#)

mysql_tablename

mysql_tablename

(PHP3 , PHP4)

mysql_tablename -- Get table name of field

Description

string **mysql_tablename** (int query_identifier, int field)

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_numrows\(\)](#) function may be used to determine the number of tables in the result pointer.

Beispiel 1 Mysql_tablename() example

```
<?php
mysql_connect ("localhost");
$result = mysql_list_tables ("wisconsin");
$i = 0;
while ($i < mysql_numrows ($result)) {
    $tb_names[$i] = mysql_tablename ($result, $i);
    echo $tb_names[$i] . "<BR>";
    $i++;
}
?>
```

XXXVIII. MySQL Funktionen

Diese Funktionen erlauben den Zugriff auf einen MySQL Datenbank-Server.

Weiter Informationen zu MySQL gibt es unter <http://www.mysql.com/>.

In diesem Dokument werden ein Reihe von Begriffen benutzt, die hier kurz erläutert werden sollen.

Anfrageergebnis, Ergebnis

Die Rückgabe auf eine Anfrage an den Datenbankserver. Anfragen können sowohl Datenbankabfragen sein, die Teile des Datenbankinhalts umfassen oder Informationen über eine Datenbank oder den Datenbankserver liefern. Ist das Ergebnis eine Datenbankabfrage kann man es sich als Tabelle vorstellen, in der die Zeilen die Datensätze sind und die Felder den Spalten entsprechen. Ein Ergebnis besteht aus 0 oder beliebig vielen Datensätzen.

Ergebnis-Kennung

Eine Referenz auf ein Ergebnis. Über diese Kennung erfolgt grundsätzlich der Zugriff auf das Ergebnis.

Datensatz

Eine Zeile aus einem Anfrageergebnis bestehend aus den Werten der einzelnen Felder.

Datensatzzeiger

Ein interner Zeiger auf einen Datensatz in einem Anfrageergebnis. Dieser Zeiger bestimmt auf welchen Datensatz zugegriffen wird. Der Zeiger kann sowohl impliziet als auch expliziet verändert werden.

Feld

Entspricht einer Spalte im Anfrageergebnis.

Feldzeiger

Wie Datensatzzeiger bei Zugriffen auf Felder eines Anfrageergebnisses.

Verbindungs-Kennung

Eine Referenz auf eine Verbindung zum Datenbank-Server. Mit dieser Verbindung ist beispielsweise die Datenbank, der Benutzer, der Rechnername auf dem die Datenbank läuft und weiters verknüpft. Jeder Zugriff auf den Server nutzt diese Kennung. Wenn die Kennung beim Aufruf einer Funktion nicht angegeben wird, so wird die aktuelle Verbindungs-Kennung verwendet, die intern von PHP verwaltet wird.

Inhaltsverzeichnis

[mysql_affected_rows](#) Liefert die Anzahl betroffener Datensätze einer vorhergehenden MySQL

Operation

- [mysql_change_user](#) Ändert den zur Zeit angemeldeten Benutzer
 - [mysql_close](#) Schließt eine Verbindung zum Datenbank-Server
 - [mysql_connect](#) Öffnet eine Verbindung zum Datenbank-Server
 - [mysql_create_db](#) Erzeugt eine Datenbank
 - [mysql_data_seek](#) Bewegt internen Datensatz-Zeiger
 - [mysql_db_query](#) Absetzen einer Anfrage an die Datenbank
 - [mysql_drop_db](#) Entfernt eine Datenbank
 - [mysql_errno](#) Liefert die Fehlernummer einer zuvor ausgeführten Operation
 - [mysql_error](#) Liefert den Fehlertext der zuvor ausgeführten MySQL Operation
 - [mysql_fetch_array](#) Liefert einen Datensatz als assoziatives Array
 - [mysql_fetch_field](#) Liefert ein Objekt mit Feldinformationen aus einem Anfrageergebnis
 - [mysql_fetch_lengths](#) Liefert die Länge eines jeden Feldes in einem Datensatz
 - [mysql_fetch_object](#) Liefert einen Datensatz als Objekt
 - [mysql_fetch_row](#) Liefert einen Datensatz als indiziertes Array
 - [mysql_field_name](#) Liefert den Namen eines Feldes in einem Anfrageergebnis
 - [mysql_field_seek](#) Setzt den Feldzeiger auf ein bestimmtes Feld
 - [mysql_field_table](#) Liefert den Namen der Tabelle, die das genannte Feld enthält
 - [mysql_field_type](#) Liefert den Typ eines Feldes in einem Anfrageergebnis
 - [mysql_field_flags](#) Liefert die Flags eines Feldes in einem Anfrageergebnis
 - [mysql_field_len](#) Liefert die Länge eines Feldes
 - [mysql_free_result](#) Gibt den belegten Speicher wieder frei
 - [mysql_insert_id](#) Liefert die Kennung einer vorherigen INSERT-Operation
 - [mysql_list_fields](#) Listet die Felder einer Tabelle auf
 - [mysql_list_dbs](#) Liefert eine Liste der verfügbaren Datenbanken auf dem Server
 - [mysql_list_tables](#) Liefert eine Liste der Tabellen in einer Datenbank
 - [mysql_num_fields](#) Liefert die Anzahl der Felder in einem Ergebnis
 - [mysql_num_rows](#) Liefert die Anzahl der Datensätze im Ergebnis
 - [mysql_pconnect](#) Öffnet eine persistente Verbindung zum MySQL Server
 - [mysql_query](#) Sendet eine SQL Anfrage zum Datenbankserver
 - [mysql_result](#) Liefert Ergebnis
 - [mysql_select_db](#) Wählt eine Datenbank aus
 - [mysql_tablename](#) Liefert den Namen einer Tabelle
-

[Zurück](#)

mysql_tablename

[Anfang](#)

[Hoch](#)

[Vor](#)

mysql_affected_rows

mysql_affected_rows

(PHP3 , PHP4)

mysql_affected_rows -- Liefert die Anzahl betroffener Datensätze einer vorhergehenden MySQL Operation

Beschreibung

`int mysql_affected_rows ([int Verbindungs-Kennung])`

mysql_affected_rows() liefert die Anzahl betroffener Datensätze durch die letzte INSERT, UPDATE oder DELETE Anfrage an den Server, der mit der angegebenen Verbindungskennung assoziiert wird. Wenn die Verbindungskennung nicht angegeben wird, wird die zur Zeit offene Verbindung verwendet.

Wenn die letzte Anfrage ein DELETE ohne WHERE-Anweisung war, wurden alle Datensätze aus der Tabelle gelöscht, die Funktion wird aber trotzdem Null zurückliefern.

Diese Funktion macht keinen Sinn bei SELECT-Anfragen, sondern nur wenn Datensätze modifiziert wurden. Um die Anzahl der gelieferten Datensätze einer SELECT-Anfrage zu erhalten, benutzen Sie die Funktion [mysql_num_rows\(\)](#).

mysql_change_user

(PHP3 >= 3.0.13)

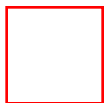
mysql_change_user -- Ändert den zur Zeit angemeldeten Benutzer

Beschreibung

int mysql_change_user (string Benutzername, string Kennwort [, string Datenbank [, int Verbindungs-Kennung]])

Mysql_change_user() ändert den angemeldeten Benutzer der Datenbank-Verbindung in den Benutzer mit dem übergebenen Namen und Kennwort. Fehlt die Verbindungs-Kennung wird die Änderung für die aktive Verbindung vorgenommen. If a database is specified, this will default or current database after the user has been changed. Wenn die Anmeldung als neuer Benutzer mit dem angegebenen Kennwort fehlschlägt, bleibt der aktuelle Benutzer aktiv.

Anmerkung:



Diese Funktion wurde mit PHP 3.0.13 eingeführt und erfordert MySQL 3.23.3 oder höher.

mysql_close

(PHP3 , PHP4)

mysql_close -- Schließt eine Verbindung zum Datenbank-Server

Beschreibung

`int mysql_close ([int Verbindungs-Kennung])`

Rückgabewert: true im Erfolgsfall, false im Fehlerfall.

mysql_close() schließt die Verbindung zur MySQL-Datenbank, die zur Verbindungs-Kennung gehört. Wenn die *Verbindungs-Kennung* nicht angegeben wird, wird die aktuelle Verbindung geschlossen.



Anmerkung: Dies ist für gewöhnlich nicht notwendig, weil nichtpersistente offene Verbindungen automatisch mit Beendigung des PHP-Skripts geschlossen werden.

mysql_close() schließt keine persistenten Verbindungen, die mit [mysql_pconnect\(\)](#) geöffnet wurden.

Beispiel 1 Verbindung zum Datenbankserver schließen

```
<?php
$link = mysql_connect ("kraemer", "marliesle", "secret")
    or die ("Could not connect");
print ("Connected successfully");
mysql_close ($link);
?>
```

Siehe auch: [mysql_connect\(\)](#) und [mysql_pconnect\(\)](#).

mysql_connect

(PHP3 , PHP4)

mysql_connect -- Öffnet eine Verbindung zum Datenbank-Server

Beschreibung

```
int mysql_connect ([string hostname[:port][:path/to/socket] [, string  
Benutzername [, string Kennwort]])
```

Rückgabewert: Eine Verbindungs-Kennung im Erfolgsfall oder false im Fehlerfall.

mysql_connect() öffnet eine Verbindung zu einem MySQL server. Alle Argumente sind optional, und wenn alle fehlen wird ('localhost', Benutzername des Benutzers dem der Server-Prozess gehört, leeres Kennwort) angenommen.

Der Parameter *hostname* kann zudem die Portnummer enthalten, z.B. "hostname:port" oder den Pfad zu einem Socket z.B. ":/path/to/socket" für Zugriffe auf dem lokalen Rechner.

Anmerkung: Unterstützung für ":port" wurde in PHP 3.0B4 hinzugefügt.

Unterstützung für ":/path/to/socket" wurde in PHP 3.0.10 hinzugefügt.

Fehlermeldung können unterdrückt werden, wenn dem Funktionsname ein '@' voran gestellt wird.

Für den Fall, dass ein Aufruf von **mysql_connect()** mit den gleichen Argumenten erfolgt, wird keine neue Verbindung aufgebaut, sondern die Verbindungs-Kennung der schon bestehenden Verbindung zurückgeliefert.

Die Verbindung zum Datenbank-Server wird geschlossen, sobald das PHP-Skript beendet wird oder [mysql_close\(\)](#) aufgerufen wird.

Beispiel 1 Verbindung zum Datenbanksserver öffnen

```
<?php
    $link = mysql_connect ("kraemer", "marliesle", "secret")
        or die ("Could not connect");
    print ("Connected successfully");
    mysql_close ($link);
?>
```

Siehe auch [mysql_pconnect\(\)](#) und [mysql_close\(\)](#).

[Zurück](#)

mysql_close

[Anfang](#)

[Hoch](#)

[Vor](#)

mysql_create_db

mysql_create_db

(PHP3 , PHP4)

mysql_create_db -- Erzeugt eine Datenbank

Beschreibung

int mysql_create_db (string Datenbankname [, int Verbindungs-Kennung])

mysql_create_db() versucht eine neue Datenbank auf dem Server mit der angegebenen Verbindungs-Kennung zu erzeugen.

Beispiel 1 Erzeugen einer Datenbank

```
<?php
$link = mysql_pconnect ("kron", "jutta", "geheim")
    or die ("Could not connect");
if (mysql_create_db ("my_db")) {
    print ("Database created successfully\n");
} else {
    printf ("Error creating database: %s\n", mysql_error ());
}
?>
```

Für Rückwärtskompatibilität kann auch **mysql_createdb()** verwendet werden.

Siehe auch: [mysql_drop_db\(\)](#).

mysql_data_seek

(PHP3 , PHP4)

mysql_data_seek -- Bewegt internen Datensatz-Zeiger

Beschreibung

int mysql_data_seek (int Ergebnis-Kennung, int Datensatznummer)

Rückgabewert: true im Erfolgsfall, false im Fehlerfall.

mysql_data_seek() bewegt den internen Datensatz-Zeiger eines Anfrageergebnisses zum Datensatz mit der übergebenen Nummer. Der nächste Aufruf von [mysql_fetch_row\(\)](#) liefert den entsprechenden Datensatz.

Das Argument *Datensatznummer* ist 0 oder größer.

Beispiel 1 Positionieren des Datensatzzeigers

```
<?php
$link = mysql_pconnect ("kron", "jutta", "geheim")
    or die ("Could not connect");

mysql_select_db ("samp_db")
    or die ("Could not select database");

$query = "SELECT last_name, first_name FROM friends";
$result = mysql_query ($query)
    or die ("Query failed");

# fetch rows in reverse order

for ($i = mysql_num_rows ($result) - 1; $i >=0; $i--) {
    if (!mysql_data_seek ($result, $i)) {
        printf ("Cannot seek to row %d\n", $i);
        continue;
    }

    if(!($row = mysql_fetch_object ($result)))
        continue;
```

```
        printf ("%s %s<BR>\n", $row->last_name, $row->first_name);
    }

    mysql_free_result ($result);
?>
```

[Zurück](#)

mysql_create_db

[Anfang](#)

[Hoch](#)

[Vor](#)

mysql_db_query

mysql_db_query

(PHP3 , PHP4)

mysql_db_query -- Absetzen einer Anfrage an die Datenbank

Beschreibung

int mysql_db_query (string Datenbank, string Anfrage [, int Verbindungs-Kennung])

Rückgabewert: Eine Ergebnis-Kennung als Referenz auf das Anfrageergebnis, oder false im Fehlerfall.

Msql_db_query() wählt eine Datenbank und führt eine Anfrage darauf aus. Wenn der optionale Parameter *Verbindungs-Kennung* nicht angegeben ist, wird versucht die aktuelle Verbindung zu nutzen. Wenn auch dies fehlschlägt, wird weiter versucht die Verbindung aufzubauen, wie dies mit [mysql_connect\(\)](#) ohne Angabe von Argumenten geschehen würde.

Für Rückwärtskompatibilität kann auch **mysql()** verwendet werden.

Siehe auch [mysql_connect\(\)](#).

mysql_drop_db

(PHP3 , PHP4)

mysql_drop_db -- Entfernt eine Datenbank

Beschreibung

`int mysql_drop_db (string Datenbankname [, int Verbindungs-Kennung])`

Rückgabewert: true im Erfolgsfall, false ein Fehlerfall.

mysql_drop_db() versucht eine komplette Datenbank vom Server mit der übergebenen Verbindungs-Kennung zu entfernen.

Siehe auch: [mysql_create_db\(\)](#). Für Rückwärtskompatibilität kann auch **mysql_dropdb()** verwendet werden.

mysql_errno

(PHP3 , PHP4)

mysql_errno -- Liefert die Fehlernummer einer zuvor ausgeführten Operation

Beschreibung

int mysql_errno ([int Verbindungs-Kennung])

Fehler, die vom MySQL Server kommen, führen nicht zu einer Ausgabe von Warnungen. Stattdessen, sollte diese Funktion verwendet werden, um die Fehlernummer zu erhalten.

```
<?php
mysql_connect("marliesle");
echo mysql_errno().": ".mysql_error()."<BR>";
mysql_select_db("nonexistentdb");
echo mysql_errno().": ".mysql_error()."<BR>";
$conn = mysql_query("SELECT * FROM nonexistenttable");
echo mysql_errno().": ".mysql_error()."<BR>";
?>
```

Siehe auch: [mysql_error\(\)](#)

mysql_error

(PHP3 , PHP4)

mysql_error -- Liefert den Fehlertext der zuvor ausgeführten MySQL Operation

Beschreibung

string **mysql_error** ([int Verbindungs-Kennung])

Fehler, die vom MySQL Server kommen, führen nicht zu einer Ausgabe von Warnungen. Stattdessen, sollte diese Funktion verwendet werden, um den Fehlertext zu erhalten.

```
<?php
mysql_connect("marliesle");
echo mysql_errno().": ".mysql_error()."<BR>";
mysql_select_db("nonexistentdb");
echo mysql_errno().": ".mysql_error()."<BR>";
$conn = mysql_query("SELECT * FROM nonexistenttable");
echo mysql_errno().": ".mysql_error()."<BR>";
?>
```

Siehe auch: [mysql_errno\(\)](#)

mysql_fetch_array

(PHP3 , PHP4)

mysql_fetch_array -- Liefert einen Datensatz als assoziatives Array

Beschreibung

array **mysql_fetch_array** (int Ergebnis-Kennung [, int Ergebnistyp])

Liefert ein Array der den aktuellen Datensatz enthält oder false, wenn keine weiteren Datensätze vorliegen.

mysql_fetch_array() ist eine erweiterte Version von [mysql_fetch_row\(\)](#). Der Datensatz wird in einem assoziativen Array abgelegt, in dem die Schlüssel den Namen der Datensatzfelder entsprechen.

Wenn zwei oder mehrere Felder des Ergebnisses den gleichen Feldnamen haben, dann wird der Wert des letzten Feldes im Array unter dem Feldnamen abgelegt. Um auch auf die anderen Felder mit dem gleichen Namen zugreifen zu können, müssen Sie den numerischen Index des Arrays verwenden oder einen Alias für das Feld erzeugen.

```
select t1.f1 as foo t2.f1 as bar from t1, t2
```

Zu betonen ist, dass der Gebrauch von **mysql_fetch_array()** NICHT signifikant langsamer ist als [mysql_fetch_row\(\)](#), obwohl die Funktion einen sichtlichen Mehrwert bedeutet.

Das optionale zweite Argumente *Ergebnistyp* in **mysql_fetch_array()** ist eine Konstante und kann die folgenden Werte annehmen: MYSQL_ASSOC, MYSQL_NUM und MYSQL_BOTH. (Diese Eigenschaft wurde in PHP 3.0.7 hinzugefügt.)

Für weitere Details, siehe auch [mysql_fetch_row\(\)](#).

Beispiel 1 Mysql_fetch_array()

```
<?php
mysql_connect ($host, $user, $password);
$result = mysql_db_query ("database","select * from table");
while ($row = mysql_fetch_array ($result)) {
    echo $row["user_id"];
    echo $row["fullname"];
}
mysql_free_result ($result);
?>
```

[Zurück](#)[mysql_error](#)[Anfang](#)[Hoch](#)[Vor](#)[mysql_fetch_field](#)

mysql_fetch_field

(PHP3 , PHP4)

`mysql_fetch_field` -- Liefert ein Objekt mit Feldinformationen aus einem Anfrageergebnis

Beschreibung

object **mysql_fetch_field** (int Ergebnis-Kennung [, int Feld-Offset])

Liefert ein Objekt mit Feldinformationen.

mysql_fetch_field() kann gebraucht werden, um Feldinformationen über Felder in einem Anfrageergebnis zu erhalten. Wenn der Feld-Offset nicht angegeben wird, werden die Information des nächsten noch nicht mit **mysql_fetch_field()** erfragten Feldes geliefert.

Des Eigenschaften des Objekts sind:

- name - Feldname
- table - Name der Tabelle zu der das Feld gehört
- max_length - maximale Länge des Feldes
- not_null - 1, wenn das Feld nicht Null sein kann
- primary_key - 1, wenn das Feld ein 'primary key' ist
- unique_key - 1, wenn das Feld ein 'unique key' ist
- multiple_key - 1, wenn das Feld ein 'non-unique key' ist
- numeric - 1, wenn das Feld vom Typ 'numeric' ist
- blob - 1, wenn das Feld vom Typ 'BLOB' ist
- type - der Typ des Feldes
- unsigned - 1, wenn das Feld vorzeichenlos ist
- zerofill - 1, wenn das Feld 'zero-filled' ist

Beispiel 1 Mysql_fetch_field()

```

<?php
mysql_connect ($host,$user,$password)
    or die ("Could not connect");
$result = mysql_db_query ("database", "select * from table")
    or die ("Query failed");
# 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);
?>

```

Siehe auch [mysql_field_seek\(\)](#).

[Zurück](#)

mysql_fetch_array

[Anfang](#)

[Hoch](#)

[Vor](#)

mysql_fetch_lengths

mysql_fetch_lengths

(PHP3 , PHP4)

mysql_fetch_lengths -- Liefert die Länge eines jeden Feldes in einem Datensatz

Beschreibung

array **mysql_fetch_lengths** (int Ergebnis-Kennung)

Rückgabewert: Ein Array, das die Längen eines jeden Feldes in dem zuletzt geholten Datensatz enthält, oder false im Fehlerfall.

Mysql_fetch_lengths() liefert die Längen eines jeden Feldes in dem mit [mysql_fetch_row\(\)](#), [mysql_fetch_array\(\)](#) oder [mysql_fetch_object\(\)](#) zuletzt geholten Datensatzes, in einem Array.

Siehe auch: [mysql_fetch_row\(\)](#).

mysql_fetch_object

(PHP3 , PHP4)

mysql_fetch_object -- Liefert einen Datensatz als Objekt

Beschreibung

object **mysql_fetch_object** (int Ergebnis-Kennung [, int Ergebnistyp])

Liefert ein Objekt mit Eigenschaften, die den Felder des geholten Datensatz entsprechen, oder false when keine weiteren Datensätze vorhanden sind.

Mysql_fetch_object() ist ähnlich zu [mysql_fetch_array\(\)](#), mit einem Unterschied - ein Objekt wird zurück geliefert, anstatt eines Arrays. Indirekt bedeutet dies, das Sie die Felder nur mit ihren Feldnamen und nicht mit dem Offset ansprechen können. (Nummer sind ungültige Namen für Eigenschaften).

Das optionale zweite Argumente *Ergebnistyp* in [mysql_fetch_array\(\)](#) ist eine Konstante und kann die folgenden Werte annehmen: MYSQL_ASSOC, MYSQL_NUM und MYSQL_BOTH.

Unter Berücksichtigung der Ausführungsgeschwindigkeit ist diese Funktion identisch zu [mysql_fetch_array\(\)](#) und fast so schnell wie [mysql_fetch_row\(\)](#) (der Unterschied ist unwesentlich).

Beispiel 1 mysql fetch object

```
<?php
mysql_connect ($host, $user, $password);
$result = mysql_db_query ("database", "select * from table");
while ($row = mysql_fetch_object ($result)) {
    echo $row->user_id;
    echo $row->fullname;
}
mysql_free_result ($result);
?>
```

Siehe auch: [mysql_fetch_array\(\)](#) und [mysql_fetch_row\(\)](#).

mysql_fetch_row

(PHP3 , PHP4)

mysql_fetch_row -- Liefert einen Datensatz als indiziertes Array

Beschreibung

array **mysql_fetch_row** (int Ergebnis-Kennung)

Rückgabewert: Ein Array mit dem geholten Datensatz oder false wenn keine weiteren Datensätze verfügbar sind.

Mysql_fetch_row() liefert einen Datensatz aus dem Anfrageergebnis mit der übergebenen Kennung. Der Datensatz wird als Array geliefert. Jeder Feldwert ist als ein Element abgelegt.

Folgende Aufrufe von **mysql_fetch_row()** liefern die weiteren Datensätze des Anfrageergebnisses oder false, wenn keine weiteren Datensätze verfügbar sind.

Siehe auch: [mysql_fetch_array\(\)](#), [mysql_fetch_object\(\)](#), [mysql_data_seek\(\)](#), [mysql_fetch_lengths\(\)](#) und [mysql_result\(\)](#).

mysql_field_name

(PHP3 , PHP4)

mysql_field_name -- Liefert den Namen eines Feldes in einem Anfrageergebnis

Beschreibung

string **mysql_field_name** (int Ergebnis-Kennung, int Feldindex)

Mysql_field_name() liefert den Namen eines Feldes in einem Anfrageergebnis. Argumente der Funktion ist die Kennung des Anfrageergebnisses und der Feldindex, z.B.

```
mysql_field_name($result, 2);
```

Dies liefert den Namen des zweiten Feldes im Anfrageergebnis.

Für Rückwärtskompatibilität kann **mysql_fieldname()** verwendet werden.

mysql_field_seek

(PHP3 , PHP4)

mysql_field_seek -- Setzt den Feldzeiger auf ein bestimmtes Feld

Beschreibung

int mysql_field_seek (int Ergebnis-Kennung, int Feldoffset)

Setzt den Feldzeiger auf den angegebenen Feldoffset. Wenn mit dem nächste Aufruf von [mysql_fetch_field\(\)](#) keinen Feldoffset übergeben wird, wird das Feld auf den der Zeiger verweist zurückgeliefert.

Siehe auch: [mysql_fetch_field\(\)](#).

mysql_field_table

(PHP3 , PHP4)

mysql_field_table -- Liefert den Namen der Tabelle, die das genannte Feld enthält

Beschreibung

string **mysql_field_table** (int Ergebnis-Kennung, int Feldoffset)

Liefert den Namen der Tabelle, die das genannte Feld enthält. Für Rückwärtskompatibilität kann **mysql_fieldtable()** verwendet werden.

mysql_field_type

(PHP3 , PHP4)

mysql_field_type -- Liefert den Typ eines Feldes in einem Anfrageergebnis

Beschreibung

string **mysql_field_type** (int Ergebnis-Kennung, int Feldoffset)

Mysql_field_type() ist ähnlich zu [mysql_field_name\(\)](#). Die Argumente sind identisch, aber der Feldtyp wird zurückgeliefert. Dieser kann einer von "int", "real", "string", "blob", oder anderen sein, wie in der Dokumentation zu MySQL angegeben.

Beispiel 1 Mysql Field Types

```
<?php
mysql_connect ("localhost:3306");
mysql_select_db ("wisconsin");
$result = mysql_query ("SELECT * FROM onek");
$fields = mysql_num_fields ($result);
$rows    = mysql_num_rows ($result);
$i = 0;
$table = mysql_field_table ($result, $i);
echo "Your '". $table. "' table has ". $fields. " fields and ". $rows. " records <BR>";
echo "The table has the following fields <BR>";
while ($i < $fields) {
    $type = mysql_field_type ($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. "<BR>";
    $i++;
}
mysql_close();
?>
```

Für Rückwärtskompatibilität kann **mysql_fieldtype()** verwendet werden.

mysql_field_flags

(PHP3 , PHP4)

mysql_field_flags -- Liefert die Flags eines Feldes in einem Anfrageergebnis

Beschreibung

string **mysql_field_flags** (int Ergebnis-Kennung, int Feldoffset)

Mysql_field_flags() liefert die Flags des Feldes mit dem übergeben Offset. Die Flags werden als einzelne Worte pro Flag und Leerzeichen getrennt geliefert, so dass sie leicht mit [explode\(\)](#) getrennt werden können.

Die folgenden Flags werden geliefert, wenn Ihre Version von MySQL diese unterstützt: "not_null", "primary_key", "unique_key", "multiple_key", "blob", "unsigned", "zerofill", "binary", "enum", "auto_increment", "timestamp".

Für Rückwärtskompatibilität kann **mysql_fieldflags()** verwendet werden.

mysql_field_len

(PHP3 , PHP4)

mysql_field_len -- Liefert die Länge eines Feldes

Beschreibung

int **mysql_field_len** (int Ergebnis-Kennung, int Feldoffset)

Mysql_field_len() liefert die Länge des Feldes mit dem übergebenen Feldoffset. Für Rückwärtskompatibilität kann **mysql_fieldlen()** verwendet werden.

mysql_free_result

(PHP3 , PHP4)

mysql_free_result -- Gibt den belegten Speicher wieder frei

Beschreibung

`int mysql_free_result (int Ergebnis-Kennung)`

mysql_free_result() entfernt ein Ergebnis und gibt den belegten Speicher frei. Die Funktion muss nur dann aufgerufen werden, wenn der Speicherbedarf zur Laufzeit des PHP-Skripts zu hoch wird. Nach Ablauf des PHP-Skripts wird der Speicher ohnehin freigegeben.

Für Rückwärtskompatibilität kann **mysql_freeresult()** verwendet werden.

mysql_insert_id

(PHP3 , PHP4)

mysql_insert_id -- Liefert die Kennung einer vorherigen INSERT-Operation

Beschreibung

`int mysql_insert_id ([int Verbindungs-Kennung])`

Mysql_insert_id() liefert die Kennung (ID), die bei der letzten INSERT-Operation für eine automatisch inkrementiertes Feld (AUTO_INCREMENTED) vergeben wurde. Wenn die *Verbindungs-Kennung* nicht angegeben wird, wird die aktuelle Verbindung verwendet.

mysql_list_fields

(PHP3 , PHP4)

mysql_list_fields -- Listet die Felder einer Tabelle auf

Beschreibung

int mysql_list_fields (string Datenbankname, string Tabellename [,
int Verbindungs-Kennung])

Mysql_list_fields() fordert Information über eine Tabelle an. Argumente sind der Datenbankname und der Tabellename. the table name. Die zurückgelieferte Ergebnis-Kennung kann von den Funktionen [mysql_field_flags\(\)](#), [mysql_field_len\(\)](#), [mysql_field_name\(\)](#) und [mysql_field_type\(\)](#) verwendet werden.

Die Erbebnis-Kennung ist ein positiver ganzzahliger Wert. Die Funktion liefert -1 im Fehlerfall. Eine mögliche Fehlermeldung wird in der Variablen \$phperrormsg abgelegt, es sei denn die Funktion wurde als @mysql() aufgerufen. In diesem Fall wird die Fehlermeldung ausgegeben.

Für Rückwärtskompatibilität kann **mysql_listfields()** verwendet werden.

mysql_list_dbs

(PHP3 , PHP4)

mysql_list_dbs -- Liefert eine Liste der verfügbaren Datenbanken auf dem Server

Beschreibung

`int mysql_list_dbs ([int Verbindungs-Kennung])`

mysql_list_dbs() liefert die Kennung eines Ergebnisses, dass alle Datenbanken auf dem Datenbankserver enthält. Benutzen Sie die Funktion [mysql_tablename\(\)](#), um das Ergebnis zu durchlaufen.

Für Rückwärtskompatibilität kann **mysql_listdbs()** verwendet werden.

mysql_list_tables

(PHP3 , PHP4)

mysql_list_tables -- Liefert eine Liste der Tabellen in einer Datenbank

Beschreibung

`int mysql_list_tables (string Datenbankname [, int Verbindungs-Kennung])`

Mysql_list_tables() liefert eine Liste der Datenbank mit dem übergebenen Namen, ähnlich wie die Funktion [mysql_db_query\(\)](#) bei Datenbankabfragen. Die Funktion [mysql_tablename\(\)](#) sollte genutzt werden, um die tatsächlichen Namen der Tabellen im Ergebnis zu erfragen.

Für Rückwärtskompatibilität kann **mysql_listtables()** verwendet werden.

mysql_num_fields

(PHP3 , PHP4)

mysql_num_fields -- Liefert die Anzahl der Felder in einem Ergebnis

Beschreibung

`int mysql_num_fields (int Ergebnis-Kennung)`

mysql_num_fields() liefert die Anzahl der Felder in einem Ergebnis.

Siehe auch: [mysql_db_query\(\)](#), [mysql_query\(\)](#), [mysql_fetch_field\(\)](#) und [mysql_num_rows\(\)](#).

Für Rückwärtskompatibilität kann **mysql_numfields()** verwendet werden.

mysql_num_rows

(PHP3 , PHP4)

mysql_num_rows -- Liefert die Anzahl der Datensätze im Ergebnis

Beschreibung

`int mysql_num_rows (int Ergebnis-Kennung)`

mysql_num_rows() liefert die Anzahl der Datensätze im Ergebnis.

Siehe auch: [mysql_db_query\(\)](#), [mysql_query\(\)](#) und [mysql_fetch_row\(\)](#).

Für Rückwärtskompatibilität kann **mysql_numrows()** verwendet werden.

mysql_pconnect

(PHP3 , PHP4)

mysql_pconnect -- Öffnete eine persistente Verbindung zum MySQL Server

Beschreibung

```
int mysql_pconnect ([string hostname[:port][:path/to/socket] [,  
string Benutztername [, string Benutzerkennwort]])
```

Rückgabewert: Eine gültige Verbindung-Kennung im Erfolgsfall oder false im Fehlerfall.

Mysql_pconnect() baut eine Verbindung zum MySQL Server auf. Alle Argumente sind optional, und wenn alle fehlen wird ('localhost', Benutzername des Benutzers dem der Server-Prozess gehört, leeres Kennwort) angenommen.

Der Parameter *hostname* kann zudem die Portnummer enthalten, z.B. "hostname:port" oder den Pfad zu einem Socket z.B. ":/path/to/socket" für Zugriffe auf dem lokalen Rechner.

Anmerkung: Unterstützung für ":port" wurde in PHP 3.0B4 hinzugefügt.

Unterstützung für ":/path/to/socket" wurde in PHP 3.0.10 hinzugefügt.

Mysql_pconnect() verhält sich sehr ähnlich zu [mysql_connect\(\)](#), weist aber zwei wesentliche Unterschiede auf.

Erstens, vor dem Verbindungsaufbau wird zunächst versucht eine (persistente) Verbindung zum gleichen Host und mit dem gleichen Benutzernamen und Kennwort zu finden. Wenn dies der Fall ist, wird die Verbindungskennung dieser Verbindung zurückgeliefert und keine neue Verbindung geöffnet.

Zweitens, die Verbindung zum SQL Server wird nicht mit Beendigung des PHP-Skripts geschlossen. Sie bleibt bestehen für einen möglichen, zukünftigen Verbindungsaufbau. (**Mysql_close()** schließt keine mit **mysql_pconnect()** geöffnete Verbindungen).

Dieser Verbindungstyp wird daher 'persistent' genannt.

mysql_query

(PHP3 , PHP4)

mysql_query -- Sendet eine SQL Anfrage zum Datenbankserver

Beschreibung

int mysql_query (string Anfrage [, int Verbindungs-Kennung])

mysql_query() sendet eine Anfrage zu der zur Zeit aktiven Datenbank-Verbindung oder der Verbindung mit der übergebenen Kennung. Wenn keine offene Verbindung besteht, wird versucht eine Verbindung aufzubauen, als ob [mysql_connect\(\)](#) ohne Argumente aufgerufen wird.

Die Anfrage sollte nicht mit einem Semikolon enden.

mysql_query() liefert TRUE (non-zero) oder FALSE, um anzuzeigen, ob die Anfrage erfolgreich war oder nicht. Wird im Erfolgsfall TRUE zurückgeliefert, dann sagt dies nichts über die Zahl veränderter oder gelieferter Datensätze aus. Es ist durchaus möglich, dass eine Anfrage erfolgreich ist, ohne einen einzigen Datensatz zu verändern oder zu liefern.

Die folgende Anfrage ist syntaktisch falsch, deshalb liefert **mysql_query()** FALSE zurück:

Beispiel 1 Mysql_query()

```
<?php
$result = mysql_query ("SELECT * WHERE 1=1")
    or die ("Invalid query");
?>
```

Die folgende Anfrage ist semantisch falsch, wenn `my_col` ist kein Feld in der Tabelle `my_tbl`. Deshalb schlägt **mysql_query()** fehl und liefert FALSE:

Beispiel 2 Mysql_query()

```
<?php
$result = mysql_query ("SELECT my_col FROM my_tbl")
    or die ("Invalid query");
?>
```

mysql_query() schlägt ebenso fehl und liefert FALSE, wenn Sie keinen Zugriff auf die Tabelle(n)

haben.

Angenommen eine Anfrage ist erfolgreich, dann können Sie mit [mysql_affected_rows\(\)](#) herausfinden wieviele Datensätze beeinflusst wurden (für DELETE, INSERT, REPLACE, oder UPDATE Anweisungen). Bei SELECT Anweisungen, liefert **mysql_query()** eine Ergebnis-Kennung, die man mit [mysql_result\(\)](#) weiter verarbeiten kann. Wenn das Ergebnis nicht mehr gebraucht wird, können die belegten Ressourcen mit [mysql_free_result\(\)](#) freigegeben werden.

Siehe auch: [mysql_affected_rows\(\)](#), [mysql_db_query\(\)](#), [mysql_free_result\(\)](#), [mysql_result\(\)](#), [mysql_select_db\(\)](#) und [mysql_connect\(\)](#).

[Zurück](#)

mysql_pconnect

[Anfang](#)

[Hoch](#)

[Vor](#)

mysql_result

mysql_result

(PHP3 , PHP4)

mysql_result -- Liefert Ergebnis

Beschreibung

int mysql_result (int Ergebnis-Kennung, int Datensatz-Index [, mixed Feld])

Mysql_result() liefert den Inhalt eines Feldes aus einem Anfrageergebnis. Das Argument *Feld* kann der Feldname, der Feldoffset ein Bezeichner in der Form 'Feldname.Tabellenname' sein. Wenn das Feld einen Alias besitzt ('select foo as bar from...') muss der Alias anstatt des Feldnamens verwendet werden.

Wenn auf Anfrageergebnisse mit vielen Datensätzen zugegriffen werden soll, sollten Sie die Funktionen, die auf ganze Datensätze zugreifen, in Betracht ziehen (siehe unten). Diese Funktionen sind sehr viel schneller als **mysql_result()**. Beachten Sie auch, dass die Angabe eines numerischen Offsets für das Feld sehr viel schneller ist als die übrigen möglichen Angaben.

Aufrufe von **mysql_result()** sollten nicht mit Aufruf anderen Funktion, die auch auf das Ergebnis zugreifen, verschachtelt werden.

Empfohlende, leistungsfähigere Alternativen: [mysql_fetch_row\(\)](#), [mysql_fetch_array\(\)](#) und [mysql_fetch_object\(\)](#).

mysql_select_db

(PHP3 , PHP4)

mysql_select_db -- Wählt eine Datenbank aus

Beschreibung

int mysql_select_db (string Datenbankname [, int Verbindungs-Kennung])

Rückgabewert: true im Erfolgsfall, false im Fehlerfall.

mysql_select_db() setzt die aktuelle Datenbank auf dem Server zu dem die Verbindung mit der übergebenen Verbindungs-Kennung besteht. Fehlt die Verbindungs-Kennung wird die aktive Verbindung genommen. Wenn auch diese nicht existiert, wird weiter versucht die Verbindung aufzubauen, wie dies mit [mysql_connect\(\)](#) ohne Angabe von Argumenten geschehen würde.

Jeder weitere Aufruf von [mysql_query\(\)](#) nutzt die aktive Verbindung.

Siehe auch: [mysql_connect\(\)](#), [mysql_pconnect\(\)](#) und [mysql_query\(\)](#).

Für Rückwärtskompatibilität kann **mysql_selectdb()** verwendet werden.

mysql_tablename

(PHP3 , PHP4)

mysql_tablename -- Liefert den Namen einer Tabelle

Beschreibung

string **mysql_tablename** (int Ergebnis-Kennung, int i)

MySQL_tablename() ermittelt aus einem Ergebnis, das von [mysql_list_tables\(\)](#) geliefert wurde sowie einem Index *i* den Namen der Tabelle. Die Funktion [mysql_num_rows\(\)](#) kann verwendet werden, um die Anzahl der Tabellen im Ergebnis zu ermitteln.

Beispiel 1 MySQL_tablename() Beispiel

```
<?php
mysql_connect ("localhost:3306");
$result = mysql_list_tables ("wisconsin");
$i = 0;
while ($i < mysql_num_rows ($result)) {
    $tb_names[$i] = mysql_tablename ($result, $i);
    echo $tb_names[$i] . "<BR>";
    $i++;
}
?>
```

XXXIX. Netzwerk Funktionen

Inhaltsverzeichnis

[checkdnsrr](#) Prüft DNS-Einträge auf Übereinstimmung mit einem gegebenen Internet-Host-Namen oder einer IP Adresse

[closelog](#) Schließt die Verbindung zum System-Logger

[debugger_off](#) Schaltet den internen PHP-Debugger aus

[debugger_on](#) Schaltet den internen PHP-Debugger ein

[fsockopen](#) Stellt eine Internet- oder Unix-Domain-Socket-Verbindung her

[gethostbyaddr](#) Ermittelt den Internet-Host-Namen (z.B. tux.meinsein.de) passend zur angegebenen IP-Adresse (z.B. 192.168.0.2)

[gethostbyname](#) Ermittelt die IP-Adresse (z.B. 192.168.0.2) passend zum angegebenen Internet-Host-Namen (z.B. tux.meinsein.de)

[gethostbyname_l](#) Ermittelt eine Liste von IP-Adressen passend zum angegebenen Internet-Host-Namen

[getmxrr](#) Ermittelt die (DNS) MX-Datensätze passend zu einem angegebenen Internet-Host-Namen.

[getprotobyname](#) Ermittelt die Protokoll-Nummer anhand des Protokoll-Namens

[getprotobynumber](#) Ermittelt den Protokoll-Namen anhand der Protokoll-Nummer

[getservbyname](#) Ermittelt einen Port-Wert passend zu einem Internet-Dienst und Protokoll

[getservbyport](#) Ermittelt einen Internet-Dienst passend zu einem Port und Protokoll

[openlog](#) Stellt eine Verbindung zu einem Log-Dienst des Systems her

[pfsockopen](#) Stellt eine dauernde/permanente Internet oder Unix-Domain-Socket Verbindung her

[socket_set_blocking](#) Schaltet den Block-Modus eines Sockets an oder aus

[syslog](#) Erzeugt eine Meldung im System-Logging

checkdnsrr

(PHP3 , PHP4)

checkdnsrr -- Prüft DNS-Einträge auf Übereinstimmung mit einem gegebenen Internet-Host-Namen oder einer IP Adresse

Beschreibung

int **checkdnsrr** (string host [, string type])

Durchsucht den DNS (Domain-Name-Service) auf Daten des Typs *type* auf Übereinstimmung mit *host*. Gibt true/wahr zurück, wenn ein Datensatz gefunden wurde. Ansonsten - oder wenn ein Fehler vorliegt - wird false/falsch zurück gegeben.

Der Parameter *type* kann sein: A, MX, NS, SOA, PTR, CNAME, or ANY. Die Vorgabe ist MX.

Der Parameter *host* kann eine durch Punkte getrennte IP-Adresse (z.B. 192.168.0.2) oder ein Host-Name (z.B. tux.meinsein.de) sein.

Siehe auch [getmxrr\(\)](#), [gethostbyaddr\(\)](#), [gethostbyname\(\)](#), [gethostbyname\(\)](#) und das named(8) Manual.

closelog

(PHP3 , PHP4)

closelog -- Schließt die Verbindung zum System-Logger

Beschreibung

```
int closelog(void);
```

Closelog() schließt den Descriptor, der benötigt wird um in den System-Logger zu schreiben. Der Gebrauch von **closelog()** ist optional.

debugger_off

(PHP3)

debugger_off -- Schaltet den internen PHP-Debugger aus

Beschreibung

```
int debugger_off(void);
```

Schaltet den internen PHP-Debugger aus. Der Debugger befindet sich noch in der Entwicklung.

[Zurück](#)[Anfang](#)[Vor](#)

closelog

[Hoch](#)

debugger_on

debugger_on

(PHP3)

debugger_on -- Schaltet den internen PHP-Debugger ein

Beschreibung

int **debugger_on** (string address)

Schaltet den internen PHP-Debugger, zeigend auf *address*, ein. Der Debugger befindet sich noch in der Entwicklung.

[Zurück](#)

debugger_off

[Anfang](#)[Hoch](#)[Vor](#)

fsockopen

fsockopen

(PHP3 , PHP4)

fsockopen -- Stellt eine Internet- oder Unix-Domain-Socket-Verbindung her

Beschreibung

```
int fsockopen (string hostname, int port [, int errno [, string errstr  
[, double timeout]])
```

Öffnet eine Socket-Verbindung zum Internet (AF_INET) oder unter Unix (AF_UNIX). Für eine Internet-Verbindung wird eine TCP-Socket-Verbindung zu *hostname* über den Port *port* hergestellt. Bei einer UNIX-Verbindung wird *hostname* als Socket-Pfad interpretiert, *port* muss in diesem Fall 0 sein. Der optionale Parameter *timeout* kann benutzt werden, um eine Fehlerwartezeit (timeout) für den Verbindungsaufbau (in Sekunden) zu setzen.

Die Funktion **fsockopen()** gibt einen Zeiger auf eine Datei zurück, der in Verbindung mit anderen Datei-Funktionen (wie z.B. [fgets\(\)](#), [fgetss\(\)](#), [fputs\(\)](#), [fclose\(\)](#), [feof\(\)](#)) gebraucht werden kann.

Wenn der Aufruf der Funktion scheitert, gibt er false / falsch zurück und wenn die optionalen *errno* und *errstr* Argumente vorhanden sind, enthalten sie den aktuellen System-Fehler, der beim Aufruf der System-Funktion `connect ( )` (Verbindungsaufbau) erscheint bzw. passiert ist. Wenn die erhaltene *errno* 0 ist und die Funktion false/falsch zurück gibt, ist dies ein Hinweis, dass der Fehler vor dem `>connect ( )`-Aufruf geschah. Das ist ein Problem, dass häufig mit der Initialisierung des Sockets zusammen hängt. Beachten sie bitte, dass das *errno* und *errstr*-Argument als Referenz zu übergeben sind.

Abhängig von der vorhandenen Umgebung kann die Verbindung zu einer UNIX-Domäne oder der optionale Verbindungsaufbau-Zeitfehler nicht zur Verfügung stehen.

Voreingestellt ist der Verbindungsaufbau im Block-Modus. Sie können dies auf Nicht-Block-Modus ändern, indem sie **socket_set_blocking()** benutzen.

Beispiel 1 Fsockopen() Beispiel

```
$fp = fsockopen ("www.php.net", 80, &$errno, &$errstr, 30);  
if (!$fp) {  
    echo "$errstr ($errno)<br>\n";  
} else {  
    fputs ($fp, "GET / HTTP/1.0\n\n");  
    while (!feof($fp)) {  
        echo fgets($fp,128);  
    }  
    fclose($fp);  
}
```

Siehe auch: [pfsockopen\(\)](#)

[Zurück](#)[debugger_on](#)[Anfang](#)[Hoch](#)[Vor](#)[gethostbyaddr](#)

gethostbyaddr

(PHP3 , PHP4)

gethostbyaddr -- Ermittelt den Internet-Host-Namen (z.B. tux.meinsein.de) passend zur angegebenen IP-Adresse (z.B. 192.168.0.2)

Beschreibung

string **gethostbyaddr** (string *ip_address*)

Gibt den Internet-Host-Namen passend zu *ip_address* zurück. Bei einem Fehler wird die *ip_address* zurück gegeben.

Siehe auch: [gethostbyname\(\)](#).

gethostbyname

(PHP3 , PHP4)

gethostbyname -- Ermittelt die IP-Adresse (z.B. 192.168.0.2) passend zum angegebenen Internet-Host-Namen (z.B. tux.meinsein.de)

Beschreibung

string **gethostbyname** (string hostname)

Gibt die IP-Adresse passend zu *hostname* zurück.

Siehe auch: [gethostbyaddr\(\)](#).

gethostbyname1

(PHP3 , PHP4)

gethostbyname1 -- Ermittelt eine Liste von IP-Adressen passend zum angegebenen Internet-Host-Namen

Beschreibung

array **gethostbyname1** (string hostname)

Gibt eine Liste von IP-Adressen zurück, zu denen der angegebene Internet-Host *hostname* aufgelöst wurde.

Siehe auch [gethostbyname\(\)](#), [gethostbyaddr\(\)](#), [checkdnsrr\(\)](#), [getmxrr\(\)](#), und die `named(8)` Bedienungs-Anleitung.

getmxrr

(PHP3 , PHP4)

getmxrr -- Ermittelt die (DNS) MX-Datensätze passend zu einem angegebenen Internet-Host-Namen.

Beschreibung

int **getmxrr** (string hostname, array mxhosts [, array weight])

Sucht im DNS nach MX-Datensätzen die zu *hostname* passen. Gibt true/wahr zurück, wenn irgendwelche Daten gefunden wurden, ansonsten - oder wenn ein Fehler auftrat - wird false/falsch zurück gegeben.

Eine Liste der gefundenen MX-Einträge wird im Array *mxhosts* zurück gegeben. Wenn das optionale *weight*-Array vorhanden ist, wird es mit den gefundenen Prioritäts-Informationen gefüllt.

See also [checkdnsrr\(\)](#), [gethostbyname\(\)](#), [gethostbynameel\(\)](#), [gethostbyaddr\(\)](#), und die `named(8)` Bedienungs-Anleitung.

getprotobyname

(PHP4 >= 4.0b4)

getprotobyname -- Ermittelt die Protokoll-Nummer anhand des Protokoll-Namens

Beschreibung

int getprotobyname (string name)

Die Funktion **getprotobyname()** gibt die Protokoll-Nummer passend zu dem Protokoll *name* (z.B. TCP, IP, UDP, RAW) anhand der Datei `/etc/protocols` zurück.

Siehe auch [getprotobyname\(\)](#).

getprotobynumber

(PHP4 >= 4.0b4)

getprotobynumber -- Ermittelt den Protokoll-Namen anhand der Protokoll-Nummer

Beschreibung

string **getprotobynumber** (int number)

Die Funktion **getprotobynumber()** gibt den Protokoll-Namen passend zur angegebenen Protokoll-*number* anhand der Datei `/etc/protocols` zurück

Siehe auch: [getprotobyname\(\)](#).

getservbyname

(PHP4 >= 4.0b4)

getservbyname -- Ermittelt einen Port-Wert passend zu einem Internet-Dienst und Protokoll

Beschreibung

int **getservbyname** (string service, string protocol)

Die Funktion **getservbyname()** gibt den Internet-Port zurück, der zu *service* passt. Dies geschieht anhand des angegebenen Wertes für *protocol* mittels der Datei */etc/services*. Der Inhalt von *protocol* ist entweder TCP oder UDP.

Siehe auch: [getservbyport\(\)](#).

getservbyport

(PHP4 >= 4.0b4)

getservbyport -- Ermittelt einen Internet-Dienst passend zu einem Port und Protokoll

Beschreibung

string **getservbyport** (int port, string protocol)

Die Funktion **getservbyport()** gibt den Internet-Dienst zurück, der zu *port* für das angegebene *protocol* passt (anhand der Datei */etc/services*). Der Inhalt von *protocol* ist entweder TCP oder UDP.

Siehe auch: [getservbyname\(\)](#).

openlog

(PHP3 , PHP4)

openlog -- Stellt eine Verbindung zu einem Log-Dienst des Systems her

Beschreibung

int **openlog** (string *ident*, int *option*, int *facility*)

Die Funktion **openlog()** kontaktiert die System-Log-Dienste eines Programms. Die Zeichenkette *ident* wird jeder Meldung angefügt. Werte für die Parameter *option* und *facility* werden im nächsten Abschnitt gegeben. Der Gebrauch von **openlog()** ist optional. Er wird - falls erforderlich - automatisch aufgerufen von [syslog\(\)](#). Dann ist der Vorgabewert von *ident* falsch/false.

Siehe auch [syslog\(\)](#) und [closelog\(\)](#).

pfsockopen

(PHP3 >= 3.0.7, PHP4)

pfsockopen -- Stellt eine dauernde/permanente Internet oder Unix-Domain-Socket Verbindung her

Beschreibung

```
int pfsockopen (string hostname, int port [, int errno [, string  
errstr [, int timeout]])
```

Diese Funktion beinhaltet das gleiche wie [fsockopen\(\)](#) mit dem Unterschied, dass die Verbindung nicht beendet wird, sobald das Script beendet wird. Sie ist die permanente-Version von [fsockopen\(\)](#).

[Zurück](#)[Anfang](#)[Vor](#)

openlog

[Hoch](#)

socket_set_blocking

socket_set_blocking

(PHP4 >= 4.0b4)

socket_set_blocking -- Schaltet den Block-Modus eines Sockets an oder aus

Beschreibung

int socket_set_blocking (int socket descriptor, int mode)

Wenn *mode* false / falsch ist, wird die angegebene Socket- Bezeichnung auf Nicht-Block-Modus gesetzt, anderenfalls auf Block-Modus. Dies beeinflusst Aufrufe von [fgets\(\)](#), die über diesen Sockets lesen. Im Nicht-Block-Modus wird ein [fgets\(\)](#)-Aufruf sofort zum Script zurückkehren, während er im Block-Modus auf eingehende Daten (auf diesem Socket) warten wird.

syslog

(PHP3 , PHP4)

syslog -- Erzeugt eine Meldung im System-Logging

Beschreibung

int **syslog** (int priority, string message)

Die Funktion **syslog()** erzeugt eine Status-Meldung, die an die System-Log-Funktion weitergereicht wird. Der Parameter *priority* ist eine Kombination der System-Möglichkeiten und dem Level. Werte hierzu werden im nächsten Abschnitt genannt. Es verbleibt die zu sendende Nachricht, in der die beiden Zeichen %m durch den Fehlertext (strerror) ersetzt werden, der dem Wert errno entspricht.

Mehr Informationen über die System-Möglichkeiten können in der Bedienungs-Anleitung (man pages auf UNIX-Systemen) zu syslog gefunden werden.

Unter Windows-NT wird der Syslog-Dienst emuliert, indem der Ereignis-Protokollierdienst genutzt wird.

XL. ODBC Funktionen

Inhaltsverzeichnis

- [odbc_autocommit](#) Ändert das Autocommit-Verhalten
- [odbc_binmode](#) Die Behandlung von Binärdaten
- [odbc_close](#) Beendet eine ODBC-Verbindung
- [odbc_close_all](#) Beendet alle ODBC-Verbindungen
- [odbc_commit](#) Führt eine ODBC-Transaktion aus
- [odbc_connect](#) Baut die Verbindung zu einer ODBC-Datenquelle auf
- [odbc_cursor](#) Findet den Cursornamen heraus
- [odbc_do](#) Ein Synonym für [odbc_exec\(\)](#)
- [odbc_exec](#) Bereitet einen SQL-Befehl auf und führt ihn aus
- [odbc_execute](#) Führt ein vorbereiteten SQL-Befehl aus
- [odbc_fetch_into](#) Eine Ergebniszeile in ein Array stellen
- [odbc_fetch_row](#) Liefert eine Datenzeile zurück
- [odbc_field_name](#) Liefert die Spaltenbezeichnung
- [odbc_field_type](#) Liefert den Datentyp eines Feldes
- [odbc_field_len](#) Bestimmt die Länge eines Feldes
- [odbc_free_result](#) Gibt den durch ein Abfrageergebnis belegten Speicher wieder frei
- [odbc_longreadlen](#) Steuert die Nutzung von LONG-Spalten
- [odbc_num_fields](#) Liefert die Anzahl der Ergebnisspalten
- [odbc_pconnect](#) Öffnet eine persistente Datenbankverbindung
- [odbc_prepare](#) Stellt einen SQL-Befehl zur Ausführung bereit
- [odbc_num_rows](#) Ergibt die Zeilenzahl des Abfrageergebnisses
- [odbc_result](#) Erlaubt den Zugriff auf die Ergebnisdaten
- [odbc_result_all](#) Gibt das aktuelle Abfrageergebnis als HTML-Tabelle aus
- [odbc_rollback](#) Hebt eine Transaktion wieder auf
- [odbc_setoption](#) Verändert die ODBC-Einstellungen

syslog

[Hoch](#)

odbc_autocommit

odbc_autocommit

(PHP3 >= 3.0.6, PHP4)

odbc_autocommit -- Ändert das Autocommit-Verhalten

Beschreibung

int odbc_autocommit (int *connection_id* [, int *OnOff*])

Ohne den Parameter *OnOff* gibt **odbc_autocommit()** den aktuellen autocommit-Status für die Datenbankverbindung *connection_id* zurück. Wenn autocommit aktiviert ist, wird **true** zurückgeliefert, False wenn es deaktiviert ist oder ein Fehler auftritt.

In der Standardeinstellung ist autocommit für eine Datenbankverbindung aktiviert. Das Deaktivieren von autocommit entspricht dem Durchführen einer Transaktion.

Siehe auch [odbc_commit\(\)](#) und [odbc_rollback\(\)](#).

odbc_binmode

(PHP3 >= 3.0.6, PHP4)

odbc_binmode -- Die Behandlung von Binärdaten

Beschreibung

`int odbc_binmode (int result_id, int mode)`

(Betrifft die ODBC SQL Typen: BINARY, VARBINARY und LONGVARBINARY)

- ODBC_BINMODE_PASSTHRU: gibt Binärdaten direkt zum Output durch
- ODBC_BINMODE_RETURN: liefert die Binärdaten unverändert zurück
- ODBC_BINMODE_CONVERT: konvertiert die Binärdaten zu Zeichen und liefere sie zurück

Im ODBC_BINMODE_CONVERT werden die Binärdaten zu Character-C-data gewandelt. Jedes Byte (8 Bits) der ursprünglichen Daten wird zu zwei ASCII-Zeichen, die der hexadezimalen Darstellung des Wertes entsprechen. Der Binärwert 00000001 wird zum Beispiel zu "01" gewandelt, der Binärwert 11111111 zu "FF".

Tabelle 1 Die Benutzung von LONGVARBINARY

binmode	longreadlen	Resultat
ODBC_BINMODE_PASSTHRU	0	Direkte Ausgabe
ODBC_BINMODE_RETURN	0	Direkte Ausgabe
ODBC_BINMODE_CONVERT	0	Direkte Ausgabe
ODBC_BINMODE_PASSTHRU	0	Direkte Ausgabe
ODBC_BINMODE_PASSTHRU	>0	Direkte Ausgabe
ODBC_BINMODE_RETURN	>0	Direkte Ausgabe
ODBC_BINMODE_CONVERT	>0	Rückgabe als CHAR-Werte

Wenn [odbc_fetch_into\(\)](#) genutzt wird, wird bei direkter Ausgabe ein leerer String als Spaltenwert zurückgeliefert.

Wenn *result_id* gleich 0 ist, wird die Einstellung als Defaultwert für die nächsten Transaktionen gesetzt.

Anmerkung: Der Defahltwert für longreadlen ist 4096, für `odbc_binmode()`

ODBC_BINMODE_RETURN, also die unveränderte Rückgabe. Das Verhalten von LONGVARBINARY wird ebenfalls durch die Funktion [odbc_longreadlen\(\)](#) bestimmt.

[Zurück](#)

odbc_autocommit

[Anfang](#)[Hoch](#)[Vor](#)

odbc_close

odbc_close

(PHP3 >= 3.0.6, PHP4)

odbc_close -- Beendet eine ODBC-Verbindung

Beschreibung

void **odbc_close** (int connection_id)

odbc_close() beendet die Verbindung zum Datenbankserver, die durch das Verbindungskennzeichen `connection_id` gekennzeichnet wird.

Anmerkung: Diese Funktion scheitert, wenn auf dieser Datenbankverbindung noch Transaktionen ausstehen. In diesem Fall bleibt die Verbindung bestehen.

odbc_close_all

(PHP3 >= 3.0.6, PHP4)

odbc_close_all -- Beendet alle ODBC-Verbindungen

Beschreibung

```
void odbc_close_all(void);
```

Odbc_close_all() beendet alle bestehenden Verbindungen zu allen Datenbankservern.

Anmerkung: Wie [odbc_close\(\)](#) scheitert auch dieser Befehl, wenn auf einer Datenbankverbindung noch Transaktionen ausstehen oder aktiv sind. In diesem Fall bleibt die betroffene Verbindung auch nach dem **odbc_close_all()** bestehen.

odbc_commit

(PHP3 >= 3.0.6, PHP4)

odbc_commit -- Führt eine ODBC-Transaktion aus

Beschreibung

int odbc_commit (int connection_id)

Gibt bei Erfolg **true** zurück, bei Mißerfolg **false**. Alle ausstehenden Transaktionen auf der Datenbankverbindung *connection_id* werden ausgeführt.

[Zurück](#)[Anfang](#)[Vor](#)

odbc_close_all

[Hoch](#)

odbc_connect

odbc_connect

(PHP3 >= 3.0.6, PHP4)

odbc_connect -- Baut die Verbindung zu einer ODBC-Datenquelle auf

Beschreibung

int odbc_connect (string dsn, string user, string password [, int cursor_type])

Liefert eine ODBC-Verbindungskennung `connection_id` oder 0 (`false`) bei Fehlfunktion zurück.

Die Verbindungskennung `connection_id` wird von den anderen ODBC-Funktionen benötigt. Es können mehrere ODBC-Verbindungen nebeneinander bestehen. Der optionale vierte Parameter `cursor_type` setzt die Art des Cursors, der für diese Verbindung genutzt werden soll. Normalerweise wird er nicht benötigt, kann aber bei manchen ODBC-Treiberproblemen nützlich sein.

Es können etwa Probleme bei der Ausführung von komplexen Prozeduren auftreten. Die Fehlermeldung lautet dann zum Beispiel "Cannot open a cursor on a stored procedure that has anything other than a single select statement in it". Dieser Fehler kann durch die Verwendung von `SQL_CUR_USE_ODBC` als `cursor_type` kann hier helfen. Einige ODBC-Treiber unterstützen den optionalen Parameter `row_number` in der Funktion [odbc_fetch_row\(\)](#) nicht. Auch in diesem Fall kann `SQL_CUR_USE_ODBC` helfen.

Folgende Konstanten sind als Cursortyp definiert:

- `SQL_CUR_USE_IF_NEEDED`
- `SQL_CUR_USE_ODBC`
- `SQL_CUR_USE_DRIVER`
- `SQL_CUR_DEFAULT`

Für persistente Verbindungen schauen Sie sich bitte [odbc_pconnect\(\)](#) an.

odbc_cursor

(PHP3 >= 3.0.6, PHP4)

odbc_cursor -- Findet den Cursornamen heraus

Beschreibung

string **odbc_cursor** (int result_id)

Odbc_cursor() liefert einen Cursornamen für die gegebene Ergebniskennung *result_id* zurück.

odbc_do

(PHP3 >= 3.0.6, PHP4)

odbc_do -- Ein Synonym für [odbc_exec\(\)](#)

Beschreibung

`int odbc_do (int conn_id, string query)`

Odbc_do() führt eine Anfrage auf der gegebenen Verbindung durch.

[Zurück](#)[Anfang](#)[Vor](#)

odbc_cursor

[Hoch](#)

odbc_exec

odbc_exec

(PHP3 >= 3.0.6, PHP4)

odbc_exec -- Bereitet einen SQL-Befehl auf und führt ihn aus

Beschreibung

int odbc_exec (int connection_id, string query_string)

Liefert bei einem Fehler `false` zurück, sonst eine ODBC-Ergebniskennung `result_id`.

odbc_exec() sendet einen SQL-Befehl zu dem Datenbankserver, der durch die Verbindungskennung `connection_id` bezeichnet wird. Dieser Parameter muß durch [odbc_connect\(\)](#) oder [odbc_pconnect\(\)](#) erzeugt worden sein.

Siehe auch: [odbc_prepare\(\)](#) und [odbc_execute\(\)](#) für die gleichzeitige Ausführung von mehreren SQL-Befehlen.

odbc_execute

(PHP3 >= 3.0.6, PHP4)

odbc_execute -- Führt ein vorbereiteten SQL-Befehl aus

Beschreibung

int odbc_execute (int result_id [, array parameters_array])

Führt ein oder mehrere durch [odbc_prepare\(\)](#) vorbereitete SQL-Befehle aus. Nach erfolgreicher Ausführung wird `true` zurückgeliefert, sonst `false`. Das Array *parameters_array* braucht nur angegeben zu werden, wenn im Befehl tatsächlich Parameter vorhanden sind.

odbc_fetch_into

(PHP3 >= 3.0.6, PHP4)

odbc_fetch_into -- Eine Ergebniszeile in ein Array stellen

Beschreibung

```
int odbc_fetch_into (int result_id [, int rownumber, array  
result_array])
```

Liefert die Anzahl der Ergebnisspalten zurück, bei einem Fehler `false`. Das Zielarray *result_array* muß immer übergeben werden, kann aber einen beliebigen Typ haben - es wird ohnehin zu einem Array gemacht. Das Array enthält die Ergebnisspalten beginnend mit 0.

[Zurück](#)[Anfang](#)[Vor](#)

odbc_execute

[Hoch](#)

odbc_fetch_row

odbc_fetch_row

(PHP3 >= 3.0.6, PHP4)

odbc_fetch_row -- Liefert eine Datenzeile zurück

Beschreibung

int odbc_fetch_row (int result_id [, int row_number])

Wenn **odbc_fetch_row()** erfolgreich durchgeführt wurde (also eine oder mehrere Datenzeilen gefunden wurden), dann wird **true** zurückgeliefert. Wenn keine (weitere) Zeile vorhanden ist, wird **false** zurückgegeben.

Odbc_fetch_row() stellt eine Zeile der von [odbc_do\(\)/odbc_exec\(\)](#) zurückgegeben Daten bereit. Nachdem **odbc_fetch_row()** aufgerufen wurde, können die Felder dieser Zeile mit [odbc_result\(\)](#) genutzt werden.

Wenn *row_number* nicht definiert ist, versucht **odbc_fetch_row()** die nächste Zeile aus dem Abfrageergebnis *result_id* zu holen. **odbc_fetch_row()** kann abwechselnd ohne und mit dem Parameter *row_number* genutzt werden.

Um durch ein Abfrageergebnis mehrfach zu durchlaufen, kann man **odbc_fetch_row()** mit *row_number* gleich 1 aufrufen, um dann wie gewohnt mit **odbc_fetch_row()** ohne Zeilenparameter fortzufahren. Wenn der ODBC-Treiber die Herausgabe von bestimmten Zeilen nicht unterstützt, wird der Parameter ignoriert.

odbc_field_name

(PHP3 >= 3.0.6, PHP4)

odbc_field_name -- Liefert die Spaltenbezeichnung

Beschreibung

string **odbc_field_name** (int result_id, int field_number)

Odbc_field_name() gibt den Namen der Spalte mit der Nummer `field_number` für des Abfrageergebnisses `result_id` zurück. Die Nummerierung der Spalten beginnt mit 1. Bei einem Fehler wird `false` zurückgegeben.

odbc_field_type

(PHP3 >= 3.0.6, PHP4)

odbc_field_type -- Liefert den Datentyp eines Feldes

Beschreibung

string **odbc_field_type** (int result_id, int field_number)

Odbc_field_type() liefert den SQL-Datentyp eines Feldes zurück, das in dem Abfrageergebnis *result_id* an der Stelle *field_number* zu finden ist.

odbc_field_len

(PHP3 >= 3.0.6, PHP4)

odbc_field_len -- Bestimmt die Länge eines Feldes

Beschreibung

int **odbc_field_len** (int result_id, int field_number)

Odbc_field_len() gibt die Länge eines Feldes an. Das Feld wird durch die Feldnummer *field_number* im Abfrageergebnis *result_id* bestimmt. Die Feldnummerierung beginnt mit 1.

[Zurück](#)[Anfang](#)[Vor](#)[odbc_field_type](#)[Hoch](#)[odbc_free_result](#)

odbc_free_result

(PHP3 >= 3.0.6, PHP4)

`odbc_free_result` -- Gibt den durch ein Abfrageergebnis belegten Speicher wieder frei

Beschreibung

```
int odbc_free_result (int result_id)
```

Gibt immer `true` zurück.

`odbc_free_result()` braucht nur aufgerufen zu werden, wenn Sie einen zu hohen Speicherverbrauch durch das aktuell laufende Skript vermeiden wollen. Das gesamte Abfrageergebnis wird automatisch nach Beendigung des Skriptes gelöscht. Aber wenn Sie sicher sind, dass Sie dieses Ergebnis nicht länger benötigen, dann können Sie bei Speicherproblemen die Funktion **`odbc_free_result()`** benutzen.

Anmerkung: Wenn `autocommit` deaktiviert ist (siehe [`odbc\_autocommit\(\)`](#)) und Sie **`odbc_free_result()`** bevor ein Commit gemacht wird, werden alle austehenden Transaktionen zurückgenommen.

odbc_longreadlen

(PHP3 >= 3.0.6, PHP4)

odbc_longreadlen -- Steuert die Nutzung von LONG-Spalten

Beschreibung

`int odbc_longreadlen (int result_id, int length)`

(Betrifft folgende ODBC-SQL-Typen: LONG, LONGVARBINARY) Die Anzahl der an PHP zurückgelieferten Bytes wird von durch den Parameter *length* bestimmt. Wenn er auf 0 gesetzt ist, wird der Inhalt von LONG-Spalten direkt an den Klienten weitergegeben.

Anmerkung: Die Behandlung von LONGVARBINARY-Spalten wird auch von [odbc_binmode\(\)](#) beeinflusst.

odbc_num_fields

(PHP3 >= 3.0.6, PHP4)

odbc_num_fields -- Liefert die Anzahl der Ergebnisspalten

Beschreibung

```
int odbc_num_fields (int result_id)
```

Odbc_num_fields() liefert die Anzahl der Spalten, die in einem ODBC-Ergebnis vorhanden sind, zurück. Bei einem Fehler ergibt **odbc_num_fields()** -1.

odbc_pconnect

(PHP3 >= 3.0.6, PHP4)

odbc_pconnect -- Öffnet eine persistente Datenbankverbindung

Beschreibung

int odbc_pconnect (string dsn, string user, string password [, int cursor_type])

Gibt eine ODBC-Verbindungskennung zurück, bei einem Fehler 0 (`false`). Diese Funktion ist [odbc_connect\(\)](#) sehr ähnlich. Der einzige Unterschied besteht darin, dass die Verbindung nach dem Ende des Skriptes erhalten bleibt. Folgende Verbindungen mit gleicher Datenquelle, Nutzer und Passwort-Kombination über [odbc_connect\(\)](#) oder **odbc_pconnect()** können dann die bestehende Verbindung weiterbenutzen.

Anmerkung: Persistente Verbindungen können bei einem CGI-PHP nicht benutzt werden. In diesem Fall entspricht **odbc_pconnect()** [odbc_connect\(\)](#).

Für weitere Informationen über den optionalen Parameter *cursor_type* schauen Sie bitte bei [odbc_connect\(\)](#). Weitere Informationen über persistente Datenbankverbindungen finden Sie in der PHP-FAQ

odbc_prepare

(PHP3 >= 3.0.6, PHP4)

odbc_prepare -- Stellt einen SQL-Befehl zur Ausführung bereit

Beschreibung

int odbc_prepare (int connection_id, string query_string)

Liefert bei einem Fehler `false` zurück.

Gibt eine ODBC-Ergebniskennung zurück, wenn der SQL-Befehl erfolgreich bereitgestellt wurde. Die Ergebniskennung kann später zur Ausführung des Befehles mit [odbc_execute\(\)](#) genutzt werden.

odbc_num_rows

(PHP3 >= 3.0.6, PHP4)

odbc_num_rows -- Ergibt die Zeilenzahl des Abfrageergebnisses

Beschreibung

```
int odbc_num_rows (int result_id)
```

Odbc_num_rows() liefert die Anzahl Zeilen eines ODBC-Abfrageergebnisses. Tritt ein Fehler auf, ist sie -1. Bei INSERT, UPDATE und DELETE-Befehlen wird die Zahl der betroffenen Zeilen zurückgegeben. Bei einer SELECT-Abfrage kann dies die Anzahl der vorhandenen Zeilen sein.

Anmerkung: Bei vielen Treibern wird -1 als Anzahl möglicher Zeilen nach einem SELECT-Befehl zurückgeliefert.

odbc_result

(PHP3 >= 3.0.6, PHP4)

odbc_result -- Erlaubt den Zugriff auf die Ergebnisdaten

Beschreibung

string **odbc_result** (int result_id, mixed field)

Liefert den Inhalt eines bestimmten Feldes zurück.

Der Feldparameter *field* kann entweder der Integerwert der Spaltennummer oder ein String mit dem Spaltennamen sein. Als Beispiel:

```
$produkt_3 = odbc_result ($Query_ID, 3 );  
$produkt_wert = odbc_result ($Query_ID, "wert");
```

Der erste Aufruf von **odbc_result()** liefert den Wert des drittes Feldes im aktuellen Datensatz im Abfrageergebnis zurück. Der zweite Funktionsaufruf gibt den Wert der Spalte "wert" im aktuellen Datensatz zurück. Wenn die Spaltennummer kleiner als 1 oder größer als die Spaltenzahl ist, tritt ein Fehler auf. Gleiches gilt für einen Spaltennamen, der im Abfrageergebnis nicht vorkommt.

Die Spaltennummern beginnen mit 1. Für die Nutzung von Binärdaten und LONG-Spalten schauen Sie bitte bei [odbc_binmode\(\)](#) und [odbc_longreadlen\(\)](#) nach.

odbc_result_all

(PHP3 >= 3.0.6, PHP4)

`odbc_result_all` -- Gibt das aktuelle Abfrageergebnis als HTML-Tabelle aus

Beschreibung

`int odbc_result_all (int result_id [, string format])`

Liefert die Anzahl Reihen im Abfrageergebnis `result_id` zurück, wenn ein Fehler auftritt `false`.

`Odbc_result_all()` gibt alle Reihen eines SQL-Abfrageergebnisses `result_id` aus einem [`odbc\_exec\(\)`](#)-Aufruf als HTML-Tabelle aus. Mit dem optionalen String *format* kann ein allgemeines Tabellenformat angegeben werden.

odbc_rollback

(PHP3 >= 3.0.6, PHP4)

odbc_rollback -- Hebt eine Transaktion wieder auf

Beschreibung

int odbc_rollback (int connection_id)

Hebt alle ausstehenden Transaktionen auf der Verbindung *connection_id* wieder auf. Bei Erfolg wird **true** zurückgegeben, sonst **false**.

odbc_setoption

(PHP3 >= 3.0.6, PHP4)

odbc_setoption -- Verändert die ODBC-Einstellungen

Beschreibung

int odbc_setoption (int id, int function, int option, int param)

Diese Funktion erlaubt das Einstellen der ODBC-Optionen für eine einzelne Verbindung. Sie soll beim Auffinden und Beheben von Problemen mit ODBC-Treibern helfen. Es empfiehlt sich, diese Funktionen nur zu nutzen, wenn man einige Erfahrung mit ODBC-Programmierung gesammelt hat und die Wirkungen der einzelnen Parameter kennt. Um mit Ihnen zu arbeiten, benötigen Sie eine Referenz der aktuellen Treiberversion des eingesetzten ODBC-Treibers, denn die Optionen sind von Treiber zu Treiber und von Version zu Version unterschiedlich.

Weil die Effekte je nach ODBC-Treiber unterschiedlich sind, sollte Skripte mit diesen Funktionen nicht veröffentlicht werden. Einige Optionen stehen nicht zur Verfügung, weil sie vor dem Aufbau einer ODBC-Verbindung eingestellt sein müssen. Wie auch immer, in einigen Fällen hilft diese Funktion zu vermeiden, dass man von seinem Vorgesetzten gezwungen wird, ein kommerzielles Programm einzusetzen - und darauf kommt es doch an ;-).

Id ist die Verbindungs oder Ergebniskennung, deren Optionen geändert werden sollen. Bei SQLSetConnectOption() ist es eine Verbindungskennung, bei SQLSetStmtOption() eine Ergebniskennung.

function ist die ODBC-Funktion, die genutzt werden soll. Die Werte sind 1 für SQLSetConnectOption() und 2 für SQLSetStntOption().

Der Parameter *option* ist die Option, die gesetzt werden soll.

Der Parameter *param* ist der Wert für die gegebene *option*.

Beispiel 1 odbc.setoption Beispiele

```
// 1. Option 102 von SQLSetConnectOption() ist SQL_AUTOCOMMIT.  
// Wert 1 für SQL_AUTOCOMMIT entspricht SQL_AUTOCOMMIT_ON.  
// Dieses Beispiel hat denselben Effekt wie  
// odbc_autocommit($conn, true);  
  
odbc_setoption ($conn, 1, 102, 1);  
  
// 2. Option 0 von SQLSetStmtOption() ist SQL_QUERY_TIMEOUT.  
// Dieses Beispiel setzt den Timeout auf 30 Sekunden.  
  
$result = odbc_prepare ($conn, $sql);  
odbc_setoption ($result, 2, 0, 30);  
odbc_execute ($result);
```

[Zurück](#)[odbc_rollback](#)[Anfang](#)[Hoch](#)[Vor](#)[Oracle functions](#)

XLI. Oracle functions

Inhaltsverzeichnis

- [Ora_Bind](#) bind a PHP variable to an Oracle parameter
- [Ora_Close](#) close an Oracle cursor
- [Ora_ColumnName](#) get name of Oracle result column
- [Ora_ColumnType](#) get type of Oracle result column
- [Ora_Commit](#) commit an Oracle transaction
- [Ora_CommitOff](#) disable automatic commit
- [Ora_CommitOn](#) enable automatic commit
- [Ora_Error](#) get Oracle error message
- [Ora_ErrorCode](#) get Oracle error code
- [Ora_Exec](#) execute parsed statement on an Oracle cursor
- [Ora_Fetch](#) fetch a row of data from a cursor
- [Ora_GetColumn](#) get data from a fetched row
- [Ora_Logoff](#) close an Oracle connection
- [Ora_Logon](#) open an Oracle connection
- [Ora_Open](#) open an Oracle cursor
- [Ora_Parse](#) parse an SQL statement
- [Ora_Rollback](#) roll back transaction

Ora_Bind

(unknown)

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

(unknown)

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

(unknown)

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.

Ora_ColumnType

(unknown)

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*. The returned type will be one of the following:

"VARCHAR2"

"VARCHAR"

"CHAR"

"NUMBER"

"LONG"

"LONG RAW"

"ROWID"

"DATE"

"CURSOR"

Ora_Commit

(unknown)

Ora_Commit -- commit an Oracle transaction

Description

```
int ora_commit (int conn)
```

Returns true on success, false on error. 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.

[Zurück](#)[Anfang](#)[Vor](#)

Ora_ColumnType

[Hoch](#)

Ora_CommitOff

Ora_CommitOff

(unknown)

Ora_CommitOff -- disable automatic commit

Description

```
int ora_commitoff (int conn)
```

Returns true on success, false on error. 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

(unknown)

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, false on error. Details about the error can be retrieved using the [ora_error\(\)](#) and [ora_errorcode\(\)](#) functions.

Ora_Error

(unknown)

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.

Anmerkung: 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

(unknown)

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.

Anmerkung: Support for connection ids was added in 3.0.4.

Ora_Exec

(unknown)

Ora_Exec -- execute parsed statement on an Oracle cursor

Description

```
int ora_exec (int cursor)
```

Returns true on success, false on error. Details about the error can be retrieved using the [ora_error\(\)](#) and [ora_errorcode\(\)](#) functions.

Ora_Fetch

(unknown)

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 occurred). If an error occurred, 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.

Ora_GetColumn

(unknown)

Ora_GetColumn -- get data from a fetched row

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

(unknown)

Ora_Logoff -- close an Oracle connection

Description

int **ora_logoff** (int connection)

Returns true on success, False on error. Details about the error can be retrieved using the [ora_error\(\)](#) and [ora_errorcode\(\)](#) functions. Logs out the user and disconnects from the server.

Ora_Logon

(unknown)

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_Open

(unknown)

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

(unknown)

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 0 on success or -1 on error.

Ora_Rollback

(unknown)

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, false on error. Details about the error can be retrieved using the [ora_error\(\)](#) and [ora_errorcode\(\)](#) functions.

XLII. Oracle 8 functions

Diese Funktionen gestatten den Zugriff auf Oracle 7 und Oracle8 Datenbanken. Es wird hierbei das Oracle8 Call-Interface (OCI8) verwendet, welches installierte Oracle8 Client Bibliotheken (Libraries) benötigt.

Diese Schnittstelle ist flexibler als die Standard Oracle Funktionen. Es unterstützt Binding von lokalen und globalen PHP Variablen an Oracle Platzhalter und hat volle LOB,FILE und ROWID Unterstützung und gestattet dem Benutzer die Verwendung von Variablendefines.

Inhaltsverzeichnis

[OCIDefineByName](#) Use a PHP variable for the define-step during a SELECT

[OCIBindByName](#) Bind a PHP variable to an Oracle Placeholder

[OCILogon](#) Establishes a connection to Oracle

[OCIPLogon](#) Connect to an Oracle database and log on using a persistant connection. Returns a new session.

[OCINLogon](#) Connect to an Oracle database and log on using a new connection. Returns a new session.

[OCILogOff](#) Disconnects from Oracle

[OCIExecute](#) Execute a statement

[OCICommit](#) Commits outstanding transactions

[OCIRollback](#) Rolls back outstanding transactions

[OCINewDescriptor](#) Initialize a new empty descriptor LOB/FILE (LOB is default)

[OCIRowCount](#) Gets the number of affected rows

[OCINumCols](#) Return the number of result columns in a statement

[OCIResult](#) Returns coulumn value for fetched row

[OCIFetch](#) Fetches the next row into result-buffer

[OCIFetchInto](#) Fetches the next row into result-array

[OCIFetchStatement](#) Fetch all rows of result data into an array.

[OCIColumnIsNULL](#) test whether a result column is NULL

[OCIColumnSize](#) return result column size

[OCIServerVersion](#) Return a string containing server version information.

[OCIStatementType](#) Return the type of an OCI statement.

[OCINewCursor](#) return a new cursor (Statement-Handle) - use this to bind ref-cursors!

[OCIFreeStatement](#) Gibt alle mit dem Statement verbundenen Ressourcen frei.

[OCIFreeCursor](#) Gibt alle mit dem cursor verbundenen Ressourcen frei.

[OCIColumnName](#) Liefert den Namen der Spalte zurück.

[OCIColumnType](#) Liefert den Datentyp einer Spalte zurück.

[OCIParse](#) Analysiert eine Abfrage und gibt eine Anweisung zurück.

[OCIError](#) Gibt den letzten Fehler von stmt|conn|global zurück oder false, wenn kein Fehler aufgetreten ist.

[OCIInternalDebug](#) Aktiviert oder deaktiviert die interne Debugausgabe. Die Voreinstellung ist deaktiviert.

[Zurück](#)

Ora_Rollback

[Anfang](#)

[Hoch](#)

[Vor](#)

OCIDefineByName

OCIDefineByName

(unknown)

OCIDefineByName -- Use a PHP variable for the define-step during a SELECT

Description

int OCIDefineByName (int stmt, string Column-Name, mixed &variable [, int type])

OCIDefineByName() uses fetches SQL-Columns into user-defined PHP-Variables. Be careful that Oracle user ALL-UPPERCASE column-names, whereby in your select you can also write lower-case. **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.

Beispiel 1 OCIDefineByName

```
<?php
/* OCIDefineByPos example thies@digicol.de (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);  
?>
```

[Zurück](#)

[Anfang](#)

[Vor](#)

Oracle 8 functions

[Hoch](#)

OCIBindByName

OCIBindByName

(unknown)

OCIBindByName -- Bind a PHP variable to an Oracle Placeholder

Description

`int OCIBindByName (int stmt, string ph_name, mixed &variable, intlength [, 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* paramter 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).

Beispiel 1 OCIDefineByName

```
<?php
/* OCIBindByPos example thies@digicol.de (980221)

   inserts 3 resords 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);
?>
```

[Zurück](#)

OCIDefineByName

[Anfang](#)[Hoch](#)[Vor](#)

OCILogon

OCILogon

(unknown)

OCILogon -- Establishes a connection to Oracle

Description

int **OCILogon** (string username, string password [, string db])

OCILogon() returns a 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.

Beispiel 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()**.

[Zurück](#)

OCIBindByName

[Anfang](#)

[Hoch](#)

[Vor](#)

OCIPLogon

OCIPLogon

(unknown)

OCIPLogon -- Connect to an Oracle database and log on using a persistent connection. Returns a new session.

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()**.

OCINLogon

(unknown)

OCINLogon -- Connect to an Oracle database and log on using a new connection. Returns a new session.

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.

Beispiel 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()**.

[Zurück](#)

OCIPLogon

[Anfang](#)

[Hoch](#)

[Vor](#)

OCILogOff

OCILogOff

(unknown)

OCILogOff -- Disconnects from Oracle

Description

int **OCILogOff** (int connection)

OCILogOff() closes an Oracle connection.

[Zurück](#)

OCILogon

[Anfang](#)[Hoch](#)[Vor](#)

OCIExecute

OCIExecute

(unknown)

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 automaticly specify OCI_DEFAULT as your mode.

[Zurück](#)[Anfang](#)[Vor](#)

OCILogOff

[Hoch](#)

OCICommit

OCICommit

(unknown)

OCICommit -- Commits outstanding transactions

Description

```
int OCICommit (int connection)
```

OCICommit() commits all outstanding statements for Oracle connection *connection*.

[Zurück](#)[Anfang](#)[Vor](#)

OCIExecute

[Hoch](#)

OCIRollback

OCIRollback

(unknown)

OCIRollback -- Rolls back outstanding transactions

Description

int **OCIRollback** (int connection)

OCIRollback() rolls back all outstanding statements for Oracle connection *connection*.

[Zurück](#)[Anfang](#)[Vor](#)

OCICommit

[Hoch](#)

OCINewDescriptor

OCINewDescriptor

(unknown)

OCINewDescriptor -- Initialize a new empty descriptor LOB/FILE (LOB is default)

Description

string **OCINewDescriptor** (int connection [, int type])

OCINewDescriptor() Allocates storage to hold descriptors or LOB locators. Valid values for the valid *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.

Beispiel 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.php3" method="post" enctype="multipart/form-data">
 * <input type="file" name="lob_upload">
 * ...
 */
if(!isset($lob_upload) || $lob_upload == 'none'){
?>
<form action="upload.php3" 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
    $conn = OCILogon($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);
    if($lob->savefile($lob_upload)){
        OCICommit($conn);
        echo "Blob successfully uploaded\n";
    }else{
        echo "Couldn't upload Blob\n";
    }
    OCIFreeDescriptor($lob);
    OCIFreeStatement($stmt);
    OCILogoff($conn);
}
?>

```

[Zurück](#)

OCIRollback

[Anfang](#)[Hoch](#)[Vor](#)

OCIRowCount

OCIRowCount

(unknown)

OCIRowCount -- Gets the number of affected rows

Description

int **OCIRowCount** (int statement)

OCIRowCounts() returns the number of rows affected for eg update-statements. This funtions will not tell you the number of rows that a select will return!

Beispiel 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>";
?>
```

OCINumCols

(unknown)

OCINumCols -- Return the number of result columns in a statement

Description

int **OCINumCols** (int stmt)

OCINumCols() returns the number of columns in a statement

Beispiel 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";
?>
```

OCIResult

(unknown)

OCIResult -- Returns coulumn 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).

OCIFetch

(unknown)

OCIFetch -- Fetches the next row into result-buffer

Description

int **OCIFetch** (int statement)

OCIFetch() fetches the next row (for SELECT statements) into the internal result-buffer.

OCIFetchInto

(unknown)

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 one-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 addig 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 one. (DEFAULT)

OCI_RETURN_NULLS Return empty columns.

OCI_RETURN_LOBS Return the value of a LOB instead of the desxriptor.

OCIFetchStatement

(unknown)

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.

Beispiel 1 OCIFetchStatement

```
<?php
/* OCIFetchStatement example mbritton@verinet.com (990624) */

$conn = OCILogin("scott","tiger");

$stmt = OCIParse($conn,"select * from emp");

OCIExecute($stmt);

$rows = OCIFetchStatement($stmt,$results);
if ( $rows > 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 < $rows; $i++ ) {
        reset($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);  
  
?>
```

[Zurück](#)

OCIFetchInto

[Anfang](#)[Hoch](#)[Vor](#)

OCIColumnIsNULL

OCIColumnIsNULL

(unknown)

OCIColumnIsNULL -- test whether a result column is NULL

Description

int **OCIColumnIsNULL** (int stmt, mixed column)

OCIColumnIsNULL() returns true if the returned column *col* 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.

[Zurück](#)

OCIFetchStatement

[Anfang](#)[Hoch](#)[Vor](#)

OCIColumnSize

OCIColumnSize

(unknown)

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.

Beispiel 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>";
$numcols = OCINumCols($stmt);
for ( $i = 1; $i <= $numcols; $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()**.

[Zurück](#)

OCIColumnIsNULL

[Anfang](#)[Hoch](#)[Vor](#)

OCIServerVersion

OCIServerErrorVersion

(unknown)

OCIServerErrorVersion -- Return a string containing server version information.

Description

string **OCIServerErrorVersion** (int conn)

Beispiel 1 OCIServerErrorVersion

```
<?php
    $conn = OCILogon("scott","tiger");
    print "Server Version: " . OCIServerErrorVersion($conn);
    OCILogOff($conn);
?>
```

OCIStatementType

(unknown)

OCIStatementType -- Return the type of an OCI statement.

Description

string **OCIStatementType** (int stmt)

OCIStatementType() returns on of the following values:

1. "SELECT"
2. "UPDATE"
3. "DELETE"
4. "INSERT"
5. "CREATE"
6. "DROP"
7. "ALTER"
8. "BEGIN"
9. "DECLARE"
10. "UNKNOWN"

Beispiel 1 Code 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>";
?>
```

[Zurück](#)[Anfang](#)
[Hoch](#)[Vor](#)

OCIServerVersion

OCINewCursor

OCINewCursor

(unknown)

OCINewCursor -- return a new cursor (Statement-Handle) - use this to bind ref-cursors!

Description

int **OCINewCursor** (int conn)

OCINewCursor() allocates a new statement handle on the specified connection.

Beispiel 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);
}

OCIFreeCursor($stmt);
OCIFreeStatement($curs);
OCILogoff($conn);
?>
```

Beispiel 2 Using a REF CURSOR in a select statement

```

<?php
print "<HTML><BODY>";
$conn = OCILogon("scott","tiger");
$count_cursor = "CURSOR(select count(empno) num_ems 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_ems = $subdata["NUM_EMPS"];
        print "<TD>$num_ems</TD>";
    }
    print "</TR>";
}
print "</TABLE>";
print "</BODY></HTML>";
OCIFreeStatement($stmt);
OCILogoff($conn);
?>

```

[Zurück](#)[Anfang](#)[Vor](#)

OCIStatementType

[Hoch](#)

OCIFreeStatement

OCIFreeStatement

(unknown)

OCIFreeStatement -- Gibt alle mit dem Statement verbundenen Ressourcen frei.

Beschreibung

```
int OCIFreeStatement (int stmt)
```

OCIFreeStatement() gibt true zurück, wenn es erfolgreich war, false, wenn nicht.

OCIFreeCursor

(unknown)

OCIFreeCursor -- Gibt alle mit dem cursor verbundenen Ressourcen frei.

Beschreibung

```
int OCIFreeCursor (int stmt)
```

OCIFreeCursor() gibt true zurück, wenn die Aktion erfolgreich war, false, wenn nicht.

OCIColumnName

(unknown)

OCIColumnName -- Liefert den Namen der Spalte zurück.

Beschreibung

string **OCIColumnName** (int stmt, int col)

OCIColumnName() liefert den Name einer Spalte zurück passend zu der Spaltennummer (beginnend bei 1).

Beispiel 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>";
$numcols = OCINumCols($stmt);
for ( $i = 1; $i <= $numcols; $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>";
}
OCIFreeStatement($stmt);
OCILogoff($conn);
print "</PRE>";
print "</HTML>\n";
```

[?>](#)

Bitte beachten Sie auch **OCINumCols()**, **OCIColumnType()**, und **OCIColumnSize()**.

[Zurück](#)

OCIFreeCursor

[Anfang](#)[Hoch](#)[Vor](#)

OCIColumnType

OCIColumnType

(unknown)

OCIColumnType -- Liefert den Datentyp einer Spalte zurück.

Beschreibung

mixed **OCIColumnName** (int stmt, int col)

OCIColumnType() liefert den Datentyp einer Spalte zurück passend zu der Spaltennummer (beginnend bei 1).

Beispiel 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>";
$numcols = OCINumCols($stmt);
for ( $i = 1; $i <= $numcols; $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>";
}
OCIFreeStatement($stmt);
OCILogoff($conn);
print "</PRE>";
print "</HTML>\n";
```

[?>](#)

Bitte beachten Sie auch **OCINumCols()**, **OCIColumnName()**, und **OCIColumnSize()**.

[Zurück](#)

OCIColumnName

[Anfang](#)[Hoch](#)[Vor](#)

OCIParse

OCIParse

(unknown)

OCIParse -- Analysiert eine Abfrage und gibt eine Anweisung zurück.

Beschreibung

int **OCIParse** (int conn, strint query)

OCIParse() analysiert die *query* und benutzt dabei die *conn*. Es liefert true zurück, wenn die Abfrage gültig ist, false, wenn nicht. *query* kann jede Form einer gültigen SQL Anfrage sein.

OCIError

(unknown)

OCIError -- Gibt den letzten Fehler von `stmt|conn|global` zurück oder `false`, wenn kein Fehler aufgetreten ist.

Beschreibung

mixed **OCIError** ([int `stmt|conn`])

OCIError() liefert den letzten Fehler. Ist der optionale Parameter *stmt|conn* nicht angegeben, wird der zuletzt aufgetretene Fehler zurückgeliefert. Ist kein Fehler gefunden, gibt **OCIError()** `false` zurück.

OCIError() gibt den Fehler als assoziatives Array zurück, wobei *code* den Oracle Fehlercode und *message* den Fehlerstring beinhaltet.

OCIInternalDebug

(unknown)

OCIInternalDebug -- Aktiviert oder deaktiviert die interne Debugausgabe. Die Voreinstellung ist deaktiviert.

Beschreibung

void **OCIInternalDebug** (int onoff)

OCIInternalDebug() aktiviert die interne Debugausgabe. *onoff* kann auf 0 gesetzt werden, um die Ausgabe abzuschalten und auf 1, um sie anzuschalten.

XLIII. PDF Funktionen

Mit Hilfe der PDF-Bibliothek von Thomas Merz können mit PHP PDF Dateien erzeugt werden. Die Bibliothek ist unter <http://www.pdflib.com/pdflib/index.html>; verfügbar. Zudem sind noch zwei weitere Bibliotheken notwendig, [die JPEG Bibliothek](#) und [die TIFF Bibliothek](#), um PHP zu übersetzen. Diese beiden Bibliothek bereiten häufig Probleme bei der Konfiguration von PHP. Sie sollten unbedingt die Meldungen des Konfigurations-Skripts befolgen, um aufkommende Probleme zu lösen.

Mit Version 2.20 von pdflib wurden einige Veränderungen an der API vorgenommen. Zudem ist Unterstützung für asiatische Zeichensätze hinzugekommen. Dies hat leider auch zu Änderungen am php4-Modul geführt (nicht php3). Wenn Sie pdflib 2.20 benutzen, dann sollten Sie die Erzeugung von Dokumenten im Speicher mit Vorsicht benutzen. Bis zur entgültigen Version 3.0 von pdflib könnte dies instabil sein. Der encoding Parameter der Funktion [pdf_set_font\(\)](#) hat sich in eine Zeichenkette geändert. Dies bedeutet, dass anstatt von beispielsweise 4 jetzt 'winansi' verwendet werden muss.

Wenn Sie die Version 2.30 von pdflib verwenden, dann steht Ihnen die Funktion [pdf_set_text_matrix\(\)](#) nicht mehr zur Verfügung. Diese Funktion wurde komplett entfernt. Grundsätzlich ist es ratsam die release notes der verwendeten Version von pdflib zu lesen.

Versionen von PHP4 die nach dem 9. März 2000 erscheinen unterstützen nur noch die Versionen >3.0 von pdflib. PHP3 sollte hingegen nicht mit Versionen >2.01 verwendet werden.

Bei Gebrauch von pdflib 2.01 sollte überprüft werden, ob die Bibliothek richtig installiert wurde. Es sollte die Datei oder ein Verweis libpdf.so im Installationsverzeichnis von pdflib existieren. Version 2.01 erzeugt nur die Bibliothek mit dem Namen libpdf2.01.so, die so nicht vom Linker des Systems gefunden werden kann. In diesem Fall müssen Sie den Verweis von libpdf.so nach libpdf2.01.so selbst anlegen.

Beachten sie zudem die exzellente Dokumentation die mit pdflib ausgeliefert wird. Sie gibt einen guten Überblick über die Möglichkeiten von pdflib.

Die meisten Funktionen in pdflib sind in ähnlicher Form auch in PHP vorhanden. Die Parameter sind in der Regel auch identisch. Sie sollten zudem die Grundkonzepte von PDF und Postscript verstanden haben, um das PDF Module effizient nutzen zu können.

Alle Längen und Koordinatenangaben sind in Postscript-Punkten gemessen. Für gewöhnlich entsprechen 72 PostScript-Punkte 1 Inch, was jedoch von der Auflösung des Ausgabegeräts abhängt.

Neben diesem Modul gibt es noch ein weiteres zur Erzeugung von PDF-Dateien basierend auf der Bibliothek ClibPDF von [FastIO](#). Es hat eine geringfügig andere Programmierschnittstelle. Schauen Sie in die [Funktionsübersicht](#) für weitere Details.

Zur Zeit werden alle Versionen von pdflib unterstützt. Es wird empfohlen, dass die neuste Version Verwendung findet, weil nur dann alle Möglichkeiten genutzt werden können und Probleme mit älteren Versionen umgangen werden. Leider hat der Umstieg von 0.6 auf 2.x so weitreichende Änderungen hervorgerufen, dass einige der PHP Funktionen geändert werden mussten. Hier ist ein Liste der davon betroffen Funktionen:

- Die Info-Struktur existiert nicht mehr. Deshalb ist auch die Funktion [pdf_get_info\(\)](#) nicht mehr notwendig

und die Funktionen **pdf_set_info_creator()**, **pdf_set_info_title()**, **pdf_set_info_author()**, **pdf_set_info_subject()** und **pdf_set_info_keywords()** benötigen nicht mehr die Info-Struktur als ersten Parameter sondern das PDF-Dokument. Dies bedeutet zudem, dass das PDF-Dokument vor dem Aufruf dieser Funktion geöffnet werden muss. Die obigen Funktionen können und sollten zudem durch die Funktion **pdf_set_info()** ersetzt werden.

- Die Art und Weise wie ein neues PDF-Dokument geöffnet wird hat sich verändert. Die Funktion **pdf_open()** benötigt nur noch einen Parameter, den Deskriptor der mit **fopen()** geöffneten Datei.

Mit der Version 2.01 von pdflib kamen weitere Veränderungen hinzu, die weitgehend von PHP abgefangen wurden. Dennoch werden einige Funktionen nicht mehr benötigt (z. B. **pdf_put_image()**). Diese Funktionen liefern bei Aufruf eine harmlose Warnung.

Das PDF Module von PHP verwendet zwei neue Typen von Variablen (bei Verwendung von pdflib 2.x ist es nur einer). Sie werden *pdfdoc* und *pdfinfo* (*pdfinfo* existiert in pdflib 2.x nicht) genannt. *pdfdoc* ist ein Zeiger auf das PDF Dokument und wird bei fast allen Funktion als erster Parameter erwartet. *pdfinfo* enthält Meta-Daten über das PDF Dokument. Sie muss vor dem Aufruf von **pdf_open()** gesetzt werden.

Anmerkung: Der folgende Absatz ist nur gültig bei Verwendung von pdflib 0.6. Lesen sie die Dokumentation von pdflib bei Verwendung neuerer Versionen.

Zur Textausgabe in ein PDF-Dokument müssen die Zeichensatz-Metriken (afm Dateien) für jeden Zeichensatz bereitgestellt werden. Afm-Dateien enthalten die Geometrie eines jeden Buchstaben in einem Postscript-Zeichensatz. In der Grundeinstellung werden diese Dateien in dem Verzeichnis 'fonts', relativ zum Verzeichnis in dem das PHP-Skript ist, gesucht. Wie gesagt, dies galt für pdflib 0.6, neuere Versionen brauchen für gewöhnlich diese Dateien nicht.

Die meisten Funktionen sind sehr einfach zu benutzen. Das Schwierigste wird wohl sein, überhaupt ein einfaches PDF-Dokument zu erstellen. Das folgende Beispiel soll die ersten Schritte erleichtern. Es benutzt die PHP-Funktionen basierend auf pdflib 0.6. Dieses Skript erstellt die PDF-Datei test.pdf, welche nur aus einer Seite besteht. Auf der Seite befindet sich der Text "Times-Roman" in einem outlined 30pt Zeichensatz. Der Text ist zudem unterstrichen.

Beispiel 1 Erstellung eines PDF Dokuments mit pdflib 0.6

```
<?php
$fp = fopen("test.pdf", "w");
$info = PDF_get_info();
pdf_set_info_author($info, "Uwe Steinmann");
PDF_set_info_title($info, "Test for PHP wrapper of PDFlib 0.6");
PDF_set_info_creator($info, "Name of Author");
pdf_set_info_creator($info, "See Author");
pdf_set_info_subject($info, "Testing");
$pdf = PDF_open($fp, $info);
PDF_begin_page($pdf, 595, 842);
PDF_add_outline($pdf, "Page 1");
pdf_set_font($pdf, "Times-Roman", 30, 4);
pdf_set_text_rendering($pdf, 1);
PDF_show_xy($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);
fclose($fp);
echo "<A HREF=getpdf.php3>finished</A>";
?>
```

Das PHP-Skript getpdf.php3 liefert nur das PDF-Document.

```
<?php
$fp = fopen("test.pdf", "r");
header("Content-type: application/pdf");
fpassthru($fp);
fclose($fp);
?>
```

Das gleiche Beispiel mit pdflib 2.x sieht wie folgt aus:

Beispiel 2 Erstellung eines PDF-Document mit pdflib 2.x

```
<?php
$fp = fopen("test.pdf", "w");
$pdf = PDF_open($fp);
pdf_set_info_author($pdf, "Uwe Steinmann");
PDF_set_info_title($pdf, "Test for PHP wrapper of PDFlib 2.0");
PDF_set_info_author($pdf, "Name of Author");
pdf_set_info_creator($pdf, "See Author");
pdf_set_info_subject($pdf, "Testing");
PDF_begin_page($pdf, 595, 842);
PDF_add_outline($pdf, "Page 1");
pdf_set_font($pdf, "Times-Roman", 30, 4);
pdf_set_text_rendering($pdf, 1);
PDF_show_xy($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($pdf);
fclose($fp);
echo "<A HREF=getpdf.php3>finished</A>";
?>
```

Das PHP-Skript getpdf.php3 ist zu dem oben beschriebenen identisch.

Die pdflib-Distribution enthält ein komplizierteres Beispiel, welches ein Reihe Seiten erstellt die jeweils eine analoge Uhr mit der aktuellen Uhrzeit enthalten. Dieses Beispiel in ein PHP-Skript umgesetzt sieht wie folgt aus (das gleiche Beispiel wird auch in der Dokumentation zum [cpdf Modul](#)) verwendet):

Beispiel 3 pdfclock Beispiel aus der pdflib 2.x Distribution

```
<?php
$pdffilename = "clock.pdf";
$radius = 200;
$margin = 20;
$pagecount = 40;

$fp = fopen($pdffilename, "w");
$pdf = pdf_open($fp);
pdf_set_info_creator($pdf, "pdf_clock.php3");
pdf_set_info_author($pdf, "Uwe Steinmann");
pdf_set_info_title($pdf, "Analog Clock");

while($pagecount-- > 0) {
    pdf_begin_page($pdf, 2 * ($radius + $margin), 2 * ($radius + $margin));

    pdf_set_transition($pdf, 4); /* wipe */
    pdf_set_duration($pdf, 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);
}

$pdf = pdf_close($pdf);
fclose($fp);
echo "<A HREF=getpdf.php3?filename=".$pdffilename.">finished</A>";
?>

```

Das PHP-Skript getpdf.php3 liefert nur das Dokument.

```
<?php
$fp = fopen($filename, "r");
header("Content-type: application/pdf");
fpassthru($fp);
fclose($fp);
?>
```

Inhaltsverzeichnis

- [PDF_set_info](#) Setzt Info-Feld der Dokumenteninformation
- [PDF_open](#) Öffnet ein neues PDF-Dokument
- [PDF_close](#) Schließt ein PDF-Dokument
- [PDF_begin_page](#) Beginnt eine neue Seite
- [PDF_end_page](#) Beendet eine Seite
- [PDF_show](#) Schreibt einen Text an die aktuelle Position
- [PDF_show_boxed](#) Schreibt einen Text in eine Box
- [PDF_show_xy](#) Schreibt einen Text an die angegebene Position
- [PDF_set_font](#) Wählt einen Zeichensatz und dessen Größe aus
- [PDF_set_leading](#) Setzt den Abstand zwischen zwei Textzeilen
- [PDF_set_parameter](#) Setzt verschiedene Parameter
- [PDF_get_parameter](#) Liefert verschiedene Parameter
- [PDF_set_value](#) Setzt verschiedene numerische Werte
- [PDF_get_value](#) Liefert verschiedene numerische Werte
- [PDF_set_text_rendering](#) Bestimmt wie der Text ausgegeben werden soll
- [PDF_set_horiz_scaling](#) Setzt die horizontale Skalierung bei der Textausgabe
- [PDF_set_text_rise](#) Setzt die Textverschiebung
- [PDF_set_text_matrix](#) Setzt die Text-Matrix
- [PDF_set_text_pos](#) Setzt die Textposition
- [PDF_set_char_spacing](#) Setzt den Abstand zwischen Zeichen
- [PDF_set_word_spacing](#) Setzt den Abstand zwischen Wörtern
- [PDF_skew](#) Schert das Koordinatensystem
- [PDF_continue_text](#) Schreibt den Text in die nächste Zeile
- [PDF_stringwidth](#) Liefert die benötigte Breite einer Zeichenkette mit dem aktuelle Zeichensatz
- [PDF_save](#) Sichert die aktuelle Umgebung
- [PDF_restore](#) Stellt eine zuvor gesicherte Umgebung wieder her
- [PDF_translate](#) Setzt den Ursprung des Koordinatensystems
- [PDF_scale](#) Setzt den Skalierungsfaktor
- [PDF_rotate](#) Setzt die Rotation

[PDF_setflat](#) Sets flatness

[PDF_setlinejoin](#) Setzt die Verbindungsart von Linien

[PDF_setlinecap](#) Setzt den Type der Linienenden

[PDF_setmiterlimit](#) Sets miter limit

[PDF_setlinewidth](#) Setzt die Linienbreite

[PDF_setdash](#) Setzt das Muster für gestrichelte Linien

[PDF_moveto](#) Setzt die aktuelle Position

[PDF_curveto](#) Zeichnet eine Kurve

[PDF_lineto](#) Zeichnet eine Linie

[PDF_circle](#) Zeichnet einen Kreis

[PDF_arc](#) Zeichnet einen Kreisbogen

[PDF_rect](#) Zeichnet ein Rechteck

[PDF_closepath](#) Schließt einen Pfad

[PDF_stroke](#) Zeichnet eine Linie entlang eines Pfades

[PDF_closepath_stroke](#) Schließt einen Pfad und zeichnet eine Linie entlang des Pfades

[PDF_fill](#) Füllt den aktuellen Pfad

[PDF_fill_stroke](#) Füllt den aktuellen Pfad und zeichnet eine Linie entlang des Pfades

[PDF_closepath_fill_stroke](#) Schließt, füllt und zeichnet eine Linie entlang des Pfades

[PDF_endpath](#) Beendet den aktuellen Pfad

[PDF_clip](#) Begrenzt alle Zeichenoperation auf den aktuellen Pfad

[PDF_setgray_fill](#) Setzt die Füllfarbe auf einen Grauwert

[PDF_setgray_stroke](#) Setzt die Zeichenfarbe auf einen Grauwert

[PDF_setgray](#) Setzt die Zeichen- und Füllfarbe auf einen Grauwert

[PDF_setrgbcolor_fill](#) Setzt die Füllfarbe auf einen Farbwert

[PDF_setrgbcolor_stroke](#) Setzt die Zeichenfarbe auf einen Farbwert

[PDF_setrgbcolor](#) Setzt die Zeichen- und Füllfarbe auf einen Farbwert

[PDF_add_outline](#) Fügt Lesemarke zur aktuellen Seite hinzu

[PDF_set_transition](#) Setzt den Übergang zur nächsten Seite

[PDF_set_duration](#) Setzt die Zeitdauer bis zur nächsten Seite

[PDF_open_gif](#) Öffnet ein GIF-Bild

[PDF_open_png](#) Öffnet ein PNG-Bild

[PDF_open_memory_image](#) Liest ein Bild, das mit PHP erzeugt wurde

[PDF_open_jpeg](#) Öffnet ein JPEG-Bild

[PDF_close_image](#) Schließt ein Bild

[PDF_place_image](#) Plaziert ein Bild auf der Seite

[PDF_put_image](#) Speichert ein Bild im PDF-Dokument für späteren Gebrauch

[PDF_execute_image](#) Plaziert ein gespeichertes Bild auf der Seite

[pdf_add_annotation](#) Fügt eine Anmerkung hinzu

[PDF_set_border_style](#) Setzt den Stil der Umrandung von Verweisen und Anmerkungen

[PDF_set_border_color](#) Setzt die Farbe der Umrandung von Verweisen und Anmerkungen

[PDF_set_border_dash](#) Setzt das Muster für gestrichelte Linien als Umrandung von Verweisen und Anmerkungen

[Zurück](#)

[Anfang](#)

[Vor](#)

OCIInternalDebug

[Hoch](#)

PDF_set_info

PDF_set_info

(unknown)

PDF_set_info -- Setzt Info-Feld der Dokumenteninformation

Beschreibung

`void pdf_set_info (int PDF Dokument, string Feldname, string Wert)`

Die Funktion **PDF_set_info()** setzt ein Feld der Dokumenteninformation. Mögliche Informationsfelder die gesetzt werden können sind 'Subject', 'Title', 'Creator', 'Author', 'Keywords' sowie ein Feld mit einem beliebigen Namen. Die Funktion kann vor dem Beginnen einer Seite aufgerufen werden.

Beispiel 1 Dokumenteninformation setzen

```
<?php
$fd = fopen("test.pdf", "w");
$pdfdoc = pdf_open($fd);
pdf_set_info($pdfdoc, "Author", "Uwe Steinmann");
pdf_set_info($pdfdoc, "Creator", "Uwe Steinmann");
pdf_set_info($pdfdoc, "Title", "Testing Info Fields");
pdf_set_info($pdfdoc, "Subject", "Test");
pdf_set_info($pdfdoc, "Keywords", "Test, Fields");
pdf_set_info($pdfdoc, "CustomField", "What ever makes sense");
pdf_begin_page($pdfdoc, 595, 842);
pdf_end_page($pdfdoc);
pdf_close($pdfdoc);
?>
```

Anmerkung: Diese Funktion ersetzt **PDF_set_info_keywords()**, **PDF_set_info_title()**, **PDF_set_info_subject()**, **PDF_set_info_creator()**, **PDF_set_info_sybject()**.

PDF_open

(unknown)

PDF_open -- Öffnet ein neues PDF-Dokument

Beschreibung

```
int pdf_open (int Datei-Deskriptor, int Info)
```

Die Funktion **PDF_open()** öffnet ein neues PDF-Dokument. The corresponding file has to be opened with [fopen\(\)](#) and the file descriptor passed as argument *file*. *info* is the info structure that has to be created with **pdf_get_info()**. The info structure will be deleted within this function.

Anmerkung: The return value is needed as the first parameter in all other functions writing to the PDF-Dokument.

Anmerkung: This function does not allow the second parameter if pdflib 2.0 support is activated.

Siehe auch [fopen\(\)](#), **PDF_get_info()**, **PDF_close()**.

PDF_close

(unknown)

PDF_close -- Schließt ein PDF-Dokument

Beschreibung

void **pdf_close** (int PDF-Dokument)

Die **PDF_close()** Function schließt ein PDF-Dokument.

Anmerkung: In Folge einer unsauberen Implementation von pdflib 0.6 wird durch diesen Befehl auch die Datei geschlossen. Dies sollte nicht passieren, weil pdflib die Datei auch nicht öffnet, sondern eine bereits geöffnete Datei beim Aufruf von **PDF_open()** erwartet. Folglich sollte die Datei auch nicht in pdflib geschlossen werden. Um diese zu korrigieren, sollten sie die Zeile 190 aus der Datei p_basic.c in den pdflib 0.6 Quellen entfernen. Bei späteren Versionen tritt dieses Problem nicht auf.

Siehe auch **PDF_open()**, [fclose\(\)](#).

PDF_begin_page

(unknown)

PDF_begin_page -- Beginnt eine neue Seite

Beschreibung

void **pdf_begin_page** (int PDF-Dokument, double Breite, double Höhe)

Die Funktion **PDF_begin_page()** beginnt eine neue Seite mit Höhe *Höhe* und mit der Breite *Breite*. Um ein gültiges Dokument zu erzeugen muss mindestens eine Seite erzeugt werden, die mit **PDF_end_page()** beendet wird.

Siehe auch **PDF_end_page()**.

PDF_end_page

(unknown)

PDF_end_page -- Beendet eine Seite

Beschreibung

void **pdf_end_page** (int PDF-Dokument)

Die Funktion **PDF_end_page()** beendet eine Seite. Sobald eine Seite beendet ist können keine Veränderungen an ihr mehr vorgenommen werden.

Siehe auch **PDF_begin_page()**.

PDF_show

(unknown)

PDF_show -- Schreibt einen Text an die aktuelle Position

Beschreibung

void **pdf_show** (int PDF-Dokument, string Text)

Die Funktion **PDF_show()** gibt den Text *Text* an der aktuellen Textposition in dem aktuellen Zeichensatz aus.

Siehe auch **PDF_show_xy()**, **PDF_set_text_pos()**, **PDF_set_font()**.

[Zurück](#)[Anfang](#)[Vor](#)

PDF_end_page

[Hoch](#)

PDF_show_boxed

PDF_show_boxed

(unknown)

PDF_show_boxed -- Schreibt einen Text in eine Box

Beschreibung

void **pdf_show_boxed** (int PDF-Dokument, string text, double x-Koor, double y-Koor, double Breite, double Höhe, string Modus)

Die Funktion **PDF_show_boxed()** gibt den Text *text* in einer Box aus, dessen linke untere Ecke an der Position (*x-coor*, *y-coor*) ist. Die Größe der Box ist *Höhe* mal *Breite*. Der Parameter *Modus* gibt an wie der Text gesetzt wird. Wenn *Höhe* und *Breite* Null sind, darf *Modus* auf "left", "right" oder "center" gesetzt werden. Wenn *Höhe* oder *Breite* ungleich Null sind zusätzlich die Werte "justify" and "fulljustify" erlaubt.

See also **PDF_show()**, **PDF_show_xy()**.

PDF_show_xy

(unknown)

PDF_show_xy -- Schreibt einen Text an die angegebene Position

Beschreibung

void **pdf_show_xy** (int PDF-Dokument, string Text, double x-Koor, double y-Koor)

Die Funktion **PDF_show_xy()** gibt den Text *Text* an der Position (*x-Koor*, *y-Koor*) aus.

Siehe auch **PDF_show()**.

[Zurück](#)[Anfang](#)[Vor](#)

PDF_show_boxed

[Hoch](#)

PDF_set_font

PDF_set_font

(unknown)

PDF_set_font -- Wählt einen Zeichensatz und dessen Größe aus

Beschreibung

void **pdf_set_font** (int PDF-Dokument, string Zeichensatzname, double Größe, string Kodierung [, int einbetten])

Die Funktion **PDF_set_font()** setzt den aktuellen Zeichensatz, die Zeichensatzgröße und die Kodierung. Bei Verwendung von pdflib 0.6 müssen die Adobe Zeichensatz-Metriken (afm-Dateien) für den Zeichensatz in dem voreingestellten Verzeichnis abgelegt werden (voreingestellt ist ./font).

Der vorletzte Parameter *encoding* gibt die Kodierung des Zeichensatzes an. Er kann die folgenden Werte annehmen: 'buitin', 'ebcdic', 'macroman', 'macexpert', 'winansi', 'host'. 'winansi' oder 'host' ist im übrigen häufig eine gute Wahl. winansi is often a good choice. Wenn der letzte Parameter auf eins gesetzt ist, wird der Zeichensatz in das PDF-Dokument eingebettet, sonst nicht. Einen Zeichensatz einzubetten, ist immer dann eine gute Wahl, wenn der Zeichensatz nicht sehr weit verbreitet ist, und die Person, die das Dokument lesen soll, keinen Zugriff auf den Zeichensatz hat.

Anmerkung: Diese Funktion muss nach **PDF_begin_page()** aufgerufen werden um ein gültiges Dokument zu erhalten.

Anmerkung: Wenn der Zeichensatz über eine .upr-Datei referenziert wird, muss der Name in der afm-Datei mit dem Zeichensatznamen übereinstimmen. Andernfalls wird der Zeichensatz mit jedem Aufruf von **PDF_set_font()** erneut eingebettet. (Danek an Paul Haddon der dieses Problem gefunden hat.)

PDF_set_leading

(unknown)

PDF_set_leading -- Setzt den Abstand zwischen zwei Textzeilen

Beschreibung

void **pdf_set_leading** (int PDF-Dokument, double Abstand)

Die Funktion **PDF_set_leading()** setzt den Abstand zwischen zwei Textzeilen, die mit **PDF_continue_text()** ausgegeben werden.

Siehe auch **PDF_continue_text()**.

PDF_set_parameter

(unknown)

PDF_set_parameter -- Setzt verschiedene Parameter

Beschreibung

void **pdf_set_parameter** (int PDF-Dokument, string Name, string Wert)

Die **PDF_set_parameter()** Funktion setzt verschiedene Parameter von pdflib, die vom Typ string sind.

Siehe auch **PDF_get_value()**, **PDF_set_value()**, **PDF_get_parameter()**.

PDF_get_parameter

(unknown)

PDF_get_parameter -- Liefert verschiedene Parameter

Beschreibung

string **pdf_get_parameter** (int PDF-Dokument, string Name, double Modifizierer)

Die **PDF_set_parameter()** Funktion liefert verschiedene Parameter von pdflib, die vom Typ string sind. Der Funktionsparameter *Modifizierer* spezifiziert den ausgewählten Parameter genauer.

Siehe auch **PDF_get_value()**, **PDF_set_value()**, **PDF_set_parameter()**.

PDF_set_value

(unknown)

PDF_set_value -- Setzt verschiedene numerische Werte

Beschreibung

void **pdf_set_value** (int PDF-Dokument, string Name, double Wert)

Die **PDF_set_value()** Funktion setzt verschiedene numerische Parameter von pdflib.

Siehe auch **PDF_get_value()**, **PDF_set_parameter()**, **PDF_get_parameter()**.

PDF_get_value

(unknown)

PDF_get_value -- Liefert verschiedene numerische Werte

Beschreibung

double **pdf_get_value** (int PDF-Dokument, string Name, double Modifizierer)

Die **PDF_get_value()** Funktion liefert verschiedene numerische Parameter von pdflib. Der Funktionsparameter *Modifizierer* spezifiziert den ausgewählten Parameter genauer.

Siehe auch **PDF_get_value()**, **PDF_set_value()**, **PDF_set_parameter()**.

PDF_set_text_rendering

(unknown)

PDF_set_text_rendering -- Bestimmt wie der Text ausgegeben werden soll

Beschreibung

void **pdf_set_text_rendering** (int PDF-Dokument, int Modus)

Die Funktion **PDF_set_text_rendering()** bestimmt wie die Zeichen einer Textausgabe dargestellt werden. Die möglichen Werte für den Parameter *Modus* sind 0=ausgefüllt, 1=nur die Umrandung, 2=ausgefüllt und die Umrandung, 3=unsichtbar, 4=ausgefüllt und Hinzunahme zum Begrenzungspfad (clipping path), 5=Umrandung und Hinzunahme zum Begrenzungspfad, 6=ausgefüllt, umrandet und Hinzunahme zum Begrenzungspfad, 7=nur Hinzunahme zum Begrenzungspfad.

PDF_set_horiz_scaling

(unknown)

PDF_set_horiz_scaling -- Setzt die horizontale Skalierung bei der Textausgabe

Beschreibung

void **pdf_set_horiz_scaling** (int PDF-Dokument, double Skalierung)

Die Funktion **PDF_set_horiz_scaling()** setzt den horizontalen Skalierungsfaktor auf *Skalierung* Prozent.

PDF_set_text_rise

(unknown)

PDF_set_text_rise -- Setzt die Textverschiebung

Beschreibung

void **pdf_set_text_rise** (int PDF-Dokument, double Verschiebung)

Die Funktion **PDF_set_text_rise()** setzt die Textverschiebung nach oben auf *Verschiebung* Punkte.

[Zurück](#)[Anfang](#)[Vor](#)

PDF_set_horiz_scaling

[Hoch](#)

PDF_set_text_matrix

PDF_set_text_matrix

(unknown)

PDF_set_text_matrix -- Setzt die Text-Matrix

Beschreibung

`void pdf_set_text_matrix (int PDF-Dokument, array Matrix)`

Die Funktion **PDF_set_text_matrix()** setzt die Matrix, welche die Transformation auf Textausgaben beschreibt. text font. Die Matrix muss als array mit sechs Parametern übergeben werden.

PDF_set_text_pos

(unknown)

PDF_set_text_pos -- Setzt die Textposition

Beschreibung

void **pdf_set_text_pos** (int PDF-Dokument, double x-Koor, double y-Koor)

Die Funktion **PDF_set_text_pos()** setzt die Position für die nächste Textausgabe mit der Funktion [pdf_show\(\)](#).

Siehe auch **PDF_show()**, **PDF_show_xy()**.

PDF_set_char_spacing

(unknown)

PDF_set_char_spacing -- Setzt den Abstand zwischen Zeichen

Beschreibung

void **pdf_set_char_spacing** (int PDF-Dokument, double Abstand)

Die Funktion **PDF_set_char_spacing()** setzt den Abstand zwischen zwei Zeichen.

Siehe auch **PDF_set_word_spacing()**, **PDF_set_leading()**.

PDF_set_word_spacing

(unknown)

PDF_set_word_spacing -- Setzt den Abstand zwischen Wörtern

Beschreibung

void **pdf_set_word_spacing** (int PDF-Dokument, double space)

Die Funktion **PDF_set_word_spacing()** setzt den Abstand zwischen zwei Wörtern.

Siehe auch **PDF_set_char_spacing()**, **PDF_set_leading()**.

PDF_skew

(unknown)

PDF_skew -- Schert das Koordinatensystem

Beschreibung

void **pdf_skew** (int PDF-Dokument, double alpha, double beta)

Die **PDF_skew()** Funktion schert das Koordinatensystem um *alpha* (x) und *beta* (y) Grad. *alpha* und *beta* dürfen weder 90 noch 270 Grad sein.

[Zurück](#)[Anfang](#)[Vor](#)[PDF_set_word_spacing](#)[Hoch](#)[PDF_continue_text](#)

PDF_continue_text

(unknown)

PDF_continue_text -- Schreibt den Text in die nächste Zeile

Beschreibung

void **pdf_continue_text** (int PDF-Dokument, string Text)

Die Funktion **PDF_continue_text()** gibt den Text *text* in der nächsten Zeile aus. Der Abstand zwischen zwei Zeilen kann mit der Funktion **PDF_set_leading()** gesetzt werden.

Siehe auch **PDF_show_xy()**, **PDF_set_leading()**, **PDF_set_text_pos()**.

[Zurück](#)

PDF_skew

[Anfang](#)[Hoch](#)[Vor](#)

PDF_stringwidth

PDF_stringwidth

(unknown)

PDF_stringwidth -- Liefert die benötigte Breite einer Zeichenkette mit dem aktuelle Zeichensatz

Beschreibung

double **pdf_stringwidth** (int PDF-Dokument, string Text)

Die Funktion **PDF_stringwidth()** liefert die Breite eines Textes *text* unter Verwendung des aktuellen Zeichensatzes. Um diese Funktion benutzen zu können muss zuvor ein Zeichensatz mit **PDF_set_font()** gesetzt worden sein.

Siehe auch **PDF_set_font()**.

PDF_save

(unknown)

PDF_save -- Sichert die aktuelle Umgebung

Beschreibung

void **pdf_save** (int PDF-Dokument)

Die Funktion **PDF_save()** sichert die aktuelle Umgebung. Sie wirkt wie das Postscript-Kommando gsave. Die Funktion ist sehr sinnvoll, wenn Sie einzelne Objekte drehen oder verschieben möchte ohne andere Objekte zu beeinflussen. Auf **PDF_save()** sollte immer die Funktion **PDF_restore()** folgen, um die ursprüngliche Umgebung wieder herzustellen.

Siehe auch **PDF_restore()**.

PDF_restore

(unknown)

PDF_restore -- Stellt eine zuvor gesicherte Umgebung wieder her

Beschreibung

void **pdf_restore** (int PDF-Dokument)

Die Funktion **PDF_restore()** stellt eine frühere Umgebung wieder her, die mit **PDF_save()** gesichert wurde. Die Funktion arbeitet wie das Postscript-Kommando grestore.

Beispiel 1 Save and Restore

```
<?php PDF_save($pdf);  
// Alle möglichen Rotationen, Transformationen, ...  
PDF_restore($pdf) ?>
```

Siehe auch **PDF_save()**.

PDF_translate

(unknown)

PDF_translate -- Setzt den Ursprung des Koordinatensystems

Beschreibung

void **pdf_translate** (int PDF-Dokument, double x-Koor, double y-Koor)

Die Funktion **PDF_translate()** setzt den Ursprung des Koordinatensystems auf den Punkt (*x-Koor*, *y-Koor*) relativ zum aktuellen Koordinatenursprung. Das folgende Beispiel zeichnet eine Linie vom Punkt (0, 0) zum Punkt (200, 200) relativ zum anfänglichen Ursprung. Die aktuelle Position muss nach **PDF_translate()** und vor dem weiteren Zeichnen gesetzt werden.

Beispiel 1 Translation

```
<?php PDF_moveto($pdf, 0, 0);  
PDF_lineto($pdf, 100, 100);  
PDF_stroke($pdf);  
PDF_translate($pdf, 100, 100);  
PDF_moveto($pdf, 0, 0);  
PDF_lineto($pdf, 100, 100);  
PDF_stroke($pdf);  
?>
```

PDF_scale

(unknown)

PDF_scale -- Setzt den Skalierungsfaktor

Beschreibung

void **pdf_scale** (int PDF-Dokument, double x-Skalierung, double y-Skalierung)

Die Funktion **PDF_scale()** setzt den Skalierungsfaktor in beiden Richtungen. Das folgende Beispiel skaliert in x- und y-Richtung um den Faktor 72. Die folgende Linie hat damit eine Länge von etwa 1.41 Inch.

Beispiel 1 Scaling

```
<?php PDF_scale($pdf, 72.0, 72.0);
PDF_lineto($pdf, 1, 1);
PDF_stroke($pdf);
?>
```

PDF_rotate

(unknown)

PDF_rotate -- Setzt die Rotation

Beschreibung

void **pdf_rotate** (int PDF-Dokument, double Winkel)

Die Funktion **PDF_rotate()** setzt die Drehung in auf *Winkel* Grad.

PDF_setflat

(unknown)

PDF_setflat -- Sets flatness

Beschreibung

void **pdf_setflat** (int PDF-Dokument, double Wert)

Die Funktion **PDF_setflat()** setzt the flatness auf einen Wert zwischen 0 und 100.

[Zurück](#)[Anfang](#)[Vor](#)

PDF_rotate

[Hoch](#)

PDF_setlinejoin

PDF_setlinejoin

(unknown)

PDF_setlinejoin -- Setzt die Verbindungsart von Linien

Beschreibung

void **pdf_setlinejoin** (int PDF-Dokument, long Wert)

Die Funktion **PDF_setlinejoin()** setzt die Verbindungsart von Linien auf einen Wert zwischen 0 und 2.

[Zurück](#)[Anfang](#)[Vor](#)

PDF_setflat

[Hoch](#)

PDF_setlinecap

PDF_setlinecap

(unknown)

PDF_setlinecap -- Setzt den Type der Linienenden

Beschreibung

void **pdf_setlinecap** (int PDF-Dokument, int Typ)

Die Funktion **PDF_setlinecap()** setzt den Typ der Linienende auf einen Wert zwischen 0 und 2.

[Zurück](#)[Anfang](#)[Vor](#)

PDF_setlinejoin

[Hoch](#)

PDF_setmiterlimit

PDF_setmiterlimit

(unknown)

PDF_setmiterlimit -- Sets miter limit

Beschreibung

void **pdf_setmiterlimit** (int PDF-Dokument, double Wert)

Die Funktion **PDF_setmiterlimit()** setzt the miter limit größer oder gleich 1.

[Zurück](#)[Anfang](#)[Vor](#)

PDF_setlinecap

[Hoch](#)

PDF_setlinewidth

PDF_setlinewidth

(unknown)

PDF_setlinewidth -- Setzt die Linienbreite

Beschreibung

void **pdf_setlinewidth** (int PDF-Dokument, double Breite)

Die Funktion **PDF_setlinewidth()** setzt die Linienbreite auf *Breite* Punkte.

PDF_setdash

(unknown)

PDF_setdash -- Setzt das Muster für gestrichelte Linien

Beschreibung

void **pdf_setdash** (int PDF-Dokument, double Weiß, double Schwarz)

Die Funktion **PDF_setdash()** setzt das Muster für gestrichelte Linien auf *Weiß* weiße Punkte und *Schwarz* schwarze Punkte. Wenn beide Werte 0 sind wird die Linie durchgezogen.

PDF_moveto

(unknown)

PDF_moveto -- Setzt die aktuelle Position

Beschreibung

void **pdf_moveto** (int PDF-Dokument, double x-Koor, double y-Koor)

Die Funktion **PDF_moveto()** setzt die aktuelle Position auf den Punkt (*x-Koor*, *y-koor*).

[Zurück](#)[Anfang](#)[Vor](#)

PDF_setdash

[Hoch](#)

PDF_curveto

PDF_curveto

(unknown)

PDF_curveto -- Zeichnet eine Kurve

Beschreibung

void **pdf_curveto** (int PDF-Dokument, double x1, double y1, double x2, double y2, double x3, double y3)

Die Funktion **PDF_curveto()** zeichnet eine Bezier-Kurve vom aktuellen Punkt zu dem Punkt (x3, y3) unter Verwendung von (x1, y1) und (x2, y2) als Kontrollpunkte.

Siehe auch **PDF_moveto()**, **PDF_lineto()**, **PDF_stroke()**.

PDF_lineto

(unknown)

PDF_lineto -- Zeichnet eine Linie

Beschreibung

void **pdf_lineto** (int PDF-Dokument, double x-Koor, double y-Koor)

Die Funktion **PDF_lineto()** zeichnet eine Linie vom aktuellen Punkt zu dem Punkt (*x-koor*, *y-koor*).

Siehe auch **PDF_moveto()**, **PDF_curveto()**, **PDF_stroke()**.

PDF_circle

(unknown)

PDF_circle -- Zeichnet einen Kreis

Beschreibung

void **pdf_circle** (int PDF-Dokument, double x-Koor, double y-Koor, double Radius)

Die Funktion **PDF_circle()** zeichnet einen Kreis mit dem Mittelpunkt an der Position (*x-koor*, *y-koor*) und dem Radius *Radius*.

Siehe auch **PDF_arc()**, **PDF_stroke()**.

PDF_arc

(unknown)

PDF_arc -- Zeichnet einen Kreisbogen

Beschreibung

void **pdf_arc** (int PDF-Dokument, double x-Koor, double y-Koor, double Radius, double Start, double Ende)

Die Funktion **PDF_arc()** zeichnet einen Kreisbogen mit dem Mittelpunkt an der Position (*x-koor*, *y-koor*) und dem Radius *radius*, angefangen beim Winkel *Start* und beendet mit dem Winkel *Ende*.

Siehe auch **PDF_circle()**, **PDF_stroke()**.

PDF_rect

(unknown)

PDF_rect -- Zeichnet ein Rechteck

Beschreibung

void **pdf_rect** (int PDF-Dokument, double x-koor, double y-koor, double Breite, double Höhe)

Die Funktion **PDF_rect()** zeichnet ein Rechteck mit der unteren Ecke an der Position (*x-Koor*, *y-Koor*). Die Breite wird auf *width* gesetzt. Die Höhe wird auf *height* gesetzt.

Siehe auch **PDF_stroke()**.

PDF_closepath

(unknown)

PDF_closepath -- Schließt einen Pfad

Beschreibung

void **pdf_closepath** (int PDF-Dokument)

Die Funktion **PDF_closepath()** schließt den aktuellen Pfad. Viele Funktion wie z.B. **PDF_moveto()**, **PDF_circle()** und **PDF_rect()** starten einen neuen Pfad.

[Zurück](#)

[Anfang](#)

[Vor](#)

PDF_rect

[Hoch](#)

PDF_stroke

PDF_stroke

(unknown)

PDF_stroke -- Zeichnet eine Linie entlang eines Pfades

Beschreibung

void **pdf_stroke** (int PDF-Dokument)

Die Funktion **PDF_stroke()** zeichnet eine Linie entlang des aktuellen Pfades. Der aktuelle Pfad setzt sich beispielsweise aus allen Anweisung, die eine Linie gezeichnet haben zusammen. Ohne Aufruf dieser Funktion würde die Linie nicht erscheinen.

Siehe auch **PDF_closepath()**, **PDF_closepath_stroke()**.

PDF_closepath_stroke

(unknown)

PDF_closepath_stroke -- Schließt einen Pfad und zeichnet eine Linie entlang des Pfades

Beschreibung

void **pdf_closepath_stroke** (int PDF-Dokument)

Die Funktion **PDF_closepath_stroke()** ist eine Kombination von **PDF_closepath()** und **PDF_stroke()**. Sie löscht gleichzeitig den Pfad.

Siehe auch **PDF_closepath()**, **PDF_stroke()**.

[Zurück](#)[Anfang](#)[Vor](#)

PDF_stroke

[Hoch](#)

PDF_fill

PDF_fill

(unknown)

PDF_fill -- Füllt den aktuellen Pfad

Beschreibung

void **pdf_fill** (int PDF-Dokument)

Die Funktion **PDF_fill()** füllt das Innere des aktuellen Pfads mit der aktuellen Füllfarbe.

Siehe auch **PDF_closepath()**, **PDF_stroke()**, **PDF_setgray_fill()**, **PDF_setgray()**, **PDF_setrgbcolor_fill()**, **PDF_setrgbcolor()**.

PDF_fill_stroke

(unknown)

PDF_fill_stroke -- Füllt den aktuellen Pfad und zeichnet eine Linie entlang des Pfades

Beschreibung

void **pdf_fill_stroke** (int PDF-Dokument)

Die Funktion **PDF_fill_stroke()** füllt das Innere des aktuellen Pfades mit der aktuellen Füllfarbe und zeichnet zudem eine Linie entlang des Pfades.

Siehe auch **PDF_closepath()**, **PDF_stroke()**, **PDF_fill()**, **PDF_setgray_fill()**, **PDF_setgray()**, **PDF_setrgbcolor_fill()**, **PDF_setrgbcolor()**.

PDF_closepath_fill_stroke

(unknown)

PDF_closepath_fill_stroke -- Schließt, füllt und zeichnet eine Linie entlang des Pfades

Beschreibung

void **pdf_closepath_fill_stroke** (int PDF-Dokument)

Die Funktion **PDF_closepath_fill_stroke()** schließt, füllt das Innere des aktuellen Pfads mit der aktuellen Füllfarbe und zeichnet eine Linie entlang des Pfades.

Siehe auch **PDF_closepath()**, **PDF_stroke()**, **PDF_fill()**, **PDF_setgray_fill()**, **PDF_setgray()**, **PDF_setrgbcolor_fill()**, **PDF_setrgbcolor()**.

PDF_endpath

(unknown)

PDF_endpath -- Beendet den aktuellen Pfad

Beschreibung

void **pdf_endpath** (int PDF-Dokument)

Die Funktion **PDF_endpath()** beendet den aktuellen Pfad ohne ihn zu schließen.

Siehe auch **PDF_closepath()**.

PDF_clip

(unknown)

PDF_clip -- Begrenzt alle Zeichenoperation auf den aktuellen Pfad

Beschreibung

```
void pdf_clip (int PDF-Dokument)
```

Die Funktion **PDF_clip()** begrenzt alle Zeichenoperationen auf das innere des aktuellen Pfads.

[Zurück](#)[Anfang](#)[Vor](#)

PDF_endpath

[Hoch](#)

PDF_setgray_fill

PDF_setgray_fill

(unknown)

PDF_setgray_fill -- Setzt die Füllfarbe auf einen Grauwert

Beschreibung

void **pdf_setgray_fill** (int PDF-Dokument, double Grauwert)

Die Funktion **PDF_setgray_fill()** setzt den aktuellen Grauwert mit dem ein Pfad gefüllt wird.

Siehe auch **PDF_setrgbcolor_fill()**.

PDF_setgray_stroke

(unknown)

PDF_setgray_stroke -- Setzt die Zeichenfarbe auf einen Grauwert

Beschreibung

void **pdf_setgray_stroke** (int PDF-Dokument, double Grauwert)

Die Funktion **PDF_setgray_stroke()** setzt die aktuellen Grauwert zum Zeichen von Linien.

Siehe auch **PDF_setrgbcolor_stroke()**.

PDF_setgray

(unknown)

PDF_setgray -- Setzt die Zeichen- und Füllfarbe auf einen Grauwert

Beschreibung

void **pdf_setgray** (int PDF-Dokument, double Grauwert)

Die Funktion **PDF_setgray()** setzt den aktuellen Grauwert zum Zeichnen und Füllen.

Siehe auch **PDF_setrgbcolor_stroke()**, **PDF_setrgbcolor_fill()**.

[Zurück](#)[Anfang](#)[Vor](#)

PDF_setgray_stroke

[Hoch](#)

PDF_setrgbcolor_fill

PDF_setrgbcolor_fill

(unknown)

PDF_setrgbcolor_fill -- Setzt die Füllfarbe auf einen Farbwert

Beschreibung

void **pdf_setrgbcolor_fill** (int PDF-Dokument, double Rotwert, double Grünwert, double Blauwert)

Die Funktion **PDF_setrgbcolor_fill()** setzt die aktuelle Füllfarbe auf einen RGB-Farbwert.

Siehe auch **PDF_setrgbcolor_fill()**.

[Zurück](#)[Anfang](#)[Vor](#)

PDF_setgray

[Hoch](#)

PDF_setrgbcolor_stroke

PDF_setrgbcolor_stroke

(unknown)

PDF_setrgbcolor_stroke -- Setzt die Zeichenfarbe auf einen Farbwert

Beschreibung

void **pdf_setrgbcolor_stroke** (int PDF-Dokument, double Rotwert, double Grünwert, double Blauwert)

Die Funktion **PDF_setrgbcolor_stroke()** setzt die aktuelle Zeichenfarbe auf einen RGB-Farbwert.

Siehe auch **PDF_setrgbcolor_stroke()**.

PDF_setrgbcolor

(unknown)

PDF_setrgbcolor -- Setzt die Zeichen- und Füllfarbe auf einen Farbwert

Beschreibung

void **pdf_setrgbcolor** (int PDF-Dokument, double Rotwert, double Grünwert, double Blauwert)

Die Funktion **PDF_setrgbcolor_stroke()** setzt sowohl die aktuelle Füll- und Zeichenfarbe auf einen RGB-Farbwert.

Siehe auch **PDF_setrgbcolor_stroke()**, **PDF_setrgbcolor_fill()**.

PDF_add_outline

(unknown)

PDF_add_outline -- Fügt Lesemarke zur aktuellen Seite hinzu

Beschreibung

```
int pdf_add_outline (int PDF-Dokument, string Text [, int Mutter [,  
int offen]])
```

Die Funktion **PDF_add_outline()** fügt der aktuellen Seite eine Lesemarke mit dem Text *text* hinzu.

Die Lesemarke wird als Kind von *Mutter* hinzugefügt und wird geöffnet dargestellt von der Parameter *offen* ungleich 0 ist. Der Rückgabewert ist die Kennung der Lesemarke und kann als Mutter weiterer Lesemarken dienen. Damit lassen sich Hierarchien von Lesemarken aufbauen.

Leider macht pdflib keine Kopie des Textes, so dass dies in PHP geschehen muss. Zur Zeit wird dieser Text nicht wieder freigegeben, allerdings kümmert sich der PHP-Speichermanager darum.

PDF_set_transition

(unknown)

PDF_set_transition -- Setzt den Übergang zur nächsten Seite

Beschreibung

void pdf_set_transition (int PDF-Dokument, int Transition)

Die Funktion **PDF_set_transition()** setzt die Art der Transition zur folgenden Seite. Die möglichen Werte des Parameters *transition* sind:

- 0 für keine,
- 1 für zwei, über den Bildschirm kriechende Linien, die die neue Seite entstehen lassen,
- 2 für mehrere, über den Bildschirm kriechende Linien, die die neue Seite entstehen lassen,
- 3 für einen Kasten der die Seite entstehen läßt,
- 4 für eine einzelne, über den Bildschirm kriechende Linie, die die neue Seite entstehen lassen,
- 5, wenn die Seite sich auflösen soll,
- 6 für eine Auflösungseffekt, der sich über die Seite bewegt,
- 7, wenn die Seite von der neuen Seite einfach ersetzt wird (voreingestellt).

Siehe auch **PDF_set_duration()**.

PDF_set_duration

(unknown)

PDF_set_duration -- Setzt die Zeitdauer bis zur nächsten Seite

Beschreibung

void **pdf_set_duration** (int PDF-Dokument, double Dauer)

Die Funktion **PDF_set_duration()** setzt die Dauer bis zur nächsten Seite automatisch umgeblättert wird.

Siehe auch **PDF_set_transition()**.

PDF_open_gif

(unknown)

PDF_open_gif -- Öffnet ein GIF-Bild

Beschreibung

`int pdf_open_gif (int PDF-Dokument, string Dateiname)`

Die Funktion **PDF_open_gif()** öffnet ein Bild im GIF-Format, das in der Datei *Dateiname* gespeichert ist. Die Funktion liefert ein PDF-Bild zurück.

Beispiel 1 Einbinden eines GIF Bildes

```
<?php
$im = PDF_open_gif($pdf, "test.gif");
pdf_place_image($pdf, $im, 100, 100, 1);
pdf_close_image($pdf, $im);
?>
```

Siehe auch **PDF_close_image()**, **PDF_open_jpeg()**, **PDF_open_png()**, **PDF_open_memory_image()**, **PDF_execute_image()**, **PDF_place_image()**, **PDF_put_image()**.

PDF_open_png

(unknown)

PDF_open_png -- Öffnet ein PNG-Bild

Beschreibung

`int pdf_open_png (int PDF-Dokument, string Dateiname)`

Die Funktion **PDF_open_png()** öffnet ein Bild im PNG-Format, das in der Datei *Dateiname* gespeichert ist. Die Funktion liefert ein PDF-Bild zurück.

Beispiel 1 Einbinden eines PNG Bildes

```
<?php
$im = PDF_open_png($pdf, "test.png");
pdf_place_image($pdf, $im, 100, 100, 1);
pdf_close_image($pdf, $im);
?>
```

Siehe auch **PDF_close_image()**, **PDF_open_jpeg()**, **PDF_open_gif()**, **PDF_open_memory_image()**, **PDF_execute_image()**, **PDF_place_image()**, **PDF_put_image()**.

PDF_open_memory_image

(unknown)

PDF_open_memory_image -- Liest ein Bild, das mit PHP erzeugt wurde

Beschreibung

int pdf_open_memory_image (int PDF-Dokument, int Bild)

Die Funktion **PDF_open_memory_image()** nimmt ein Bild, das mit den PHP eigenen Funktion zur Bilderzeugung erstellt wurde und macht es für das PDF-Dokument verfügbar. Die Funktion liefert ein PDF-Bild zurück.

Beispiel 1 Einfügen eines Bildes

```
<?php
$im = ImageCreate(100, 100);
$col = ImageColorAllocate($im, 80, 45, 190);
ImageFill($im, 10, 10, $col);
$vim = PDF_open_memory_image($pdf, $im);
ImageDestroy($im);
pdf_place_image($pdf, $vim, 100, 100, 1);
pdf_close_image($pdf, $vim);
?>
```

Siehe auch **PDF_close_image()**, **PDF_open_jpeg()**, **PDF_open_gif()**, **PDF_execute_image()**, **PDF_place_image()**, **PDF_put_image()**.

PDF_open_jpeg

(unknown)

PDF_open_jpeg -- Öffnet ein JPEG-Bild

Beschreibung

`int pdf_open_jpeg (int PDF-Dokument, string Dateiname)`

Die Funktion **PDF_open_jpeg()** öffnet ein Bild im JPEG-Format, das in der Datei *Dateiname* gespeichert ist. Die Funktion liefert ein PDF-Bild zurück.

Siehe auch **PDF_close_image()**, **PDF_open_gif()**, **PDF_open_memory_image()**, **PDF_execute_image()**, **PDF_place_image()**, **PDF_put_image()**.

PDF_close_image

(unknown)

PDF_close_image -- Schließt ein Bild

Beschreibung

void **pdf_close_image** (int Bild)

Die Funktion **PDF_close_image()** schließt ein Bild, das zuvor mit einer der **PDF_open_xxx()** Funktionen geöffnet wurde.

Siehe auch **PDF_open_jpeg()**, **PDF_open_gif()**, **PDF_open_memory_image()**.

[Zurück](#)[Anfang](#)[Vor](#)

PDF_open_jpeg

[Hoch](#)

PDF_place_image

PDF_place_image

(unknown)

PDF_place_image -- Plaziert ein Bild auf der Seite

Beschreibung

void **pdf_place_image** (int PDF-Dokument, int Bild, double x-Koor, double y-Koor, double scale)

Die Funktion **PDF_place_image()** plaziert ein Bild auf der Seite an Position (*x-Koor*, *x-Koor*). Das Bild kann gleichzeitig skaliert werden.

Siehe auch **PDF_put_image()**.

PDF_put_image

(unknown)

PDF_put_image -- Speichert ein Bild im PDF-Dokument für späteren Gebrauch

Beschreibung

void **pdf_put_image** (int PDF-Dokument, int image)

Die Funktion **PDF_put_image()** platziert ein Bild in dem PDF-Dokument ohne es anzuzeigen. Das gespeicherte Bild kann durch die Funktion **PDF_execute_image()** so oft wie notwendig angezeigt werden. Dies ist sinnvoll, wenn das gleiche Bild mehrfach genutzt werden soll, um die Dateigröße des PDF-Dokuments klein zu halten. Der Gebrauch von **PDF_put_image()** und **PDF_execute_image()** ist insbesondere bei größeren Bildern zu empfehlen, wenn sie mehrfach verwendet werden.

Anmerkung: Diese Funktion ist bedeutungslos seit der Version 2.01 von pdflib. Es wird lediglich eine Warnung ausgegeben.

Siehe auch **PDF_put_image()**, **PDF_place_image()**, **PDF_execute_image()**.

PDF_execute_image

(unknown)

PDF_execute_image -- Plaziert ein gespeichertes Bild auf der Seite

Beschreibung

void **pdf_execute_image** (int PDF-Dokument, int image, double x-Koor, double y-Koor, double Skalierung)

Die Funktion **PDF_execute_image()** zeigt ein Bild, das zuvor mit **PDF_put_image()** in dem PDF-Dokument abgelegt wurde.

Das Bild kann gleichzeitig skaliert werden. Ein Skalierungsfaktor von 1.0 zeigt das Bild in Originalgröße.

Anmerkung: Diese Funktion ist bedeutungslos seit der Version 2.01 von pdflib. Es wird lediglich eine Warnung ausgegeben.

Beispiel 1 Multiple show of an image

```
<?php
$im = ImageCreate(100, 100);
$col1 = ImageColorAllocate($im, 80, 45, 190);
ImageFill($im, 10, 10, $col1);
$pim = PDF_open_memory_image($pdf, $im);
pdf_put_image($pdf, $pim);
pdf_execute_image($pdf, $pim, 100, 100, 1);
pdf_execute_image($pdf, $pim, 200, 200, 2);
pdf_close_image($pdf, $pim);
?>
```

pdf_add_annotation

(PHP3 >= 3.0.12, PHP4 >= 4.0b2)

pdf_add_annotation -- Fügt eine Anmerkung hinzu

Beschreibung

void **pdf_add_annotation** (int PDF-Dokument, double *llx*, double *lly*, double *urx*, double *ury*, string Titel, string Inhalt)

Die Funktion **pdf_add_annotation()** fügt eine Anmerkung mit der unteren linken Ecke an der Position (*llx*, *lly*) und der oberen rechten Ecke an der Position (*urx*, *ury*) ein.

[Zurück](#)[Anfang](#)[Vor](#)

PDF_execute_image

[Hoch](#)

PDF_set_border_style

PDF_set_border_style

(unknown)

PDF_set_border_style -- Setzt den Stil der Umrandung von Verweisen und Anmerkungen

Beschreibung

void **pdf_set_border_style** (int PDF-Dokument, string Stil, double Breite)

Die Funktion **PDF_set_border_style()** setzt den Stil und die Breite der Umrandung von Verweisen und Anmerkungen. Der Parameter *Stil* kann die Werte 'solid' oder 'dashed' annehmen.

Siehe auch **PDF_set_border_color()**, **PDF_set_border_dash()**.

PDF_set_border_color

(unknown)

PDF_set_border_color -- Setzt die Farbe der Umrandung von Verweisen und Anmerkungen

Beschreibung

void **pdf_set_border_color** (int PDF-Dokument, double Rot, double Grün, double Blau)

Die Funktion **PDF_set_border_color()** setzt die Farbe der Umrandung von Verweisen und Anmerkungen. Die drei Farbkomponenten haben eine Wert zwischen 0.0 und 1.0.

Siehe auch **PDF_set_border_style()**, **PDF_set_border_dash()**.

PDF_set_border_dash

(unknown)

PDF_set_border_dash -- Setzt das Muster für gestrichelte Linien als Umrandung von Verweisen und Anmerkungen

Beschreibung

`void pdf_set_border_dash (int PDF-Dokument, double Schwarz, double Weiß)`

Die Funktion **PDF_set_border_dash()** setzt die Länge der schwarzen und weißen Bereiche einer gestrichelten Umrandung von Verweisen und Anmerkungen. links and annotations.

Siehe auch **PDF_set_border_style()**, **PDF_set_border_color()**.

XLIV. Payflow Pro functions

This extension allows you to process credit cards using Verisign Payment Services, formerly known as Signio (<http://www.verisign.com/payment/>).

These functions are only available if PHP has been compiled with the `--with-pfpro[=DIR]` option. You will require the appropriate SDK for your platform, which may be downloaded [from within the manager interface](#) once you have registered.

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`.

When using these functions, you should call [pfpro_init\(\)](#) once before you process any transactions, and [pfpro_cleanup\(\)](#) afterwards. You may make any number of transactions using [pfpro_process\(\)](#) between the two.

These functions have been added in PHP 4.0.2.

Inhaltsverzeichnis

[pfpro_init](#) Initialises the Payflow Pro library

[pfpro_cleanup](#) Shuts down the Payflow Pro library

[pfpro_process](#) Process a transaction with Payflow Pro

[pfpro_process_raw](#) Process a raw transaction with Payflow Pro

[pfpro_version](#) Returns the version of the Payflow Pro software

pfpro_init

(PHP4 CVS only)

pfpro_init -- Initialises the Payflow Pro library

Description

```
void pfpro_init(void);
```

pfpro_init() is used to initialise the Payflow Pro library. It must be called before processing any transactions.

See also [pfpro_cleanup\(\)](#).

pfpro_cleanup

(PHP4 CVS only)

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 your script terminates.

See also [pfpro_init\(\)](#).

pfpro_process

(PHP4 CVS only)

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.

Anmerkung: You must have called [pfpro_init\(\)](#) before calling this function.

Beispiel 1 Payflow Pro example

```
<?php

pfpro_init();

$transaction = array(USER      => 'mylogin',
                      PWD      => 'mypassword',
                      TRXTYPE  => 'S',
                      TENDER   => 'C',
                      AMT       => 1.50,
                      ACCT      => '4111111111111111',
                      EXPDATE   => '0904'
                      );

$response = pfpro_process($transaction);

if (!$response) {
    die("Couldn't establish link to Signio.\n");
}

echo "Signio 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()

?>
```

[Zurück](#)[Anfang](#)[Vor](#)[pfpro_cleanup](#)[Hoch](#)[pfpro_process_raw](#)

pfpro_process_raw

(PHP4 CVS only)

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.

Anmerkung: You must have called [pfpro_init\(\)](#) before calling this function.

Beispiel 1 Payflow Pro raw example

```
<?php
pfpro_init();

$response =
pfpro_process("USER=mylogin&PWD[5]=m&ndy&TRXTYPE=S&TENDER=C&AMT=1.50&ACCT=4111111111111111&EXPDATE=0904");

if (!$response) {
    die("Couldn't establish link to Signio.\n");
}

echo "Signio raw response was ".$response;

pfpro_cleanup()

?>
```



pfpro_version

(PHP4 CVS only)

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.

XLV. PHP Optionen und Informationen

Inhaltsverzeichnis

- [error_log](#) Sendet eine Fehlermeldung
- [error_reporting](#) Gibt an, wie PHP-Fehlermeldungen gezeigt werden
- [extension_loaded](#) Zeigt an, ob eine Bibliothek geladen wurde
- [getenv](#) Zeigt den Wert einer Umgebungsvariablen an
- [get_cfg_var](#) Zeigt den Wert einer Option der PHP-Konfiguration
- [get_current_user](#) Den Besitzer des aktuellen PHP-Scripts anzeigen.
- [get_magic_quotes_gpc](#) Zeigt die aktuelle Konfiguration von magic quotes gpc
- [get_magic_quotes_runtime](#) Zeigt die aktuelle Konfiguration von magic_quotes_runtime
- [getlastmod](#) Zeigt die Uhrzeit der letzten Änderung einer Seite
- [getmyinode](#) Gibt den Inode des aktuellen Skripts aus
- [getmypid](#) Zeigt die ID des PHP-Prozesses
- [getmyuid](#) Zeigt die UID des Besitzers eines PHP-Scripts
- [getrusage](#) Zeigt den aktuellen Ressourcenverbrauch an
- [phpinfo](#) Zeigt viele Informationen zu PHP
- [php_logo_guid](#) Get the logo guid
- [php_sapi_name](#) Returns the type of interface between web server and PHP
- [phpversion](#) Zeigt die aktuell installierte PHP-Version
- [putenv](#) Setzt den Wert einer Umgebungsvariablen.
- [set_magic_quotes_runtime](#) Setzt magic_quotes_runtime
- [set_time_limit](#) Setzt die maximale Ausführungszeit
- [zend_logo_guid](#) Get the zend guid
- [get_loaded_extensions](#) Returns an array with the names of all modules compiled and loaded
- [get_extension_funcs](#) Returns an array with the names of the function of a module
- [get_required_files](#) Returns an array with the names of the files require_once()'d in a script
- [get_included_files](#) Returns an array with the names of the files include_once()'d in a script

error_log

(PHP3 , PHP4)

error_log -- Sendet eine Fehlermeldung

Beschreibung

```
int error_log (string message, int message_type [, string destination [, string extra_headers]])
```

Sendet eine Fehlermeldung an das Log-File des Webservers, einen TCP Port oder eine Datei. Der erste Parameter, *message*, ist die Fehlermeldung, die mitgeschrieben werden soll. Der zweite Parameter, *message_type* gibt an, wo die Meldung abgelegt werden soll:

Tabelle 1 Error_log() Typem

0	<i>Message</i> wird mit Hilfe des internen Log-Systems des Betriebssystems oder einer Datei geschrieben, je nachdem, wie error_log eingestellt ist.
1	<i>Message</i> wird via Email an die Adresse gesendet, die mit <i>destination</i> amngegeben wurde. Dies ist der einzige Punkt, bei dem der vierte Parameter <i>extra_headers</i> verwendet wird. Die Nachricht verwendet die selbe interne Funktion wie mail() .
2	<i>Message</i> wird über den PHP-Debugger gesendet. Diese Option ist nur verfügbar, wenn der Debugger gestartet wurde . Ist das der Fall, so wird mit dem <i>destination</i> Parameter der Hostname oder die IP-Adresse gesetzt. Optional auch der Port, unter dem der Debugger läuft.
3	<i>Message</i> wird an die Datei angefügt, die mit <i>destination</i> angegeben wurde.

Beispiel 1 Error_log() Beispiele

```
// Server sendet eine Nachricht an den Administrator,
// daß keine Verbindung zur Datenbank möglich ist.
if (!Ora_Logon($username, $password)) {
    error_log ("Oracle database not available!", 0);
}

// Server sendet eine Nachricht an den Administrator, daß F00 langsam ausgeht
if (!$foo = allocate_new_foo()) {
    error_log ("Big trouble, we're all out of F00s!", 1,
              "operator@mydomain.com");
}

// AndereMöglichkeiten, error_log() aufzurufen:
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");
```

[Zurück](#)

PHP Optionen und Informationen

[Anfang](#)

[Hoch](#)

[Vor](#)

error_reporting

error_reporting

(PHP3 , PHP4)

error_reporting -- Gibt an, wie PHP-Fehlermeldungen gezeigt werden

Beschreibung

int **error_reporting** ([int level])

Setzt die Stufe von PHP's Fehlermeldungen. Diese Stufe ist der Fehlermeldung hängt von den folgenden Werten ab. (Hinter den folgenden Links verbirgt sich die Bedeutung der internen Werte):

Tabelle 1 Error_reporting() Werte

Werte	interne Namen
1	E_ERROR
2	E_WARNING
4	E_PARSE
8	E_NOTICE
16	E_CORE_ERROR
32	E_CORE_WARNING

extension_loaded

(PHP3 >= 3.0.10, PHP4 >= 4.0b4)

extension_loaded -- Zeigt an, ob eine Bibliothek geladen wurde

Beschreibung

bool **extension_loaded** (string name)

Ist wahr, wenn die Bibliothek, die mit *name* definiert wurde, geladen ist. Einen Überblick über die verschiedenen Bibliotheken erhält mal mit [phpinfo\(\)](#).

Siehe auch [phpinfo\(\)](#).

Anmerkung: Diese Funktion wurde mit Version 3.0.10 eingeführt.

getenv

(PHP3 , PHP4)

getenv -- Zeigt den Wert einer Umgebungsvariablen an

Beschreibung

string **getenv** (string varname)

Zeigt den Wert der Umgebungsvariablen *varname* an oder false bei einem Fehler.

```
$ip = getenv ("REMOTE_ADDR"); // get the ip number of the user
```

Eine Liste aller Umgebungsvariablen wird mit [phpinfo\(\)](#) angezeigt. Ein Überblick über viele Umgebungsvariablen und deren Bedeutung findet sich unter den [CGI Spezifikationen](#), speziell auf der Seite der [Umgebungsvariablen](#).

get_cfg_var

(PHP3 , PHP4)

get_cfg_var -- Zeigt den Wert einer Option der PHP-Konfiguration

Description

string **get_cfg_var** (string varname)

Zeigt den derzeitigen Wert der Variablen der PHP-Konfiguration, die mit *varname* ausgewählt wurde oder false bei einem Fehler.

Bringt keine Informationen über die Konfiguration, mit der PHP kompiliert wurde, oder die aus der Apache-Konfiguration gelesen wird (mittels der php3_configuration_option Anweisungen).

Um herauszubekommen, ob ein [configuration file](#), benutzt wird, sollte versucht werden, den Wert der cfg_file_path Einstellung zu erhalten. Ist dieser Wert verfügbar, wird eine Konfigurationsdatei verwendet.

include()

Die [include\(\)](#)-Anweisung liest die angegebene Datei ein u

get_current_user

(PHP3 , PHP4)

get_current_user -- Den Besitzer des aktuellen PHP-Scripts anzeigen.

Beschreibung

string **get_current_user** (void)

Zeigt den Namen des Besitzers des aktuellen PHP-Scripts an.

Siehe auch [getmyuid\(\)](#), [getmypid\(\)](#), [getmyinode\(\)](#), und [getlastmod\(\)](#).

get_magic_quotes_gpc

(PHP3 >= 3.0.6, PHP4)

get_magic_quotes_gpc -- Zeigt die aktuelle Konfiguration von magic quotes gpc

Beschreibung

long **get_magic_quotes_gpc** (void)

Zeigt die aktuelle Konfiguration von [magic_quotes_gpc](#). (0 für aus, 1 für an).

Siehe auch [get_magic_quotes_runtime\(\)](#) und [set_magic_quotes_runtime\(\)](#).

get_magic_quotes_runtime

(PHP3 >= 3.0.6, PHP4)

get_magic_quotes_runtime -- Zeigt die aktuelle Konfiguration von magic_quotes_runtime

Beschreibung

Long **get_magic_quotes_runtime** (void)

Zeigt die aktuelle Konfiguration von [magic_quotes_runtime](#). (0 für aus, 1 für an).

Siehe auch [get_magic_quotes_gpc\(\)](#) und [set_magic_quotes_runtime\(\)](#).

getlastmod

(PHP3 , PHP4)

getlastmod -- Zeigt die Uhrzeit der letzten Änderung einer Seite

Beschreibung

int getlastmod (void)

Zeigt die Uhrzeit der letzten Änderung der aktuellen Seite. Der Wert wird als Unix-Timestamp angezeigt, welcher für [date\(\)](#) benutzt werden kann. Bringt false bei einem Fehler.

Beispiel 1 getlastmod() Beispiel

```
// outputs e.g. 'Last modified: March 04 1998 20:43:59.'  
echo "Last modified: ".date ("F d Y H:i:s.", getlastmod());
```

Siehe auch [date\(\)](#), [getmyuid\(\)](#), [get_current_user\(\)](#), [getmyinode\(\)](#) und [getmypid\(\)](#).

getmyinode

(PHP3 , PHP4)

getmyinode -- Gibt den Inode des aktuellen Skripts aus

Beispiel

```
int getmyinode (void)
```

Zeigt den inode des aktuellen Scripts oder false bei einem Fehler.

Siehe auch [getmyuid\(\)](#), [get_current_user\(\)](#), [getmypid\(\)](#) und [getlastmod\(\)](#).

getmypid

(PHP3 , PHP4)

getmypid -- Zeigt die ID des PHP-Prozesses

Beschreibung

int getmypid (void)

Zeigt die aktuelle ID des PHP-Prozesses oder false bei einem Fehler.

Achtung: Wenn PHP als Server-Modul läuft, ist es nicht garantiert, dass Scripts unter verschiedenen PIDs laufen.

Siehe auch [getmyuid\(\)](#), [get_current_user\(\)](#), [getmyinode\(\)](#) und [getlastmod\(\)](#).

getmyuid

(PHP3 , PHP4)

getmyuid -- Zeigt die UID des Besitzers eines PHP-Scripts

Beschreibung

int getmyuid (void)

Zeigt die User-ID des aktuellen Scripts oder false bei einem Fehler.

Siehe auch [getmypid\(\)](#), [get_current_user\(\)](#), [getmyinode\(\)](#) und [getlastmod\(\)](#).

getrusage

(PHP3 >= 3.0.7, PHP4 >= 4.0b2)

getrusage -- Zeigt den aktuellen Ressourcenverbrauch an

Beschreibung

array **getrusage** ([int who])

Dies ist eine Schnittstelle zu getrusage(2). Stellt ein assoziatives Array mit den Daten zur Verfügung, die der Systemaufruf ausgibt. Wenn who 1 ist, wird getrusage mit RUSAGE_CHILDREN aufgerufen.

Alle Einträge können über ihre Dokumentenfeldernamen aufgerufen werden.

Beispiel 1 Getrusage() Beispiel

```
$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)
```

Siehe auch im Handbuch des Betriebssystems für weitere Details.

phpinfo

(PHP3 , PHP4)

phpinfo -- Zeigt viele Informationen zu PHP

Beschreibung

int phpinfo (void)

Zeigt eine grosse Anzahl von Informationen über die aktuelle Konfiguration von PHP an. Unter anderem die Optionen während der Kompilierens und die Erweiterungen, die PHP Version, Informationen über den Server, die Umgebung (wenn PHP als Modul kompiliert wurde), die PHP Umgebung, Version und Informationen zum Betriebssystem, Pfade, Haupt- and lokale Werte der Konfigurationsoptionen, HTTP Header, und die GNU Public License.

Siehe auch [phpversion\(\)](#).

php_logo_guid

(PHP4 >= 4.0b4)

php_logo_guid -- Get the logo guid

Description

string **php_logo_guid** (void)

Anmerkung: This functionality was added in PHP4 Beta 4.

php_sapi_name

(PHP4 >= 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.

Beispiel 1 Php_sapi_name() Example

```
$inter_type = php_sapi_name();
if ($inter_type == "cgi")
    print "You are using CGI PHP\n";
else
    print "You are not using CGI PHP\n";
```

phpversion

(PHP3 , PHP4)

phpversion -- Zeigt die aktuell installierte PHP-Version

Beschreibung

string **phpversion** (void)

Zeigt die Version des installierten PHP-Parsers.

Beispiel 1 Phpversion() Beispiel

```
// Gibt z.B. 'Current PHP version: 3.0rel-dev' aus
echo "Current PHP version: ".phpversion();
```

Siehe auch [phpinfo\(\)](#).

putenv

(PHP3 , PHP4)

putenv -- Setzt den Wert einer Umgebungsvariablen.

Beschreibung

void **putenv** (string setting)

Setzt *setting* als Umgebungsvariable.

Beispiel 1 Setzen einer Umgebungsvariablen

```
putenv ("UNIQID=$uniqid");
```

set_magic_quotes_runtime

(PHP3 >= 3.0.6, PHP4)

set_magic_quotes_runtime -- Setzt magic_quotes_runtime

Beschreibung

long **set_magic_quotes_runtime** (int new_setting)

Setzt [magic_quotes_runtime](#). (0 für aus, 1 für an).

Siehe auch [get_magic_quotes_gpc\(\)](#) und [get_magic_quotes_runtime\(\)](#).

set_time_limit

(PHP3 , PHP4)

set_time_limit -- Setzt die maximale Ausführungszeit

Beschreibung

`void set_time_limit (int seconds)`

Legt die Zeit in Sekunden fest, die ein Script laufen darf. Ist diese Zeit abgelaufen, wird ein Fehler zurückgegeben. Der Standart liegt bei 30 Sekunden, es sei denn, max_execution_time ist [configuration file](#) eingetragen. Bei einer Einstellung von 0 Sekunden ist die maximale Ausführungszeit nicht begrenzt.

Wenn **set_time_limit()** aufgerufen wird, dann startet der Zähler neu. Das heisst, wenn die Standarteinstellung s 30 Sekunden beträgt und auf 25 Sekunden durch ein Script geändert werden (wie z.B. durch set_time_limit(25), darf das Script 45 Sekunden laufen, bevor eine Fehlermeldung ausgegeben wird.

Achtung: Wenn PHP im Safemode läuft, hat **set_time_limit()** keinen Effekt. Die einzige Möglichkeit, dies zu umgehen, ist den Safemode im [configuration file](#) auszuschalten.

zend_logo_guid

(PHP4 >= 4.0b4)

zend_logo_guid -- Get the zend guid

Description

string **zend_logo_guid** (void)

Anmerkung: This functionality was added in PHP4 Beta 4.

get_loaded_extensions

(PHP4 >= 4.0b4)

`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

```
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\(\)](#)

get_extension_funcs

(PHP4 >= 4.0b4)

`get_extension_funcs` -- Returns an array with the names of the function 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_required_files

(PHP4 >= 4.0RC2)

`get_required_files` -- Returns an array with the names of the files `require_once()`'d in a script

Description

array **get_required_files** (void)

This function returns an associative array of the names of all the files that have been loaded into a script using [require_once\(\)](#). The indexes of the array are the file names as used in the [require_once\(\)](#) without the ".php" extension.

The example below

Beispiel 1 Printing the required and included files

```
<?php

require_once("local.php");
require_once("../inc/global.php");

for ($i=1; $i<5; $i++)
    include "util".$i.".php";

echo "Required_once files\n";
print_r(get_required_files());

echo "Included_once files\n";
print_r(get_included_files());
```

will generate the following output:

Required_once files

Array

```
(
    [local] => local.php
    [../inc/global] => /full/path/to/inc/global.php
)
```

Included_once files

Array

```
(
    [util1] => util1.php
    [util2] => util2.php
    [util3] => util3.php
    [util4] => util4.php
)
```

Anmerkung: As of PHP 4.0.1pl2 this function assumes that the `required_once` files end in the extension ".php", other extensions do not work.

See also: [require_once\(\)](#), [include_once\(\)](#), [get_included_files\(\)](#)

[Zurück](#)[get_extension_funcs](#)[Anfang](#)[Hoch](#)[Vor](#)[get_included_files](#)

get_included_files

(PHP4 >= 4.0RC1)

`get_included_files` -- Returns an array with the names of the files `include_once()`'d in a script

Description

array **get_included_files** (void)

This function returns an associative array of the names of all the files that have been loaded into a script using [include_once\(\)](#). The indexes of the array are the file names as used in the [include_once\(\)](#) without the ".php" extension.

Anmerkung: As of PHP 4.0.1pl2 this function assumes that the `include_once` files end in the extension ".php", other extensions do not work.

See also: [require_once\(\)](#), [include_once\(\)](#), [get_required_files\(\)](#)

XLVI. POSIX functions

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 PHP3 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.

Inhaltsverzeichnis

- [posix_kill](#) Send a signal to a process
- [posix_getpid](#) Return the current process identifier
- [posix_getppid](#) Return the parent process identifier
- [posix_getuid](#) Return the real user ID of the current process
- [posix_geteuid](#) Return the effective user ID of the current process
- [posix_getgid](#) Return the real group ID of the current process
- [posix_getegid](#) Return the effective group ID of the current process
- [posix_setuid](#) Set the effective UID of the current process
- [posix_setgid](#) Set the effective GID of the current process
- [posix_getgroups](#) Return the group set of the current process
- [posix_getlogin](#) Return login name
- [posix_getpgrp](#) Return the current process group identifier
- [posix_setsid](#) Make the current process a session leader
- [posix_setpgid](#) set process group id for job control
- [posix_getpgid](#) Get process group id for job control
- [posix_getsid](#) Get the current sid of the process
- [posix_uname](#) Get system name
- [posix_times](#) Get process times
- [posix_ctermid](#) Get path name of controlling terminal
- [posix_ttyname](#) Determine terminal device name
- [posix_isatty](#) Determine if a file descriptor is an interactive terminal
- [posix_getcwd](#) Pathname of current directory
- [posix_mkfifo](#) Create a fifo special file (a named pipe)

[posix_getgrnam](#) Return info about a group by name
[posix_getgrgid](#) Return info about a group by group id
[posix_getpwnam](#) Return info about a user by username
[posix_getpwuid](#) Return info about a user by user id
[posix_getrlimit](#) Return info about system ressource limits

[Zurück](#)

get_included_files

[Anfang](#)

[Hoch](#)

[Vor](#)

posix_kill

posix_kill

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

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_getpid

(PHP3 >= 3.0.10, PHP4 >= 4.0b4)

posix_getpid -- Return the current process identifier

Description

```
int posix_getpid (void )
```

Return the process identifier of the current process.

[Zurück](#)[Anfang](#)[Vor](#)

posix_kill

[Hoch](#)

posix_getppid

posix_getppid

(PHP3 >= 3.0.10, PHP4 >= 4.0b4)

posix_getppid -- Return the parent process identifier

Description

```
int posix_getppid (void )
```

Return the process identifier of the parent process of the current process.

[Zurück](#)[Anfang](#)[Vor](#)

posix_getpid

[Hoch](#)

posix_getuid

posix_getuid

(PHP3 >= 3.0.10, PHP4 >= 4.0b4)

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_geteuid

(PHP3 >= 3.0.10, PHP4 >= 4.0b4)

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

(PHP3 >= 3.0.10, PHP4 >= 4.0b4)

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_getegid

(PHP3 >= 3.0.10, PHP4 >= 4.0b4)

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_setuid

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

posix_setuid -- Set the effective 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_setgid

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

posix_setgid -- Set the effective 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_getgroups

(PHP3 >= 3.0.10, PHP4 >= 4.0b4)

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.

[Zurück](#)[Anfang](#)[Vor](#)

posix_setgid

[Hoch](#)

posix_getlogin

posix_getlogin

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

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_getpgrp

(PHP3 >= 3.0.10, PHP4 >= 4.0b4)

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.

[Zurück](#)[Anfang](#)[Vor](#)

posix_getlogin

[Hoch](#)

posix_setsid

posix_setsid

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

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_setpgid

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

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_getpgid

(PHP3 >= 3.0.10, PHP4 >= 4.0b4)

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_getsid

(PHP3 >= 3.0.10, PHP4 >= 4.0b4)

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_uname

(PHP3 >= 3.0.10, PHP4 >= 4.0b4)

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)

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.

posix_times

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

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_ctermid

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

posix_ctermid -- Get path name of controlling terminal

Description

string **posix_ctermid** (void)

Needs to be written.

[Zurück](#)[Anfang](#)[Vor](#)[posix_times](#)[Hoch](#)[posix_ttyname](#)

posix_ttyname

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

posix_ttyname -- Determine terminal device name

Description

string **posix_ttyname** (int fd)

Needs to be written.

[Zurück](#)[Anfang](#)[Vor](#)

posix_ctermid

[Hoch](#)

posix_isatty

posix_isatty

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

posix_isatty -- Determine if a file descriptor is an interactive terminal

Description

```
bool posix_isatty (int fd)
```

Needs to be written.

[Zurück](#)[Anfang](#)[Vor](#)

posix_ttyname

[Hoch](#)

posix_getcwd

posix_getcwd

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

posix_getcwd -- Pathname of current directory

Description

string **posix_getcwd** (void)

Needs to be written ASAP.

[Zurück](#)[Anfang](#)[Vor](#)

posix_isatty

[Hoch](#)

posix_mkfifo

posix_mkfifo

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

posix_mkfifo -- Create a fifo special file (a named pipe)

Description

bool **posix_getcwd** (string pathname, int mode)

Needs to be written ASAP..

[Zurück](#)[Anfang](#)[Vor](#)

posix_getcwd

[Hoch](#)

posix_getgrnam

posix_getgrnam

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

posix_getgrnam -- Return info about a group by name

Description

array **posix_getgrnam** (string name)

Needs to be written.

[Zurück](#)[Anfang](#)[Vor](#)

posix_mkfifo

[Hoch](#)

posix_getgrgid

posix_getgrgid

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

posix_getgrgid -- Return info about a group by group id

Description

array **posix_getgrgid** (int gid)

Needs to be written.

posix_getpwnam

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

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:

Tabelle 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 <i>username</i> 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

(PHP3 >= 3.0.13, PHP4 >= 4.0b4)

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:

Tabelle 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 <i>uid</i> 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

(PHP3 >= 3.0.10, PHP4 >= 4.0b4)

posix_getrlimit -- Return info about system ressource limits

Description

array **posix_getrlimit** (void)

Needs to be written ASAP.

[Zurück](#)[Anfang](#)[Vor](#)

posix_getpwuid

[Hoch](#)

PostgreSQL Funktionen

XLVII. PostgreSQL Funktionen

Postgres, ursprünglich entwickelt im UC Berkeley Computer Science Department, hat Pionierarbeit bei objektrelationalen Datenbankkonzepten geleistet, die jetzt Einzug in kommerzielle Datenbanken halten. Postgres bietet SQL92/SQL3 Sprachunterstützung, Transaktionen und erweiterbare Typenklassen. PostgreSQL ist eine Public Domain und Open Source Weiterentwicklung des ursprünglichen Berkeley-Codes.

PostgreSQL ist kostenlos. Die aktuelle Version ist erhältlich bei www.PostgreSQL.org.

Seit Version 6.3 (03/02/1998) benutzt PostgreSQL Unix Domain Sockets, siehe folgende Tabelle. Der Socket ist in `/tmp/.s.PGSQL.5432` zu finden. Der Schalter `-i`, der dem **postmaster** mitgegeben werden kann, weist diesen an, sowohl über TCP/IP als auch über UNIX Domain Sockets eine Verbindung bereitzustellen.

Tabelle 1 Postmaster und PHP

Postmaster	PHP	Status
postmaster &	<code>pg_connect("", "", "", "", "dbname");</code>	OK
postmaster -i &	<code>pg_connect("", "", "", "", "dbname");</code>	OK
postmaster &	<code>pg_connect("localhost", "", "", "", "dbname");</code>	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.php3 on line 20.
postmaster -i &	<code>pg_connect("localhost", "", "", "", "dbname");</code>	OK

Eine Verbindung läßt sich auch mit folgendem Befehl herstellen: `$conn = pg_Connect("host=localhost port=5432 dbname=chris");`

Um die Large Object-Schnittstelle zu benutzen, ist es nötig, diese in einem Transaktionsblock einzuschließen. Ein Transaktionsblock beginnt mit einem **begin** und endet, wenn die Transaktion gültig war, mit **commit** oder **end**. Wenn die Transaktion fehlschlägt, sollte sie mit **abort** oder **rollback** geschlossen werden.

Beispiel 1 Grosse Objekte benutzen

```
<?php
    $database = pg_Connect ("", "", "", "", "jacarta");
    pg_exec ($database, "begin");
    $oid = pg_locreate ($database);
    echo ("$oid\n");
    $handle = pg_loopen ($database, $oid, "w");
    echo ("$handle\n");
    pg_lowrite ($handle, "gaga");
    pg_loclose ($handle);
    pg_exec ($database, "commit")
?>
```

Inhaltsverzeichnis

- [pg_Close](#) Schliesst eine PostgreSQL-Verbindung
- [pg_cmdTuples](#) Gibt die Anzahl betroffener Tupel zurück
- [pg_Connect](#) Öffnet eine Verbindung
- [pg_DBname](#) Name der Datenbank
- [pg_ErrorMessage](#) Fehlermeldung
- [pg_Exec](#) Führt eine Abfrage aus
- [pg_Fetch_Array](#) Holt eine Datenbankreihe als Array
- [pg_Fetch_Object](#) Holt einen Datensatz als Objekt
- [pg_Fetch_Row](#) Holt einen Datensatz als numerisches Array
- [pg_FieldIsNull](#) Prüft, ob ein Feld NULL ist
- [pg_FieldName](#) Gibt den Namen eines Feldes zurück
- [pg_FieldNum](#) Liefert die Feldnummer eines Feldes
- [pg_FieldPrtLen](#) Liefert die angegebene Feldlänge
- [pg_FieldSize](#) Liefert die interne Speichergröße des benannten Felds
- [pg_FieldType](#) Liefert den Feldtyp der entsprechenden Feldnummer
- [pg_FreeResult](#) Gibt durch Ergebnisse belegten Speicher frei
- [pg_GetLastOid](#) Gibt die letzte Objektkennung aus
- [pg_Host](#) Gibt den Hostnamen zurück
- [pg_loclose](#) Schliesst ein großes Objekt
- [pg_locreate](#) Erstellt ein großes Objekt
- [pg_loexport](#) export a large object to file

[pg_loimport](#) import a large object from file
[pg_loopen](#) Öffnet ein grosses Objekt
[pg_loread](#) Liest ein grosses Objekt
[pg_loreadall](#) Liest ein grosses Objekt vollständig
[pg_lounlink](#) Ein grosses Objekt löschen
[pg_lowrite](#) Schreibt in ein grosses Objekt
[pg_NumFields](#) Gibt die Anzahl der Felder aus
[pg_NumRows](#) Gibt die Anzahl der Zeilen aus
[pg_Options](#) Liefert Verbindungsoptionen
[pg_pConnect](#) Stellt eine persistente Datenbankverbindung her
[pg_Port](#) Gibt die Portnummer aus
[pg_Result](#) Liefert Werte eines bezeichneten Ergebnisses
[pg_trace](#) Trace a connection to PostgreSQL server
[pg_tty](#) Gibt den tty-Namen aus
[pg_untrace](#) Stop tracing a connection to PostgreSQL server

[Zurück](#)

[Anfang](#)

[Vor](#)

posix_getrlimit

[Hoch](#)

pg_Close

pg_Close

(unknown)

pg_Close -- Schliesst eine PostgreSQL-Verbindung

Beschreibung

bool **pg_close** (int connection)

Gibt false zurück, wenn die Verbindung zur Datenbank keine gültige Kennung hat, ansonsten ist der Rückgabewert true. Schliesst eine PostgreSQL Datenbank-Verbindung mit der angegebenen Verbindungskennung.

pg_cmdTuples

(unknown)

pg_cmdTuples -- Gibt die Anzahl betroffener Tupel zurück

Beschreibung

int pg_cmdtuples (int result_id)

pg_cmdTuples() gibt die Anzahl der Tupel (instances) einer INSERT, UPDATE oder DELETE Abfrage zurück. Die Funktion liefert 0 zurück, wenn die Abfrage keinen Tupel liefert.

Beispiel 1 pg_cmdtuples

```
<?php
$result = pg_exec($conn, "INSERT INTO verlag VALUES ('Autor')");
$cmdtuples = pg_cmdtuples($result);
echo $cmdtuples . " <- cmdtuples affected.";
?>
```

pg_Connect

(unknown)

pg_Connect -- Öffnet eine Verbindung

Beschreibung

int pg_connect (string host, string port, string options, string tty, string dbname)

Öffnet eine Verbindung zu einer PostgreSQL-Datenbank und liefert bei Erfolg die Verbindungskennung zurück - oder false, falls keine Verbindung hergestellt werden konnte. Jedes der Argumente, inklusive der Portnummer, sollte ein in Anführungszeichen gesetzter String sein. Die options- und tty-Argumente sind optional und können ausgelassen werden. Die von dieser Funktion zurückgelieferte Verbindungskennung wird von anderen PostgreSQL-Funktionen benötigt. Es ist auch möglich, mehrere Verbindungen gleichzeitig zu öffnen.

Eine Verbindung kann auch mit folgendem Befehl hergestellt werden: **\$conn = pg_connect("dbname=marliese port=5432");** Weitere Parameter neben *dbname* und *port* sind *host*, *tty*, *options*, *user* und *password*.

Siehe auch **pg_pConnect()**.

pg_DBname

(unknown)

pg_DBname -- Name der Datenbank

Beschreibung

string **pg_dbname** (int connection)

Gibt den Namen der Datenbank, deren Verbindungskennung übergeben wurde, aus - oder false, wenn es sich um eine ungültige Kennung handelt.

pg_ErrorMessage

(unknown)

pg_ErrorMessage -- Fehlermeldung

Beschreibung

string **pg_errormessage** (int connection)

Liefert einen String, der die Fehlermeldung enthält oder false bei Versagen. Details über den Fehler können aber eher nicht durch pg_errormessage bezogen werden. Wenn der Fehler bei der letzten Datenbank-Aktion, für die eine gültige Kennung existierte, auftrat, liefert die Funktion einen String mit der Fehlermeldung des Backend-Servers aus.

pg_Exec

(unknown)

pg_Exec -- Führt eine Abfrage aus

Beschreibung

`int pg_exec (int connection, string query)`

Gibt eine Ergebniskennung zurück, wenn die Abfrage ausgeführt wurde oder false bei gescheiterter Abfrage oder ungültiger Verbindungskennung. (Details über den Fehler kann man bei gültiger Verbindung durch den Gebrauch von **pg_ErrorMessage()** erhalten). Diese Funktion sendet ein SQL-Statement an die durch die Verbindungskennung spezifizierte PostgreSQL Datenbank. Die Verbindung muß eine gültige, durch **pg_Connect()** zurückgegebene Kennung haben. Der Rückgabewert dieser Funktion ist eine Kennung, die dazu verwendet kann, das Abfrageergebnis mit anderen PostgreSQL-Funktionen anzusprechen.

Anmerkung: PHP/FI lieferte als Rückgabewert 1, wenn die Abfrage kein Ergebnis liefern würde (beispielsweise INSERTs oder UPDATEs), und sogar einen Wert größer 1 bei SELECTs, die nichts zurückgaben. Dies trifft für PHP nicht mehr zu.

pg_Fetch_Array

(unknown)

pg_Fetch_Array -- Holt eine Datenbankreihe als Array

Beschreibung

array **pg_fetch_array** (int result, int row [, int result_type])

Rückgabewert: Ein Array, das mit der eingelesenen Reihe identisch ist, oder false, wenn keine Reihe mehr gelesen werden kann.

pg_fetch_array() ist eine erweiterte Version von [pg_fetch_row\(\)](#). Zusätzlich zur Speicherung der Daten in einem aufgezählten Array speichert diese Funktion die Daten in assoziativen Indexen, indem es den Feldnamen als Schlüssel verwendet.

Das dritte optionale Argument *result_type* in **pg_fetch_array()** ist eine Konstante, die folgende Werte annehmen kann: PGSQL_ASSOC, PGSQL_NUM und PGSQL_BOTH.

Anmerkung: *Result_type* wurde in PHP 4.0 hinzugefügt.

Wichtig zu bemerken ist, daß **pg_fetch_array()** nicht wesentlich langsamer als [pg_fetch_row\(\)](#) ist, obwohl eine umfassendere Funktion zur Verfügung gestellt wird.

Für weitere Einzelheiten siehe auch [pg_fetch_row\(\)](#)

Beispiel 1 PostgreSQL fetch array

```
<?php
$conn = pg_pconnect("", "", "", "", "verlag");
if (!$conn) {
    echo "Es ist ein Fehler aufgetreten.\n";
    exit;
}

$result = pg_Exec ($conn, "SELECT * FROM autoren");
if (!$result) {
    echo "Es ist ein Fehler aufgetreten.\n";
    exit;
}
```

```
$arr = pg_fetch_array ($result, 0);  
echo $arr[0] . " <- array\n";  
$arr = pg_fetch_array ($result, 1);  
echo $arr["autor"] . " <- array\n";  
?>
```

[Zurück](#)

pg_Exec

[Anfang](#)

[Hoch](#)

[Vor](#)

pg_Fetch_Object

pg_Fetch_Object

(unknown)

pg_Fetch_Object -- Holt einen Datensatz als Objekt

Beschreibung

object **pg_fetch_object** (int result, int row [, int result_type])

Rückgabewert: Ein Objekt, dessen Eigenschaften mit denen der Felder des erhaltenen Datensatzes übereinstimmen oder false, wenn kein Datensatz mehr zurückgegeben wird.

pg_fetch_object() ist [pg_fetch_array\(\)](#) ähnlich, mit einem Unterschied: Statt eines Arrays wird ein Objekt zurückgegeben. Indirekt heißt das, dass man die Daten nur über die Feldnamen und nicht über ihre Offsets ansprechen kann (Zahlen sind nicht erlaubte Eigenschaftsnamen).

Das dritte, optionale Argument *result_type* in **pg_fetch_object()** ist eine Konstante und kann die folgenden Werte annehmen: PGSQL_ASSOC, PGSQL_NUM, and PGSQL_BOTH.

Anmerkung: *Result_type* wurde in PHP 4.0 hinzugefügt

Von der Geschwindigkeit her ist die Funktion mit [pg_fetch_array\(\)](#) identisch und fast so schnell wie [pg_fetch_row\(\)](#) (Der Unterschied ist unwesentlich).

Siehe auch: [pg_fetch_array\(\)](#) und [pg_fetch_row\(\)](#).

Beispiel 1 Postgres fetch object

```
<?php
$database = "verlag";
$db_conn = pg_connect ("localhost", "5432", "", "", $database);
if (!$db_conn): ?>
    <H3>Konnte keine Datenbankverbindung zu
    <? echo $database ?> herstellen.</H3> <?
    exit;
endif;

$qu = pg_exec ($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++;
    endwhile; ?>

```

```

<PRE><?php
$fields[] = Array ("autor", "Autor");
$fields[] = Array ("jahr", " Jahr");
$fields[] = Array ("titel", " Titel");
$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;
    $row++;
endwhile;
echo "-----\n"; ?>
</PRE> <?php
pg_freeResult ($qu);
pg_close ($db_conn);
?>

```

[Zurück](#)
[pg_Fetch_Array](#)
[Anfang](#)
[Hoch](#)
[Vor](#)
[pg_Fetch_Row](#)

pg_Fetch_Row

(unknown)

pg_Fetch_Row -- Holt einen Datensatz als numerisches Array

Beschreibung

array **pg_fetch_row** (int result, int row)

Rückgabewert: Ein Array, das mit dem gelesenen Datensatz übereinstimmt oder false, wenn kein Datensatz mehr zurückgegeben wird.

pg_fetch_row() holt eine Datensatz aus dem durch die Kennung festgelegten Ergebnis. Der Datensatz wird als numerisches Array zurückgegeben. Jedes Ergebnisfeld wird in einem Array-Offset gespeichert, angefangen bei Offset 0.

Ein nachfolgender Aufruf von **pg_fetch_row()** wird den nächste Datensatz im Ergebnissatz ausgeben, oder false, wenn keine auszugebender Datensatz mehr vorhanden ist.

Siehe auch: [pg_fetch_array\(\)](#), [pg_fetch_object\(\)](#), [pg_result\(\)](#).

Beispiel 1 Postgres fetch row

```
<?php
$conn = pg_pconnect("", "", "", "", "verlag");
if (!$conn) {
    echo "Es ist ein Fehler aufgetreten.\n";
    exit;
}

$result = pg_Exec ($conn, "SELECT * FROM autoren");
if (!$result) {
    echo "Es ist ein Fehler aufgetreten.\n";
    exit;
}

$row = pg_fetch_row ($result, 0);
echo $row[0] . " <- row\n";

$row = pg_fetch_row ($result, 1);
echo $row[0] . " <- row\n";
```

```
$row = pg_fetch_row ($result, 2);  
echo $row[1] . " <- row\n";  
?>
```

[Zurück](#)

pg_Fetch_Object

[Anfang](#)

[Hoch](#)

[Vor](#)

pg_FieldIsNull

pg_FieldIsNull

(unknown)

pg_FieldIsNull -- Prüft, ob ein Feld NULL ist

Beschreibung

`int pg_fieldisnull (int result_id, int row, mixed field)`

Prüft, ob ein Feld NULL ist oder nicht. Die Funktion liefert 0 zurück, wenn das Feld in der übergebenen Reihe nicht NULL ist und 1, wenn das Feld NULL ist. Das zu prüfende Feld kann als Nummer oder Feldname angegeben werden. Die Numerierung der Felder eines Datensatzes beginnt bei 0.

[Zurück](#)[Anfang](#)[Vor](#)[pg_Fetch_Row](#)[Hoch](#)[pg_FieldName](#)

pg_FieldName

(unknown)

pg_FieldName -- Gibt den Namen eines Feldes zurück

Beschreibung

string **pg_fieldname** (int result_id, int field_number)

pg_FieldName() gibt den Feldnamen des Feldes, das zu der übergebenen Feldnummer gehört, aus dem durch die Kennung zu definierenden Ergebnis aus. Die Feldnummerierung beginnt bei 0.

pg_FieldNum

(unknown)

pg_FieldNum -- Liefert die Feldnummer eines Feldes

Beschreibung

```
int pg_fieldnum (int result_id, string field_name)
```

pg_FieldNum() gibt die Nummer der Spalte aus, die zu dem übergebenen Feldnamen in dem durch die Kennung definierten Ergebnis gehört. Die Feldnumerierung beginnt bei 0. Diese Funktion gibt bei einem Fehler -1 aus.

pg_FieldPrtLen

(unknown)

pg_FieldPrtLen -- Liefert die angegebene Feldlänge

Beschreibung

int pg_fieldprtlen (int result_id, int row_number, string field_name)

pg_FieldPrtLen() gibt die aktuelle Länge (Anzahl der Zeichen) eines bestimmten Wertes in einem Postgres SQL-Ergebnis aus. Die Feldnumerierung beginnt bei 0. Die Funktion liefert bei einem Fehler -1 zurück.

pg_FieldSize

(unknown)

pg_FieldSize -- Liefert die interne Speichergröße des benannten Felds

Beschreibung

`int pg_fieldsize (int result_id, int field_number)`

pg_FieldSize() liefert die interne Speichergröße (in Bytes) des angegebenen Feldes in dem durch die Kennung definierten Ergebnis. Die Feldnumerierung beginnt bei 0. Eine Feldgröße von -1 deutet auf ein Feld mit variabler Länge hin. Diese Funktion gibt im Fehlerfall false zurück.

[Zurück](#)[Anfang](#)[Vor](#)

pg_FieldPrtLen

[Hoch](#)

pg_FieldType

pg_FieldType

(unknown)

pg_FieldType -- Liefert den Feldtyp der entsprechenden Feldnummer

Beschreibung

```
int pg_fieldtype (int result_id, int field_number)
```

pg_FieldType() liefert einen String, der den Feldtyp des bezeichneten Feldes aus einem definierten Ergebnis enthält. Die Feldnumerierung beginnt bei 0.

[Zurück](#)

pg_FieldSize

[Anfang](#)[Hoch](#)[Vor](#)

pg_FreeResult

pg_FreeResult

(unknown)

pg_FreeResult -- Gibt durch Ergebnisse belegten Speicher frei

Beschreibung

```
int pg_freeresult (int result_id)
```

pg_FreeResult() braucht nur aufgerufen werden, wenn man befürchten muss, dass während des Skriptablaufes zu viel Speicher verbraucht wird. Der Speicher, den Ergebnisse belegen, wird automatisch nach Beendigung des Skriptes freigegeben. Wer allerdings sicher ist, die Ergebnisdaten innerhalb des Skriptes nicht mehr zu benötigen, kann **pg_FreeResult()** mit der Ergebniskennung als Argument aufrufen, und der entsprechende Speicher wird freigegeben.

pg_GetLastOid

(unknown)

pg_GetLastOid -- Gibt die letzte Objektkennung aus

Beschreibung

```
int pg_getlastoid (int result_id)
```

pg_GetLastOid() kann verwendet werden, um die einem eingefügten Tupel zugeordnete OID zu erhalten, wenn die Ergebniskennung vom letzten mit **pg_Exec()** ausgeführten Befehls benutzt wird und dieser ein SQL INSERT war. Die Funktion gibt ein positives Integer zurück, wenn eine gültige OID existiert. -1 wird zurückgegeben, wenn ein Fehler aufgetreten ist oder der letzte mit **pg_Exec()** abgesetzte Befehl kein INSERT war.

pg_Host

(unknown)

pg_Host -- Gibt den Hostnamen zurück

Beschreibung

string **pg_host** (int connection_id)

pg_Host() liefert den Hostnamen des Datenbankservers, der durch die übergebene Kennung verbunden ist.

pg_loclose

(PHP3 , PHP4)

pg_loclose -- Schliesst ein großes Objekt

Beschreibung

void **pg_loclose** (int fd)

pg_loclose() schliesst ein Inversion Large Object. *fd* ist der Dateibeschreiber des mit [pg_loopopen\(\)](#) geöffneten Objekts.

pg_locreate

(PHP3 , PHP4)

pg_locreate -- Erstellt ein großes Objekt

Beschreibung

`int pg_locreate (int conn)`

pg_locreate() erstellt ein Inversion Large Object und liefert dessen OID zurück. *conn* bezeichnet eine gültige Datenbankverbindung. Die PostgreSQL Zugriffsmöglichkeiten INV_READ, INV_WRITE und INV_ARCHIVE werden nicht unterstützt, das Objekt wird stets mit Lese- und Schreibzugriff erstellt. INV_ARCHIVE wird seit Version 6.3 und höher von PostgreSQL selbst nicht mehr unterstützt.

[Zurück](#)[Anfang](#)[Vor](#)

pg_loclose

[Hoch](#)

pg_loexport

pg_loexport

(PHP4 >= 4.0.1)

pg_loexport -- export a large object to file

Description

bool **pg_loexport** (int oid, int file [, int connection_id])

The *oid* argument specifies the object id of the large object to export and the *filename* argument specifies the pathname of the file. Returns FALSE if an error occurred, TRUE otherwise. Remember that handling large objects in PostgreSQL must happen inside a transaction.

pg_loimport

(PHP4 >= 4.0.1)

pg_loimport -- import a large object from file

Description

```
int pg_loimport (int file [, int connection_id])
```

The *filename* argument specifies the pathname of the file to be imported as a large object. Returns FALSE if an error occurred, object id of the just created large object otherwise. Remember that handling large objects in PostgreSQL must happen inside a transaction.

pg_loopen

(PHP3 , PHP4)

pg_loopen -- Öffnet ein grosses Objekt

Beschreibung

`int pg_loopen (int conn, int objoid, string mode)`

pg_loopen() öffnet ein Inversion Large Object und liefert einen Dateibesreiber des großen Objektes. Dieser beinhaltet Informationen über die Verbindung. Der Dateibesreiber des großen Objektes sollte daher geschlossen werden, bevor die Verbindung beendet wird. *objoid* definiert die gültige OID eines grossen Objektes und *mode* kann "r", "w" oder "rw" sein.

pg_loread

(PHP3 , PHP4)

pg_loread -- Liest ein grosses Objekt

Beschreibung

string **pg_loread** (int *fd*, int *len*)

pg_loread() liest höchstens *len* Bytes eines grossen Objektes und gibt dies als String zurück. *fd* legt einen gültigen Dateibesreiber fest und *len* definiert die maximal erlaubte Grösse des Segmentes des großen Objektes.

pg_loreadall

(PHP3 , PHP4)

pg_loreadall -- Liest ein grosses Objekt vollständig

Beschreibung

```
void pg_loreadall (int fd)
```

pg_loreadall() liest ein grosses Objekt und reicht es sofort nach dem Senden der entsprechenden Header an den Browser weiter. Wird hauptsächlich für Binärdaten wie Bilder oder Sound genutzt.

pg_lounlink

(PHP3 , PHP4)

pg_lounlink -- Ein grosses Objekt löschen

Beschreibung

void **pg_lounlink** (int conn, int lobjid)

pg_lounlink() löscht das durch den *lobjid* Dateibesreiber festgelegte grosse Objekt.

[Zurück](#)

pg_loreadall

[Anfang](#)[Hoch](#)[Vor](#)

pg_lowrite

pg_lowrite

(PHP3 , PHP4)

pg_lowrite -- Schreibt in ein grosses Objekt

Beschreibung

int **pg_lowrite** (int *fd*, string *buf*)

pg_lowrite() schreibt aus einer Variablen höchstens *buf* in ein grosses Objekt und liefert die Anzahl der tatsächlich geschriebenen Bytes zurück, oder false, wenn ein Fehler auftritt. *fd* ist der Dateibesreiber des grossen Objektes aus [pg_loopopen\(\)](#).

pg_NumFields

(unknown)

pg_NumFields -- Gibt die Anzahl der Felder aus

Beschreibung

```
int pg_numfields (int result_id)
```

pg_NumFields() liefert die Anzahl der Felder (Spalten) eines PostgreSQL-Ergebnisses. Das Argument ist eine gültige Ergebniskennung, die von **pg_Exec()** zurückgegeben wurde. Diese Funktion gibt im Fehlerfall -1 aus.

pg_NumRows

(unknown)

pg_NumRows -- Gibt die Anzahl der Zeilen aus

Beschreibung

```
int pg_numrows (int result_id)
```

pg_NumRows() liefert die Anzahl der Zeilen in einem PostgreSQL-Ergebnis. Das Argument ist eine gültige Ergebniskennung, die von **pg_Exec()** zurückgegeben wurde. Diese Funktion gibt im Fehlerfall -1 aus.

pg_Options

(unknown)

pg_Options -- Liefert Verbindungsoptionen

Beschreibung

string **pg_options** (int connection_id)

pg_Options() liefert einen String, der die Optionen einer durch die Kennung bestimmten Verbindung enthält.

pg_pConnect

(unknown)

pg_pConnect -- Stellt eine persistente Datenbankverbindung her

Beschreibung

`int pg_pconnect (string host, string port, string options, string tty, string dbname)`

Gibt bei Erfolg eine Verbindungskennung zurück, false, wenn die Verbindung nicht hergestellt werden konnte. Eine persistente Verbindung zu einer PostgreSQL Datenbank wird geöffnet. Jedes der Argumente, inklusive der Portnummer, sollte ein in Anführungszeichen gesetzter String sein. Die options- und tty-Argumente sind optional und können ausgelassen werden. Diese Funktion liefert eine Verbindungskennung zurück, die von anderen PostgreSQL-Funktionen benötigt wird. Es können mehrere persistente Verbindungen gleichzeitig geöffnet werden. Siehe auch **pg_Connect()**.

Eine Verbindung kann auch mit dem folgenden Befehl hergestellt werden: **\$conn = pg_pconnect("dbname=marliese port=5432");** Weitere Parameter neben *dbname* und *port* sind *host*, *tty*, *options*, *user* und *password*.

pg_Port

(unknown)

pg_Port -- Gibt die Portnummer aus

Beschreibung

```
int pg_port (int connection_id)
```

pg_Port() Gibt die Portnummer der durch die Kennung bezeichneten PostgreSQL-Verbindung aus.

[Zurück](#)[Anfang](#)[Vor](#)

pg_pConnect

[Hoch](#)

pg_Result

pg_Result

(unknown)

pg_Result -- Liefert Werte eines bezeichneten Ergebnisses

Beschreibung

mixed **pg_result** (int result_id, int row_number, mixed fieldname)

pg_Result() liefert Werte aus einem durch **pg_Exec()** produzierten Ergebnis, dessen Kennung übergeben werden muß. *row_number* und *fieldname* geben an, welche Zelle der Ergebnistabelle zurückgegeben werden soll. Die Zeilennumerierung beginnt bei 0. Anstatt des Feldnamens kann auch der Feldindex als Zahl ohne Anführungszeichen angegeben werden. Der Feldindex beginnt bei 0.

PostgreSQL hat viele eingebaute Typen, und nur die grundlegenden werden hier direkt unterstützt. Alle Formen von Integer, Boolean und OID-Typen werden als Integer-Werte zurückgegeben, alle Formen von float und real als double. Alle anderen Formen, einschließlich der Arrays, werden als Strings in der durch den **psql**-Befehl bekannten PostgreSQL-Standardformatierung zurückgeliefert.

pg_trace

(PHP4 >= 4.0.1)

pg_trace -- Trace a connection to PostgreSQL server

Description

`bool pg_trace (string filename [, string mode [, int connection]])`

Enables tracing of the PostgreSQL frontend/backend communication to a debugging file. 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 queriew actually were sent to the PostgreSQL server.

fi lename 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.

Returns TRUE if *fi lename* could be opened for logging, FALSE otherwise.

See also [fopen\(\)](#) and [pg_untrace\(\)](#).

pg_tty

(PHP3 , PHP4)

pg_tty -- Gibt den tty-Namen aus

Beschreibung

string **pg_tty** (int connection_id)

pg_tty() gibt den tty-Namen aus, an den die serverseitigen Debuggermeldungen der durch die Kennung definierten Verbindung geschickt werden.

pg_untrace

(PHP4 >= 4.0.1)

pg_untrace -- Stop tracing a connection to PostgreSQL server

Description

bool **pg_untrace** ([int 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\(\)](#).

XLVIII. Program Execution functions

Inhaltsverzeichnis

[escapeshellcmd](#) escape shell metacharacters

[exec](#) Execute an external program

[passthru](#) Execute an external program and display raw output

[system](#) Execute an external program and display output

[Zurück](#)[Anfang](#)[Vor](#)

pg_untrace

[Hoch](#)

escapeshellcmd

escapeshellcmd

(PHP3 , PHP4)

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:

```
system (EscapeShellCmd($cmd))
```

See also [exec\(\)](#), [popen\(\)](#), [system\(\)](#), and the [backtick operator](#).

exec

(PHP3 , PHP4)

exec -- Execute an external program

Description

string **exec** (string command [, string array [, 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. 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.

Note that if you are going to allow data coming from user input to be passed to this function, then you should be using **EscapeShellCmd()** to make sure that users cannot trick the system into executing arbitrary commands.

See also [system\(\)](#), **PassThru()**, [popen\(\)](#), **EscapeShellCmd()**, and the [backtick operator](#).

passthru

(PHP3 , PHP4)

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.

See also [exec\(\)](#), [system\(\)](#), [popen\(\)](#), [EscapeShellCmd\(\)](#), and the [backtick operator](#).

system

(PHP3 , PHP4)

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.

Note, that if you are going to allow data coming from user input to be passed to this function, then you should be using the **EscapeShellCmd()** function to make sure that users cannot trick the system into executing arbitrary commands.

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](#).

XLIX. Pspell Functions

The **pspell()** functions allows you to check the spelling on a word and offer suggestions.

You need the aspell and pspell libraries, available from: <http://pspell.sourceforge.net/>.

Inhaltsverzeichnis

[pspell_new](#) Load a new dictionary

[pspell_mode](#) Change spellchecking mode

[pspell_runtogether](#) Consider run-together words as legal compounds

[pspell_check](#) Check a word

[pspell_suggest](#) Suggest spellings of a word

pspell_new

(PHP4 CVS only)

pspell_new -- Load a new dictionary

Description

```
int pspell_new (string language [, string spelling [, string jargon [, string encoding]])
```

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 two distinguish two different words lists that have the same language-tag and spelling. 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. For more information and examples, check out inline manual pspell website:<http://pspell.sourceforge.net/>.

Beispiel 1 Pspell_new()

```
$pspell_link = pspell_new ("english");
```

pspell_mode

(PHP4 CVS only)

pspell_mode -- Change spellchecking mode

Description

boolean **pspell_mode** (int dictionary_link, int mode)

Pspell_mode() changes the spellchecking mode.

There are three 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)

Beispiel 1 Pspell_mode()

```
$pspell_link = pspell_new ("english");  
pspell_mode (PSPELL_FAST);  
  
if (!pspell_check ($pspell_link, "testt")) {  
    $suggestions = pspell_suggest ($pspell_link, "testt");  
}
```

pspell_runtogether

(PHP4 CVS only)

pspell_runtogether -- Consider run-together words as legal compounds

Description

boolean **pspell_runtogether** (int dictionary_link, int mode)

Pspell_runtogether() Consider run-together words as legal compounds. That is, "thecat" will be a legal compound, although 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, and they are not affected by the calls to **php_runtogether()**.

Beispiel 1 Pspell_runtogether()

```
$pspell_link = pspell_new ("english");
pspell_runtogether (true);
echo pspell_runtogether ($pspell_link, "thecat") ? "correct" : "wrong";
```

pspell_check

(PHP4 CVS only)

pspell_check -- Check a word

Description

boolean **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.

Beispiel 1 Pspell_check()

```
$pspell_link = pspell_new ("english");

if (pspell_check ($pspell_link, "testt")) {
    echo "This is a valid spelling";
} else {
    echo "Sorry, wrong spelling";
}
```

pspell_suggest

(PHP4 CVS only)

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.

Beispiel 1 Pspell_suggest()

```
$pspell_link = pspell_new ("english");

if (!pspell_check ($pspell_link, "testt")){
    $suggestions = pspell_suggest ($pspell_link, "testt");

    for ($i=0; $i < count ($suggestions); $i++) {
        echo "Possible spelling: " . $suggestions[$i] . "<br>";
    }
}
```

L. GNU Recode functions

This module contains an interface to the GNU Recode library, version 3.5. To be able to use the functions defined in this module you must compile you PHP interpreter using the --with-recode option. In order to do so, you must have GNU Recode 3.5 or higher installed on your system.

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.

Inhaltsverzeichnis

[recode_string](#) Recode a string according to a recode request

[recode](#) Recode a string according to a recode request

[recode_file](#) Recode from file to file according to recode request

recode_string

(PHP3 >= 3.0.13, PHP4 >= 4.0RC1)

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 FALSE, if unable to comply, TRUE otherwise.

A simple recode request may be "lat1..iso646-de". See also the GNU Recode documentation of your installation for detailed instructions about recode requests.

recode

(PHP4 >= 4.0RC1)

recode -- Recode a string according to a recode request

Description

string **recode_string** (string request, string string)

Anmerkung: This is an alias for [recode_string\(\)](#). It has been added in PHP4.

recode_file

(PHP3 >= 3.0.13, PHP4 >= 4.0RC1)

recode_file -- Recode from file to file according to recode request

Description

bool **recode_file** (int input, int 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.

LI. Perl-compatible Regular Expression functions

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.

The ending delimiter may be followed by various modifiers that affect the matching. See [Pattern Modifiers](#).

Beispiel 1 Examples of valid patterns

- `/<\w+>/`
- `|(\d{3})-\d+|Sm`
- `/^(?i)php[34]/`

Beispiel 2 Examples of invalid patterns

- `/href='(.*)'` - missing ending delimiter
- `\w+\s*\w+/J` - unknown modifier 'J'
- `1-\d3-\d3-\d4|` - missing starting delimiter

Anmerkung: The Perl-compatible regular expression functions are available in PHP 4 and in PHP 3.0.9 and up.

Inhaltsverzeichnis

[preg_match](#) Perform a regular expression match

[preg_match_all](#) Perform a global regular expression match

[preg_replace](#) Perform a regular expression search and replace

[preg_split](#) Split string by a regular expression

[preg_quote](#) Quote regular expression characters

[preg_grep](#) Return array entries that match the pattern

[Pattern Modifiers](#) describes possible modifiers in regex patterns

[Pattern Syntax](#) describes PCRE regex syntax

recode_file

[Hoch](#)

preg_match

preg_match

(PHP3 >= 3.0.9, PHP4)

preg_match -- Perform a regular expression match

Description

int **preg_match** (string pattern, string subject [, array matches])

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 match the full pattern, \$matches[1] will have the text that matched the first captured parenthesized subpattern, and so on.

Returns true if a match for *pattern* was found in the subject string, or false if not match was found or an error occurred.

Beispiel 1 Getting the page number out of a string

```
if (preg_match("/page\s+#(\d+)/i", "Go to page #9.", $parts))
    print "Next page is $parts[1]";
else
    print "Page not found.";
```

See also [preg_match_all\(\)](#), [preg_replace\(\)](#), and [preg_split\(\)](#).

preg_match_all

(PHP3 >= 3.0.9, PHP4)

preg_match_all -- Perform a global regular expression match

Description

int **preg_match_all** (string *pattern*, string *subject*, array *matches* [, int *order*])

Searches *subject* for all matches to the regular expression given in *pattern* and puts them in *matches* in the order specified by *order*.

After the first match is found, the subsequent searches are continued on from end of the last match.

order can be one of two things:

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.

```
preg_match_all("|<[^>]+>(.*?)</[^>]+>|U", "<b>example: </b><div align=left>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.

```
preg_match_all("|<[^>]+>(.*?)</[^>]+>|U", "<b>example: </b><div align=left>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.

If *order* is not specified, it is assumed to be `PREG_PATTERN_ORDER`.

Returns the number of full pattern matches, or false if no match is found or an error occurred.

Beispiel 1 Getting all phone numbers out of some text.

```
preg_match_all("/\(? (\d{3})? \)? (? (1) [\-\s] ) \d{3}-\d{4}/x",  
               "Call 555-1212 or 1-800-555-1212", $phones);
```

See also [preg_match\(\)](#), [preg_replace\(\)](#), and [preg_split\(\)](#).

[Zurück](#)[preg_match](#)[Anfang](#)[Hoch](#)[Vor](#)[preg_replace](#)

preg_replace

(PHP3 >= 3.0.9, PHP4)

preg_replace -- Perform a regular expression search and replace

Description

mixed **preg_replace** (mixed pattern, mixed replacement, mixed subject)

Searches *subject* for matches to *pattern* and replaces them with *replacement* .

replacement may contain references of the form `\\n`. 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` 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.

If no matches are found in *subject*, then it will be returned unchanged.

Every parameter to **preg_replace()** can be an array.

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()**.

Anmerkung: This modifier was added in PHP 4.0.

Beispiel 1 Replacing several values

```
$patterns = array("/(19|20\\d{2})-(\\d{1,2})-(\\d{1,2})/", "/^\\s*{\\(\\w+)}\\s*=\\/");
$replace = array("\\3/\\4/\\1", "$\\1 =");
print preg_replace($patterns, $replace, "{startDate} = 1999-5-27");
```

This example will produce:

```
$startDate = 5/27/1999
```

Beispiel 2 Using /e modifier

```
preg_replace("/(<\\/?)(\\w+)([>]*>)/e", "'\\1'.strtoupper('\\2').'\\3'", $html_body);
```

This would capitalize all HTML tags in the input text.

See also [preg_match\(\)](#), [preg_match_all\(\)](#), and [preg_split\(\)](#).

[Zurück](#)

[preg_match_all](#)

[Anfang](#)

[Hoch](#)

[Vor](#)

[preg_split](#)

preg_split

(PHP3 >= 3.0.9, PHP4)

preg_split -- Split string by a regular expression

Description

```
array preg_split (string pattern, string subject [, int limit [, int flags]])
```

Anmerkung: Parameter *flags* was added in PHP 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.

If flags is PREG_SPLIT_NO_EMPTY then only non-empty pieces will be by **preg_split()**.

Beispiel 1 Getting parts of search string

```
$keywords = preg_split("/[\s,]+/", "hypertext language, programming");
```

See also [preg_match\(\)](#), [preg_match_all\(\)](#), and [preg_replace\(\)](#).

preg_quote

(PHP3 >= 3.0.9, PHP4)

preg_quote -- Quote regular expression characters

Description

string **preg_quote** (string str)

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.

The special regular expression characters are:

```
. \ \ + * ? [ ^ ] $ ( ) { } = ! < > | :
```

Anmerkung: This function was added in PHP 3.0.9.

preg_grep

(PHP4)

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*.

Beispiel 1 preg_grep() example

```
preg_grep("/^(\d+)?\.\d+$/", $array); // find all floating point numbers in the array
```

Anmerkung: This function was added in PHP 4.0.

Pattern Modifiers

(unknown)

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 *E* 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 `\` references 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.

Anmerkung: This modifier was added in PHP 4.0.

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.

E (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.

Pattern Syntax

(unknown)

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 lookahead assertions are counted, but their entries in the

offsets vector are never set. Perl sets its numerical variables 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 "\0" 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 pattern 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 repetition quantifiers is inverted, that is, by default they are not greedy, but if followed by a question mark they are.

Regular Expression Details

The syntax and semantics of the regular expressions supported 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 characters 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. The power of regular expressions comes from the ability to include alternatives and repetitions in the pattern. 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)
- .
- [start character class definition
- | start of alternative branch
- (start subpattern
-) end subpattern
- ? extends the meaning of (
 - also 0 or 1 quantifier
 - also quantifier minimizer
- * 0 or more quantifier
- + 1 or more quantifier
- { start 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-alphameric 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 following character would otherwise be interpreted as a meta-character, so it is always safe to precede a non-alphameric with "\" to specify that it stands for itself. In particular, if you want to match a backslash, you write "\\".

If a pattern is compiled with the PCRE_EXTENDED option, whitespace 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\x07" 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 introduced 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 character 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 con-

trolled 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 character 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 assertions. 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` it is always anchored, whether `PCRE_MULTILINE` is set or not.

FULL STOP (PERIOD, DOT)

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 circumflex, 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-alphameric 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 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 question 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". Furthermore, 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".

Assertion subpatterns are not capturing subpatterns, and may not be repeated, because it makes no sense to assert the same thing several times. If an assertion contains capturing subpatterns within it, these are always counted for the purposes of numbering the capturing subpatterns in the whole pattern. Substring capturing is carried out for positive assertions, but 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 look-behind 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 it. 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, 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.

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.

PERFORMANCE

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.

[Zurück](#)

Pattern Modifiers

[Anfang](#)[Hoch](#)[Vor](#)

Regular expression functions

LII. Regular expression functions

Regular expressions are used for complex string manipulation in PHP. The functions that support regular expressions are:

- [ereg\(\)](#)
- [ereg_replace\(\)](#)
- [eregi\(\)](#)
- [eregi_replace\(\)](#)
- [split\(\)](#)

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.

Beispiel 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 carriage return  
   characters in $string. */
```

Inhaltsverzeichnis

[ereg](#) regular expression match

[ereg_replace](#) replace regular expression

[eregi](#) case insensitive regular expression match

[eregi_replace](#) replace regular expression case insensitive

[split](#) split string into array by regular expression

[sql_regcase](#) make regular expression for case insensitive match

[Zurück](#)

Pattern Syntax

[Anfang](#)

[Hoch](#)

[Vor](#)

ereg

ereg

(PHP3 , PHP4)

ereg -- regular expression match

Description

int **ereg** (string pattern, string string [, array regs])

Searchs *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 *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 *string*.

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:

Beispiel 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\(\)](#), and [eregi_replace\(\)](#).

ereg_replace

(PHP3 , PHP4)

ereg_replace -- replace regular expression

Description

string **ereg_replace** (string pattern, string replacement, string string)

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:

Beispiel 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 );
```

See also [ereg\(\)](#), [eregi\(\)](#), and [eregi_replace\(\)](#).

eregi

(PHP3 , PHP4)

eregi -- case insensitive regular expression match

Description

int **eregi** (string pattern, string string [, array regs])

This function is identical to [ereg\(\)](#) save that this ignores case distinction when matching alphabetic characters.

See also [ereg\(\)](#), [ereg_replace\(\)](#), and [eregi_replace\(\)](#).

eregi_replace

(PHP3 , PHP4)

eregi_replace -- replace regular expression case insensitive

Description

string **eregi_replace** (string pattern, string replacement, string string)

This function is identical to [ereg_replace\(\)](#) save that this ignores case distinction when matching alphabetic characters.

See also [ereg\(\)](#), [eregi\(\)](#), and [ereg_replace\(\)](#).

split

(PHP3 , PHP4)

split -- split string into array by regular expression

Description

array **split** (string pattern, 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 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 get the first five fields from a line from `/etc/passwd`:

Beispiel 1 split() example

```
$passwd_list = split( ":", $passwd_line, 5 );
```

To parse a date which may be delimited with slashes, dots, or hyphens:

Beispiel 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.

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: [explode\(\)](#) and [implode\(\)](#).

[Zurück](#)

ereg_replace

[Anfang](#)

[Hoch](#)

[Vor](#)

sql_regcase

sql_regcase

(PHP3 , PHP4)

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.

Beispiel 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.

LIII. Semaphor und Shared Memory Funktionen

Dieses Modul bietet Unterstützung für die Nutzung von Semaphoren und gemeinsamen Speichersegmenten mit Hilfe der UNIX System V Mechanismen zur Interprozess-Kommunikation.

Mit Hilfe von Semaphoren kann der exklusive Zugriff auf Ressourcen eines Systems sichergestellt oder die Anzahl der Prozesse begrenzt werden, die gleichzeitig eine bestimmte Systemresource nutzen.

Mit Hilfe gemeinsamer Speichersegmente (shared memory) können system-globale Variablen angelegt und bereitgestellt werden. Diese Speicherbereiche stehen allen Prozessen des Systems, insbesondere aber allen parallel laufenden PHP-Instanzen zum Datenaustausch zur Verfügung. Beachten Sie bitte, das gemeinsame Speichersegmente keinen Schutze vor konkurrierenden gleichzeitigen Zugriffen bieten. Benutzen Sie Semaphore für die Koordination solcher Zugriffe.

Tabelle 1 Limits of Shared Memory by the Unix OS

SHMMAX	Maximalgröße eines gemeinsamen Speicherbereichs, üblich sind 131072 Byte (128KB)
SHMMIN	Minimalgröße eines gemeinsamen Speicherbereichs, normalerweise 1 Byte
SHMMNI	maximale Anzahl unterschiedlicher Speichersegmente, üblich sind 100
SHMSEG	Anzahl der Speichersegmente, die ein einzelner Prozess einbinden darf, üblich sind 6

Beispiel 1 Koordinierter Zugriff auf gemeinsamen Speicher

```
<?php
// gemeinsames Speichersegment beschaffen
if(! ($mkey = shm_attach(0x2328,1024,OctDec("666"))))
{ echo "shm_attach fehlgeschlagen<br>\n"; exit;}

// Semaphor für Zugriffskoordination auf
// Speichersegment beschaffen
if(! ($skey = sem_get(0x2328,1,OctDec("666"))))
{ echo "sem_get fehlgeschlagen<br>\n"; exit;}

// Zugriff anfordern
if(! sem_acquire($skey))
{ echo "sem_acquire fehlgeschlagen<br>\n"; exit;}

// Wert lesen, aktualisieren und schreiben
$val = @shm_get_var($mkey,1);
```



```
if($val===false) $val=1; else $val++;  
print "new value is $val<br>\n";  
shm_put_var($mkey,1,$val);  
  
// Zugriff freigeben  
sem_release($skey);  
?>
```

Inhaltsverzeichnis

[sem_get](#) Liefert ein Semaphore-Handle

[sem_acquire](#) Zugriff auf einen Semaphore anfordern

[sem_release](#) Freigabe eines angeforderten Semaphores

[shm_attach](#) Anlegen oder anbinden eines gemeinsamen Speichersegments

[shm_detach](#) Beenden der Anbindung an ein gemeinsames Speichersegment

[shm_remove](#) Entfernt ein gemeinsames Speichersegment

[shm_put_var](#) Aktualisiert einen in einem gemeinsamen Speicherbereich angelegten Wert.

[shm_get_var](#) Liest einen in einem gemeinsamen Speicherbereich angelegten Wert.

[shm_remove_var](#) Entfernt einen in einem gemeinsamen Speicherbereich angelegten Wert.

[Zurück](#)

[Anfang](#)

[Vor](#)

sql_regcase

[Hoch](#)

sem_get

sem_get

(PHP3 >= 3.0.6, PHP4)

sem_get -- Liefert ein Semaphore-Handle

Beschreibung

```
int sem_get (int key [, int max_acquire [, int perm]])
```

Diese Funktion liefert ein Handle, das für den Zugriff auf das Semaphore mit der Kennung *key* benutzt werden kann.

Falls noch kein Semaphore mit dieser Kennung existiert, so wird es neu angelegt. Die maximale Anzahl der gleichzeitig möglichen Anforderungen und die Zugriffsrechte werden dabei über die Parameter *max_acquire* und **Sem_get()** festgelegt.

Wird der angeforderte Semaphore zum Zeitpunkt des Aufrufs von keiner anderen Funktion genutzt, so kann über den optionalen Parameter *max_acquire* die Anzahl der gleichzeitig möglichen Anforderungen neu festgelegt werden, anderenfalls bleibt der ursprüngliche Wert erhalten.

Ein zweiter Aufruf der Funktion mit demselben *key* liefert ein weiteres, von dem ersten verschiedenes, Handle, das aber trotzdem denselben darunterliegenden Semaphore anspricht.

Wenn kein Semaphore mit der Kennung *key* angelegt werden kann oder die Rechte für den Zugriff auf einen bestehendennicht ausreichen, so ist das Ergebnis dieser Funktion der Wert FALSE.

Siehe auch [sem_acquire\(\)](#) und [sem_release\(\)](#).

sem_acquire

(PHP3 >= 3.0.6, PHP4)

sem_acquire -- Zugriff auf einen Semaphore anfordern

Beschreibung

int sem_acquire (int sem_identifizier)

sem_acquire() blockiert die Ausführung (falls notwendig) solange, bis eine Anforderung eines Semaphores durchgeführt werden kann, d.h. solange die Anzahl der gleichzeitigen Anforderungen bereits der maximalen Anzahl entspricht.

Wird ein erfolgreich angeforderter Sempahore innerhalb eines Requests nicht wieder freigegeben, so erfolgt die Freigabe nach Abarbeitung des Requests automatisch und es wird eine Warnung ausgegeben.

Diese Funktion liefert TRUE bei Erfolg und FALSE, wenn eine Anforderung des Semaphores wegen eines Fehlers nicht möglich ist.

Siehe auch [sem_get\(\)](#) und [sem_release\(\)](#).

Warnung

Wird ein Semaphore ein zweites mal während desselben Request angefordert, ohne zwischenzeitlich freigegeben worden zu sein, so kann dies dazu führen, das die weitere Ausführung dieses Requests nicht mehr möglich ist.

sem_release

(PHP3 >= 3.0.6, PHP4)

sem_release -- Freigabe eines angeforderten Semaphores

Beschreibung

`int sem_release (int sem_identifizier)`

sem_release() gibt einen zuvor angeforderten Semaphore wieder frei. Die Anforderung und Freigabe kann innerhalb eines Prozesses mehrfach hintereinander erfolgen.

Die Funktion liefert TRUE, wenn ein angeforderter Semaphore erfolgreich freigegeben wurde, ansonsten FALSE.

Siehe auch [sem_get\(\)](#) und [sem_acquire\(\)](#).

shm_attach

(PHP3 >= 3.0.6, PHP4)

shm_attach -- Anlegen oder anbinden eines gemeinsamen Speichersegments

Beschreibung

int shm_attach (int key [, int memsize [, int perm]])

shm_attach() gibt ein Handle für den Zugriff auf den gemeinsamen Speicherbereich mit der Kennung *key* zurück.

Falls noch kein Bereich mit dieser Kennung existiert, so wird er mit einer Größe von *memsize* Byte (Vorgabe ist der Parameter *sysvshm.init_mem* aus der [Konfigurationsdatei](#) bzw. 10000 Byte) und den Zugriffsberechtigungen gemäß *perm* (Vorgabe ist der Wert 0666, der Lese- und Schreibzugriffe für alle Benutzer gestattet) an.

Ein zweiter Aufruf der Funktion mit demselben *key* liefert ein weiteres, von dem ersten verschiedenes, Handle, das aber trotzdem denselben gemeinsamen Speicherbereich anspricht.

Wenn kein Bereich mit der Kennung *key* angelegt werden kann oder die Rechte für den Zugriff auf ein bestehendes Segment nicht ausreichen, so ist das Ergebnis dieser Funktion der Wert FALSE.

shm_detach

(PHP3 >= 3.0.6, PHP4)

shm_detach -- Beenden der Anbindung an ein gemeinsames Speichersegment

Beschreibung

int **shm_detach** (int shm_identifizier)

Shm_detach() beendet die Anbindung eines gemeinsamen Speicherbereichs mit dem Handle *shm_identifizier*, das zuvor über [shm_attach\(\)](#) angebunden wurde. Das Segment an sich und die darin enthaltenen Daten bleiben erhalten und stehen nach erneuter Anbindung in diesem oder einem anderen Prozess wieder zur Verfügung.

shm_remove

(PHP3 >= 3.0.6, PHP4)

shm_remove -- Entfernt ein gemeinsames Speichersegment

Beschreibung

int **shm_remove** (int key)

Diese Funktion gibt ein gemeinsames Speichersegment wieder frei, dabei gehen alle Daten innerhalb des Segmentes verloren.

Diese Funktion liefert TRUE, wenn das Segment erfolgreich entfernt wurde, oder FALSE, wenn kein Segment mit der Kennung *key* existierte oder die Zugriffsrechte auf ein bestehendes Segment nicht für ein Löschen ausreichen.

shm_put_var

(PHP3 >= 3.0.6, PHP4)

shm_put_var -- Aktualisiert einen in einem gemeinsamen Speicherbereich angelegten Wert.

Beschreibung

int **shm_put_var** (int shm_identifizier, int variable_key, mixed variable)

Diese Funktion weist einer Variable mit der Kennung *variable_key* in einem gemeinsamen Speichersegment den Wert von *variable* zu, beim ersten Zugriff wird *variable_key* neu angelegt.

Diese Funktion liefert FALSE zurück, wenn der verbleibende Speicherplatz im angegebenen Segment *shm_identifizier* zur Aufnahme des Wertes von *variable* nicht mehr ausreicht, ansonsten TRUE.

shm_get_var

(PHP3 >= 3.0.6, PHP4)

shm_get_var -- Liest einen in einem gemeinsamen Speicherbereich angelegten Wert.

Beschreibung

mixed **shm_get_var** (int id, int variable_key)

Shm_get_var() liefert den Wert mit der Kennung *variable_key* aus einem gemeinsamen Speicherbereich. Der Wert im gemeinsamen Speicher bleibt weiter vorhanden. Existiert kein Wert mit der angegebenen Kennung, so wird eine Warnung erzeugt und FALSE zurückgegeben.

shm_remove_var

(PHP3 >= 3.0.6, PHP4)

shm_remove_var -- Entfernt einen in einem gemeinsamen Speicherbereich angelegten Wert.

Beschreibung

```
int shm_remove_var (int id, int variable_key)
```

Diese Funktion entfernt den Wert mit der Kennung *variable_key* aus einem gemeinsamen Speichersegment und stellt den hiervon genutzten Speicher innerhalb des Segmentes wieder zur Verfügung.

Diese Funktion liefert TRUE bei Erfolg und FALSE, falls kein Wert mit der Kennung *variable_key* im Speichersegment angelegt war.

LIV. Session Funktionen

Die Unterstützung von Sessions in PHP bietet die Möglichkeit, bestimmte Daten während einer Folge von Aufrufen ihrer Website festzuhalten. Damit können Sie persönlichere Applikationen erstellen und Ihre Website ansprechender gestalten.

Falls sie mit dem Sessionmanagment von PHPLIB vertraut sind, werden sie Ähnlichkeiten zu PHP's Session-Unterstützung feststellen.

Einem Besucher wird beim Aufruf ihrer Website eine einzigartige ID, die sogenannte Session-ID, zugeordnet. Diese wird entweder benutzerseitig in einem Cookie abgelegt oder in der URL übermittelt.

Die Unterstützung von Sessions erlaubt Ihnen, eine beliebige Anzahl von Variablen zu registrieren und diese über Anfragen hinweg zu erhalten. Wenn ein Besucher Ihre Website aufruft, überprüft PHP automatisch (falls `session.auto_start` auf 1 gesetzt ist) oder auf Anfrage (explizit durch [session_start\(\)](#) oder implizit durch [session_register\(\)](#)), ob mit der Anfrage schon eine bestimmte Session-ID gesendet wurde. Wenn dies der Fall ist, wird die zuvor gespeicherte Umgebung wiederhergestellt.

Alle registrierten Variablen werden serialisiert, nachdem die Anfrage beendet ist. Registrierte Variablen, die nicht definiert sind, werden als nicht definiert gekennzeichnet. Auch bei darauffolgenden Zugriffen werden sie nicht vom Session-Modul definiert, außer der Benutzer definiert sie später.

Die Konfigurationseinstellungen von `track_vars` und `register_globals` beeinflussen, wie die Session-Variablen gespeichert und wiederhergestellt werden.

Falls `track_vars` aktiviert und `track_vars` deaktiviert ist, können nur Bestandteile des globalen assoziativen Arrays `$HTTP_SESSION_VARS` als Session-Variablen registriert werden. Wiederhergestellte Session-Variablen stehen nur im Array `$HTTP_SESSION_VARS` zur Verfügung.

Beispiel 1 Registrierung einer Variablen bei aktiviertem `track_vars`

```
<?php
session_register("count");
$HTTP_SESSION_VARS["count"]++;
?>
```

Wenn `register_globals` aktiviert ist, können alle globalen Variablen als Session-Variablen registriert werden und die Session-Variablen werden als entsprechende globale Variablen wiederhergestellt.

Beispiel 2 Registrierung einer Variablen bei aktiviertem `register_globals`

```
<?php
session_register("count");
$count++;
?>
```

Wenn sowohl `track_vars` als auch `register_globals` aktiviert sind, dann referenzieren die globalen Variablen und die Einträge von `$HTTP_SESSION_VARS` den selben Wert.

Es gibt zwei Methoden, eine Session-ID zu übermitteln:

- Cookies
- URL Parameter

Das Session-Modul unterstützt beide Methoden. Cookies sind optimal, aber da sie nicht zuverlässig sind (Clients müssen sie nicht unbedingt akzeptieren), können wir uns nicht auf sie verlassen. Die zweite Methode hängt die Session-ID direkt an die URLs.

Wenn PHP mit `--enable-trans-sid` kompiliert wurde, kann es dies transparent machen. Wenn sie diese Option aktivieren, werden die URIs automatisch geändert, sodass sie die Session-ID enthalten. Alternativ können sie die Konstante `SID` verwenden, die definiert wird, falls vom Client kein passendes Cookie gesendet wurde. `SID` hat entweder die Form `session_name=session_id` oder ist ein leerer String.

Das folgende Beispiel demonstriert, wie eine Variable registriert wird und wie unter Verwendung einer SID korrekt auf eine andere Seite verwiesen wird.

Beispiel 3 Zählen der Seitenaufrufe eines einzelnen Benutzers

```
<?php
session_register("count");
$count++;
?>
```

Hallo Besucher, Sie haben diese Seite <? echo \$count; ?> Mal aufgerufen.<p>

```
<?php
# für den Fall, dass der Benutzer Cookies deaktiviert hat,
# ist das <?=SID?> nötig, um die Session-ID zu erhalten
?>
```

Hier gehts <A HREF="nextpage.php?<?=SID?>">weiter

Zur Realisierung einer Speicherung in einer Datenbank brauchen Sie PHP-Code und eine Funktion **`session_set_save_handler()`** auf der Benutzerebene. Sie müssten die folgenden Funktionen erweitern, um damit MySQL oder eine andere Datenbank abzudecken.

Beispiel 4 Die Verwendung von session_set_save_handler()

```
<?php

function open ($save_path, $session_name) {
    echo "open ($save_path, $session_name)\n";
    return true;
}

function close() {
    echo "close\n";
    return true;
}

function read ($key) {
    echo "read ($key)\n";
    return "foo|i:1;";
}

function write ($key, $val) {
    echo "write ($key, $val)\n";
    return true;
}

function destroy ($key) {
    return true;
}

function gc ($maxlifetime) {
    return true;
}

session_set_save_handler ("open", "close", "read", "write", "destroy", "gc");

session_start();

$foo++;

?>
```

Erzeugt die folgenden Resultate:

```
$ ./php save_handler.php
Content-Type: text/html
Set-cookie: PHPSESSID=f08b925af0ecb52bdd2de97d95cdbe6b

open (/tmp, PHPSESSID)
read (f08b925af0ecb52bdd2de97d95cdbe6b)
write (f08b925af0ecb52bdd2de97d95cdbe6b, foo|i:2;)
close
```

Das `<?=SID?>` ist nicht nötig, wenn PHP mit `--enable-trans-sid` kompiliert wurde.

Das Sessionmanagementsystem unterstützt eine Anzahl von Konfigurationsoptionen, die sie in Ihrer `php.ini` setzen können. Wir geben Ihnen dazu einen kleinen Überblick.

- `session.save_handler` definiert den Namen der Prozedur, die benutzt wird, um die Daten zu speichern und zurückzuholen, die mit der Session in Verbindung stehen. Grundeinstellung `files`.
- `session.save_path` definiert das Argument, das an die Speicherprozedur übergeben wird. Wenn Sie die standardmäßige `files` Prozedur wählen, ist das der Pfad, unter dem die Dateien erzeugt werden. Grundeinstellung `/tmp`.
- `session.name` spezifiziert den Namen der Session, der als Cookie-Name verwendet wird. Grundeinstellung `PHPSESSID`.
- `session.auto_start` spezifiziert, ob das Session-Modul zu Beginn einer Anfrage automatisch eine Session startet. Grundeinstellung `0` (deaktiviert).
- `session.cookie_lifetime` spezifiziert die Cookie-Lebensdauer, die an den Browser geschickt wird, in Sekunden. Der Wert `0` bedeutet "bis der Browser geschlossen wird." Grundeinstellung `0`.
- `session.serialize_handler` definiert den Namen der Prozedur, die benutzt wird, um Daten zu serialisieren/deserialisieren. Gegenwärtig wird ein internes PHP-Format (Name `php`) und `WDDX` (name `wddx`) unterstützt. `WDDX` steht nur zur Verfügung, wenn PHP mit [WDDX support](#) kompiliert wurde. Grundeinstellung `php`.
- `session.gc_probability` spezifiziert die prozentuale Wahrscheinlichkeit, dass die `gc` (garbage collection) Routine bei jeder Anfrage gestartet wird. Grundeinstellung `1`.
- `session.gc_maxlifetime` spezifiziert die Anzahl der Sekunden, nach denen Daten als 'garbage' ('Müll') betrachtet und entsorgt werden.
- `session.referer_check` bestimmt, ob Session-IDs beseitigt werden, die von externen Seiten uebergeben werden. Wenn Session-IDs durch Verwendung der URL-Methode übermittelt werden, könnten Benutzer, die sich dieser Wirkung nicht bewusst sind, Session-IDs veröffentlichen. Das kann zu Sicherheitsproblemen führen, die diese Überprüfung zu vermeiden versucht. Grundeinstellung `0`.
- `session.entropy_file` gibt den Pfad zu einer externen Quelle (Datei) an, die bei der Erzeugung einer Session-ID als zusätzliche Entropiequelle verwendet wird. Beispiele sind `/dev/random` oder `/dev/urandom`, die auf vielen Unix-Systemen zur Verfügung stehen.
- `session.entropy_length` spezifiziert die Anzahl der Bytes, die von der oben spezifizierten Datei gelesen werden. Grundeinstellung `0` (deaktiviert).

- `session.use_cookies` spezifiziert, ob das Modul Cookies verwendet, um die Session-ID clientseitig zu speichern. Grundeinstellung 1 (aktiviert).
- `session.cookie_path` spezifiziert den Pfad, in dem das Session-Cookie gesetzt wird. Grundeinstellung `/`.
- `session.cookie_domain` spezifiziert die Domain, unter der das Session-Cookie gesetzt wird. In der Grundeinstellung überhaupt keine.
- `session.cache_limiter` spezifiziert die Methode der Cacheverwaltung, die bei Session-Seiten benutzt wird (`nocache/private/public`). Grundeinstellung `nocache`.
- `session.cache_expire` spezifiziert in Minuten, wie lange Session-Seiten im Cache bleiben. Bei `nocache` ist diese Angabe wirkungslos. Grundeinstellung 180.

Anmerkung: Die Behandlung von Sessions wurde in PHP 4.0 eingeführt.

Inhaltsverzeichnis

[session_start](#) Initialisierung einer Session

[session_destroy](#) Löscht alle in einer Session registrierten Daten

[session_name](#) Abfragen und/oder setzen des Namens der aktuellen Session

[session_module_name](#) Abfragen und/oder setzen des aktuellen Session-Moduls

[session_save_path](#) Abfragen und/oder setzen des aktuellen session save path

[session_id](#) Abfragen und/oder setzen der aktuellen Session-ID

[session_register](#) Eine oder mehrere Variablen in der aktuellen Session registrieren

[session_unregister](#) Die Aufhebung der Registrierung einer Variablen in der aktuellen Session

[session_unset](#) Befreien aller Session-Variablen

[session_is_registered](#) Überprüfen, ob eine Variable in einer Session registriert ist

[session_get_cookie_params](#) Abfragen der Session-Cookie Parameter

[session_set_cookie_params](#) Setzen der Session-Cookie Parameter

[session_decode](#) Decodiert die Daten einer Session aus einem String

[session_encode](#) Codiert die Daten der aktuellen Session als String

[Zurück](#)

shm_remove_var

[Anfang](#)

[Hoch](#)

[Vor](#)

session_start

session_start

(PHP4)

session_start -- Initialisierung einer Session

Beschreibung

```
bool session_start(void);
```

session_start() erzeugt eine Session (oder nimmt die aktuelle, basierend auf der Session-ID, die über eine GET-Variable oder ein Cookie übermittelt wurde, wieder auf).

Diese Funktion gibt immer true zurück.

Anmerkung: Diese Funktion wurde in PHP 4.0 eingeführt.

session_destroy

(PHP4)

session_destroy -- Löscht alle in einer Session registrierten Daten

Beschreibung

```
bool session_destroy(void);
```

session_destroy() löscht alle in Verbindung mit der aktuellen Session stehenden Daten.

Diese Funktion gibt nach erfolgreichem Löschen der Daten true zurück und false, wenn dabei ein Fehler auftritt.

Anmerkung: Diese Funktion wurde in PHP 4.0 eingeführt.

session_name

(PHP4)

session_name -- Abfragen und/oder setzen des Namens der aktuellen Session

Beschreibung

string **session_name** ([string name])

session_name() gibt den Namen der aktuellen Session zurück. Wenn *name* angegeben wird, wird der Name der aktuellen Session durch dessen Wert ersetzt.

Der Session-Name referenziert die Session-ID in Cookies und URLs. Er sollte nur alphanumerische Zeichen enthalten und sollte kurz und beschreibend sein (z.B. für Benutzer mit aktivierter Cookie-Warnung). Der Session-Name wird auf den Ausgangswert zurückgesetzt, der zu Beginn in `session.name` gespeichert wurde. Deshalb muss **session_name()** bei jeder Anfrage aufgerufen werden (und bevor [session_start\(\)](#) oder [session_register\(\)](#) aufgerufen wurden).

Beispiel 1 session_name() Beispiele

```
<?php
# den Session-Namen auf WebsiteID setzen

$previous_name = session_name ("WebsiteID");

echo "Der vorhergehende Session-Name war $previous_name<p>";
?>
```

Anmerkung: Diese Funktion wurde in PHP 4.0 eingeführt.

session_module_name

(PHP4)

session_module_name -- Abfragen und/oder setzen des aktuellen Session-Moduls

Beschreibung

string **session_module_name** ([string module])

session_module_name() gibt den Namen des aktuellen Session-Moduls zurück. Wenn *module* angegeben wird, wird stattdessen dieses Modul verwendet.

Anmerkung: Diese Funktion wurde in PHP 4.0 eingeführt.

session_save_path

(PHP4)

session_save_path -- Abfragen und/oder setzen des aktuellen session save path

Beschreibung

string **session_save_path** ([string path])

session_save_path() gibt den Pfad des aktuellen Verzeichnisses zurück, das zum Speichern der Session-Daten verwendet wird. Wenn *path* angegeben wird, wird der Pfad, unter dem Daten gespeichert werden, entsprechend geändert.

Anmerkung: Unter manchen Betriebssystemen, wollen Sie vielleicht einen Pfad auf einem Dateisystem angeben, das sehr viele kleine Dateien effizient behandelt. Unter Linux, zum Beispiel, bietet reiserfs eventuell eine bessere Performance als ext2fs.

Anmerkung: Diese Funktion wurde in PHP 4.0 eingeführt.

session_id

(PHP4)

session_id -- Abfragen und/oder setzen der aktuellen Session-ID

Beschreibung

string **session_id** ([string *id*])

session_id() gibt die Session-ID der aktuellen Session zurück. Wenn *id* angegeben wird, wird die die aktuelle Session-ID ersetzt.

Die Konstante SID kann auch dazu verwendet werden, den aktuellen Namen und Session-ID als String, passend zum Hinzufügen zu URLs, zu erhalten.

session_register

(PHP4)

session_register -- Eine oder mehrere Variablen in der aktuellen Session registrieren

Beschreibung

`bool session_register (mixed name [, mixed ...])`

session_register() variable Anzahl von Argumenten die jeweils entweder ein String sein können, der den Variablennamen trägt, oder ein Array, der aus solchen Variablennamen oder anderen Arrays besteht. Für jeden Variablennamen registriert **session_register()** die globale Variable in der aktuellen Session unter diesem Namen.

Diese Funktion gibt true zurück, wenn die Variable erfolgreich in der Session registriert wurde.

Anmerkung: Diese Funktion wurde in PHP 4.0 eingeführt.

session_unregister

(PHP4)

session_unregister -- Die Aufhebung der Registrierung einer Variablen in der aktuellen Session

Beschreibung

bool **session_unregister** (string name)

session_unregister() hebt die Registrierung der globalen Variablen namens *name* in der aktuellen Session auf.

Diese Funktion gibt true zurück, wenn die Aufhebung der Registrierung der Variablen in der Session erfolgt ist.

Anmerkung: Diese Funktion wurde in PHP 4.0 eingeführt.

session_unset

(PHP4 >= 4.0b4)

session_unset -- Befreien aller Session-Variablen

Beschreibung

```
void session_unset (void);
```

Die Funktion **session_unset()** befreit alle Session-Variablen, die gegenwärtig registriert sind.

session_is_registered

(PHP4)

session_is_registered -- Überprüfen, ob eine Variable in einer Session registriert ist

Beschreibung

bool **session_is_registered** (string name)

session_is_registered() gibt true zurück, wenn in der aktuellen Session eine Variable mit dem Namen *name* registriert ist.

Anmerkung: Diese Funktion wurde in PHP 4.0 eingeführt.

session_get_cookie_params

(PHP4 >= 4.0RC2)

session_get_cookie_params -- Abfragen der Session-Cookie Parameter

Beschreibung

array **session_get_cookie_params** (void);

Die Funktion **session_get_cookie_params()** gibt einen Array mit der Information des aktuellen Session-Cookies zurück. Der Array enthält folgende Felder:

- "lifetime" - Die Lebensdauer des Cookies.
- "path" - Der Pfad, unter dem Informationen gespeichert sind.
- "domain" - Die Domain des Cookies.

session_set_cookie_params

(PHP4 >= 4.0b4)

session_set_cookie_params -- Setzen der Session-Cookie Parameter

Beschreibung

```
void session_set_cookie_params (int lifetime [, string path [, string domain]])
```

Setzen der Cookie-Parameter, die in der php.ini definiert sind. Diese Funktion wirkt sich nur für die Dauer des Scripts aus.

session_decode

(PHP4)

session_decode -- Decodiert die Daten einer Session aus einem String

Beschreibung

bool **session_decode** (string data)

session_decode() decodiert die Daten der Session in *data* und setzt die Variablen, die in der Session gespeichert sind.

Anmerkung: Diese Funktion wurde in PHP 4.0 eingeführt.

session_encode

(PHP4)

session_encode -- Codiert die Daten der aktuellen Session als String

Beschreibung

```
bool session_encode(void);
```

session_encode() gibt einen String mit dem codierten Inhalt der aktuellen Session zurück.

Anmerkung: Diese Funktion wurde in PHP 4.0 eingeführt.

LV. Shockwave Flash functions

PHP ermöglicht es Shockwave Flash Dateien mit Paul Haeberli's libswf Modul zu erzeugen. Sie können das libswf-Modul unter <http://reality.sgi.com/grafica/flash/> herunterladen. Wenn Sie das Modul geladen haben, brauchen Sie nur noch PHP mit dem Modul zu konfigurieren (`--with-swf[=DIR]`), wobei DIR für das Verzeichnis steht, welches die Verzeichnisse include und lib beinhaltet. Das include-Verzeichnis muss die swf.h-Datei und das lib-Verzeichnis die libswf.a-Datei enthalten. Wenn Sie die Distribution entpacken, werden beide Dateien in ein Verzeichnis entpackt. Daher müssen Sie die Dateien manuell in die richtigen Verzeichnisse kopieren.

Haben Sie PHP erfolgreich mit Shockwave Flash Unterstützung installiert, können Sie Shockwave Flash-Dateien via PHP erstellen. Sie werden überrascht sein von den Möglichkeiten, die sich Ihnen bieten. Schauen Sie sich den folgenden Quellcode-Ausschnitt an:

Beispiel 1 SWF Beispiel

```
<?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 ();
?>

```

Dieses Beispiel wird die Animation generieren, die Sie unter folgender Adresse (URL) ansehen können: [url](#).

Anmerkung: SWF-Unterstützung wurde in PHP4 RC2 hinzugefügt.

Inhaltsverzeichnis

- [swf_openfile](#) Öffnet eine neue Shockwave Flash Datei
- [swf_closefile](#) Schließt die aktuelle SWF-Datei
- [swf_labelframe](#) Beschriftet den aktuellen Frame
- [swf_showframe](#) Zeigt den aktuellen Frame an
- [swf_setframe](#) Wechselt zu einem spezifizierten Frame
- [swf_getframe](#) Holt die Frame-Nummer des aktuellen Frames
- [swf_mulcolor](#) Setzt die "global multiply"-Farbe auf den spezifizierten rgba-Wert
- [swf_addcolor](#) Setzt die "global add"-Farbe auf den spezifizierten rgba-Wert
- [swf_placeobject](#) Place an object onto the screen
- [swf_modifyobject](#) Modify an object
- [swf_removeobject](#) Remove an object
- [swf_nextid](#) Returns the next free object id
- [swf_startdoaction](#) Start a description of an action list for the current frame

[swf_actiongotoframe](#) Play a frame and then stop

[swf_actiongeturl](#) Get a URL from a Shockwave Flash movie

[swf_actionnextframe](#) Go forward one frame

[swf_actionprevframe](#) Go backwards one frame

[swf_actionplay](#) Start playing the flash movie from the current frame

[swf_actionstop](#) Stop playing the flash movie at the current frame

[swf_actiontogglequality](#) Toggle between low and high quality

[swf_actionwaitforframe](#) Skip actions if a frame has not been loaded

[swf_actionsettarget](#) Set the context for actions

[swf_actiongotolabel](#) Display a frame with the specified label

[swf_enddoaction](#) End the current action

[swf_defineline](#) Define a line

[swf_definerect](#) Define a rectangle

[swf_definepoly](#) Define a polygon

[swf_startshape](#) Start a complex shape

[swf_shapelinesolid](#) Set the current line style

[swf_shapefilloff](#) Turns off filling

[swf_shapefillsolid](#) Set the current fill style to the specified color

[swf_shapefillbitmaptile](#) Set current fill mode to clipped bitmap

[swf_shapefillbitmaptile](#) Set current fill mode to tiled bitmap

[swf_shapemoveto](#) Move the current position

[swf_shapelineto](#) Draw a line

[swf_shapecurveto](#) Draw a quadratic bezier curve between two points

[swf_shapecurveto3](#) Draw a cubic bezier curve

[swf_shapearc](#) Draw a circular arc

[swf_endshape](#) Completes the definition of the current shape

[swf_definefont](#) Defines a font

[swf_setfont](#) Change the current font

[swf_fontsize](#) Change the font size

[swf_fontslant](#) Set the font slant

[swf_fonttracking](#) Set the current font tracking

[swf_getfontinfo](#) The height in pixels of a capital A and a lowercase x

[swf_definetext](#) Define a text string

[swf_textwidth](#) Get the width of a string

[swf_definebitmap](#) Define a bitmap

[swf_getbitmapinfo](#) Get information about a bitmap

[swf_startsymbol](#) Define a symbol

[swf_endsymbol](#) End the definition of a symbol

[swf_startbutton](#) Start the definition of a button

[swf_addbuttonrecord](#) Controls location, appearance and active area of the current button

[swf_oncondition](#) Describe a transition used to trigger an action list

[swf_endbutton](#) End the definition of the current button

[swf_viewport](#) Select an area for future drawing

[swf_ortho](#) Defines an orthographic mapping of user coordinates onto the current viewport

[swf_ortho2](#) Defines 2D orthographic mapping of user coordinates onto the current viewport

[swf_perspective](#) Define a perspective projection transformation

[swf_polarview](#) Define the viewer's position with polar coordinates

[swf_lookat](#) Define a viewing transformation

[swf_pushmatrix](#) Push the current transformation matrix back unto the stack

[swf_popmatrix](#) Restore a previous transformation matrix

[swf_scale](#) Scale the current transformation

[swf_translate](#) Translate the current transformations

[swf_rotate](#) Rotate the current transformation

[swf_posround](#) Enables or Disables the rounding of the translation when objects are placed or moved

[Zurück](#)

[Anfang](#)

[Vor](#)

session_encode

[Hoch](#)

swf_openfile

swf_openfile

(PHP4 >= 4.0RC2)

swf_openfile -- Öffnet eine neue Shockwave Flash Datei

Beschreibung

void **swf_openfile** (string filename, float width, float height, float framerate, float r, float g, float b)

Die Funktion **swf_openfile()** öffnet eine neue SWF-Datei mit dem Namen *filename*, einer Breite von *width*, einer Höhe von *height*, einer Frame-Rate von *framerate* und einem Hintergrund mit einem roten Farbwert von *r*, einem grünen Farbwert von *g* und einem blauen Farbwert von *b*.

Die Funktion **swf_openfile()** muss die erste Funktion sein, die Sie aufrufen, andernfalls wird Ihr Skript einen Segfault verursachen. Wenn Sie den Output des Skriptes auf dem Bildschirm ausgeben wollen, benutzen Sie filename: "php://stdout" (Unterstützung ab PHP 4.0.1 und höher).

swf_closefile

(PHP4 >= 4.0RC2)

swf_closefile -- Schließt die aktuelle SWF-Datei

Beschreibung

```
void swf_closefile (void);
```

Schließt eine Datei die zuvor mit **swf_openfile ()** geöffnet wurde.

[Zurück](#)[Anfang](#)[Vor](#)

swf_openfile

[Hoch](#)

swf_labelframe

swf_labelframe

(PHP4 >= 4.0RC2)

swf_labelframe -- Beschriftet den aktuellen Frame

Beschreibung

void **swf_labelframe** (string name)

Beschriftet den aktuellen Frame mit der Bezeichnung, die durch das Parameter *name* angegeben wurde.

swf_showframe

(PHP4 >= 4.0RC2)

swf_showframe -- Zeigt den aktuellen Frame an

Beschreibung

```
void swf_showframe (void);
```

Die swf_showframe() Funktion gibt den aktuellen Frame aus.

swf_setframe

(PHP4 >= 4.0RC2)

swf_setframe -- Wechselt zu einem spezifizierten Frame

Beschreibung

```
void swf_setframe (int framenummer)
```

Die **swf_setframe()** Funktion ändert den aktuellen Frame in den mit *framenummer* spezifizierten Frame.

swf_getframe

(PHP4 >= 4.0RC2)

swf_getframe -- Holt die Frame-Nummer des aktuellen Frames

Beschreibung

```
int swf_getframe (void);
```

Die **swf_getframe()** Funtkion holt die Nummer des aktuellen Frames.

swf_mulcolor

(PHP4 >= 4.0RC2)

swf_mulcolor -- Setzt die "global multiply"-Farbe auf den spezifizierten rgba-Wert

Beschreibung

void **swf_mulcolor** (float r, float g, float b, float a)

Die **swf_mulcolor()** Funktion setzt die "global multiply"-Farbe auf die mit *rgba* spezifizierte Farbe. Diese Farbe wird (implizit) von den Funktionen [swf_placeobject\(\)](#), [swf_modifyobject\(\)](#) und **swf_addbuttonrecord()** verwendet. Die Farbe des Objekts wird vervielfacht mit dem Wert von *rgba* , wenn das Objekt auf dem bildschirm ausgegeben wird.

Anmerkung: Das *rgba* Parameter kann sowohl negative wie auch positive Werte annehmen.

swf_addcolor

(PHP4 >= 4.0RC2)

swf_addcolor -- Setzt die "global add"-Farbe auf den spezifizierten rgba-Wert

Beschreibung

void **swf_addcolor** (float r, float g, float b, float a)

Die Funktion **swf_addcolor()** setzt die "global add"-Farbe auf den mit *rgba* spezifizierten Farbwert. Diese Farbe wird (implizit) von den Funktionen **swf_placeobject()**, [swf_modifyobject\(\)](#) und [swf_addbuttonrecord\(\)](#) verwendet. Die Farbe wird dem Objekt durch den *rgba*-Wert hinzugefügt, wenn es auf dem Bildschirm ausgegeben wird.

Anmerkung: Das *rgba* Parameter kann sowohl negative wie auch positive Werte annehmen.

swf_placeobject

(PHP4 >= 4.0RC2)

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.

Anmerkung: Full RGBA colors are supported.

swf_modifyobject

(PHP4 >= 4.0RC2)

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_removeobject

(PHP4 >= 4.0RC2)

swf_removeobject -- Remove an object

Description

void **swf_removeobject** (int depth)

Removes the object at the depth specified by *depth*.

swf_nextid

(PHP4 >= 4.0RC2)

swf_nextid -- Returns the next free object id

Description

```
int swf_nextid (void);
```

The **swf_nextid()** function returns the next available object id.

[Zurück](#)[Anfang](#)[Vor](#)

swf_removeobject

[Hoch](#)

swf_startdoaction

swf_startdoaction

(PHP4 >= 4.0RC2)

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_actiongotoframe

(PHP4 >= 4.0RC2)

swf_actiongotoframe -- Play a frame and then stop

Description

void **swf_actiongotoframe** (int framenummer)

The **swf_actionGotoFrame()** function will go to the frame specified by *framenummer*, play it, and then stop.

swf_actiongeturl

(PHP4 >= 4.0RC2)

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*.

[Zurück](#)[Anfang](#)[Vor](#)

swf_actiongotoframe

[Hoch](#)

swf_actionnextframe

swf_actionnextframe

(PHP4 >= 4.0RC2)

swf_actionnextframe -- Go foward one frame

Description

```
void swf_actionnextframe (void);
```

Go foward one frame.

[Zurück](#)[Anfang](#)[Vor](#)

swf_actiongeturl

[Hoch](#)

swf_actionprevframe

swf_actionprevframe

(PHP4 >= 4.0RC2)

swf_actionprevframe -- Go backwards one frame

Description

```
void swf_actionprevframe (void);
```

[Zurück](#)[Anfang](#)[Vor](#)

swf_actionnextframe

[Hoch](#)

swf_actionplay

swf_actionplay

(PHP4 >= 4.0RC2)

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.

[Zurück](#)[Anfang](#)[Vor](#)

swf_actionprevframe

[Hoch](#)

swf_actionstop

swf_actionstop

(PHP4 >= 4.0RC2)

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.

[Zurück](#)[Anfang](#)[Vor](#)

swf_actionplay

[Hoch](#)

swf_actiontogglequality

swf_actiontogglequality

(PHP4 >= 4.0RC2)

swf_actiontogglequality -- Toggle between low and high quality

Description

```
void swf_actiontogglequality (void);
```

Toggle the flash movie between high and low quality.

[Zurück](#)[Anfang](#)[Vor](#)

swf_actionstop

[Hoch](#)

swf_actionwaitforframe

swf_actionwaitforframe

(PHP4 >= 4.0RC2)

swf_actionwaitforframe -- Skip actions if a frame has not been loaded

Description

```
void swf_actionwaitforframe (int framenummer, int skipcount)
```

The **swf_actionWaitForFrame()** function will check to see if the frame, specified by the *framenummer* 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_actionsettarget

(PHP4 >= 4.0RC2)

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_actiongotolabel

(PHP4 >= 4.0RC2)

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_enddoaction

(PHP4 >= 4.0RC2)

swf_enddoaction -- End the current action

Description

```
void swf_enddoaction (void);
```

Ends the current action started by the [swf_startdoaction\(\)](#) function.

swf_defineline

(PHP4 >= 4.0RC2)

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_definerect

(PHP4 >= 4.0RC2)

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_definepoly

(PHP4 >= 4.0.0)

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_startshape

(PHP4 >= 4.0RC2)

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_shapelinesolid

(PHP4 >= 4.0RC2)

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 0.0 is given as a width then no lines are drawn.

swf_shapefilloff

(PHP4 >= 4.0RC2)

swf_shapefilloff -- Turns off filling

Description

```
void swf_shapefilloff (void);
```

The **swf_shapeFillOff()** function turns off filling for the current shape.

[Zurück](#)[Anfang](#)[Vor](#)

swf_shapelinesolid

[Hoch](#)

swf_shapefillsolid

swf_shapefillsolid

(PHP4 >= 4.0RC2)

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_shapefillbitmaptile

(PHP4 >= 4.0RC2)

swf_shapefillbitmaptile -- 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

(PHP4 >= 4.0RC2)

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_shapemoveto

(PHP4 >= 4.0RC2)

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_shapelineto

(PHP4 >= 4.0RC2)

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.

[Zurück](#)[Anfang](#)[Vor](#)

swf_shapemoveto

[Hoch](#)

swf_shapecurveto

swf_shapecurveto

(PHP4 >= 4.0RC2)

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 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_shapecurveto3

(PHP4 >= 4.0RC2)

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_shapearc

(PHP4 >= 4.0RC2)

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_endshape

(PHP4 >= 4.0RC2)

swf_endshape -- Completes the definition of the current shape

Description

```
void swf_endshape (void);
```

The **swf_endshape()** completes the definition of the current shape.

[Zurück](#)[Anfang](#)[Vor](#)

swf_shapearc

[Hoch](#)

swf_definefont

swf_definefont

(PHP4 >= 4.0RC2)

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_setfont

(PHP4 >= 4.0RC2)

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.

[Zurück](#)[Anfang](#)[Vor](#)

swf_definefont

[Hoch](#)

swf_fontsize

swf_fontsize

(PHP4 >= 4.0RC2)

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

(PHP4 >= 4.0RC2)

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 forward slant, negative values create a negative slant.

swf_fonttracking

(PHP4 >= 4.0RC2)

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_getfontinfo

(PHP4 >= 4.0RC2)

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_definetext

(PHP4 >= 4.0RC2)

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_textwidth

(PHP4 >= 4.0RC2)

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_definebitmap

(PHP4 >= 4.0RC2)

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_getbitmapinfo

(PHP4 >= 4.0RC2)

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_startsymbol

(PHP4 >= 4.0RC2)

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_endsymbol

(PHP4 >= 4.0RC2)

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_startbutton

(PHP4 >= 4.0RC2)

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_addbuttonrecord

(PHP4 >= 4.0RC2)

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.

Beispiel 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_oncondition

(PHP4 >= 4.0RC2)

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)

[Zurück](#)

swf_addbuttonrecord

[Anfang](#)

[Hoch](#)

[Vor](#)

swf_endbutton

swf_endbutton

(PHP4 >= 4.0RC2)

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_viewport

(PHP4 >= 4.0RC2)

swf_viewport -- Select an area for future drawing

Description

void **swf_viewport** (double xmin, double xmax, double ymin, double 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.

swf_ortho

(PHP4 >= 4.0.1)

swf_ortho -- Defines an orthographic mapping of user coordinates onto the current viewport

Description

```
void swf_ortho (double xmin, double xmax, double ymin, double ymax,  
double zmin, double zmax)
```

The **swf_ortho()** function defines a orthographic mapping of user coordinates onto the current viewport.

swf_ortho2

(PHP4 >= 4.0RC2)

swf_ortho2 -- Defines 2D orthographic mapping of user coordinates onto the current viewport

Description

void **swf_ortho2** (double xmin, double xmax, double ymin, double 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.

[Zurück](#)[Anfang](#)[Vor](#)

swf_ortho

[Hoch](#)

swf_perspective

swf_perspective

(PHP4 >= 4.0RC2)

swf_perspective -- Define a perspective projection transformation

Description

void **swf_perspective** (double fovy, double aspect, double near, double 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.

Anmerkung: 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_polarview

(PHP4 >= 4.0RC2)

swf_polarview -- Define the viewer's position with polar coordinates

Description

void **swf_polarview** (double dist, double azimuth, double incidence, double 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_lookat

(PHP4 >= 4.0RC2)

swf_lookat -- Define a viewing transformation

Description

```
void swf_lookat (double view_x, double view_y, double view_z, double  
reference_x, double reference_y, double reference_z, double 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_pushmatrix

(PHP4 >= 4.0RC2)

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_popmatrix

(PHP4 >= 4.0RC2)

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_scale

(PHP4 >= 4.0RC2)

swf_scale -- Scale the current transformation

Description

void **swf_scale** (double x, double y, double 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_translate

(PHP4 >= 4.0RC2)

swf_translate -- Translate the current transformations

Description

void **swf_translate** (double x, double y, double z)

The **swf_translate()** function translates the current transformation by the x, y, and z values given.

swf_rotate

(PHP4 >= 4.0RC2)

swf_rotate -- Rotate the current transformation

Description

void **swf_rotate** (double 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_posround

(PHP4 >= 4.0RC2)

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.

LVI. SNMP functions

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.

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
```

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.

Inhaltsverzeichnis

[snmpget](#) Fetch an SNMP object

[snmpset](#) Set an SNMP object

[snmpwalk](#) Fetch all the SNMP objects from an agent

[snmpwalkoid](#) Query for a tree of information about a network entity

[snmp_get_quick_print](#) Fetch the current value of the UCD library's quick_print setting

[snmp_set_quick_print](#) Set the value of quick_print within the UCD SNMP library.

snmpget

(PHP3 , PHP4)

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");
```

snmpset

(PHP3 >= 3.0.12, PHP4 >= 4.0b2)

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

(PHP3 , PHP4)

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

(PHP3 >= 3.0.8, PHP4 >= 4.0b2)

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";  
}
```

snmp_get_quick_print

(PHP3 >= 3.0.8, PHP4 >= 4.0b2)

snmp_get_quick_print -- Fetch the current value of the UCD library's quick_print setting

Description

boolean **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 on, 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

(PHP3 >= 3.0.8, PHP4 >= 4.0b2)

snmp_set_quick_print -- Set the value of quick_print within the UCD SNMP library.

Description

`void snmp_set_quick_print (boolean 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 than 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.

LVII. String-Funktionen

Mit diesen Funktionen werden Strings (Zeichenketten) auf verschiedene Arten verändert. Spezielle Anwendungen können in den Abschnitten zu regulären Ausdrücken (regular expression) und der URL-Handhabung gefunden werden.

Inhaltsverzeichnis

[AddCSlashes](#) Stellt Zeichen eines Strings ein "\" voran (wie in C)

[AddSlashes](#) Stellt bestimmten Zeichen eines Strings einen Backslash "\" (Rückstrich) voran

[bin2hex](#) Wandelt Binär-Daten in ihre hexadezimale Entsprechung um

[Chop](#) Entfernt führende Leerzeichen

[Chr](#) Gibt ein einzelnes Zeichen zurück

[chunk_split](#) Zerlegt einen String in Teile gleicher Länge

[convert_cyr_string](#) Setzt Strings von einem kyrillischen Zeichensatz in einen anderen um

[count_chars](#) Gibt Informationen über die in einem String enthaltenen Zeichen zurück

[crypt](#) DES-Verschlüsselung eines Strings

[echo](#) Gibt einen oder mehrere Strings aus

[explode](#) Zerteilt einen String anhand eines Trennzeichens

[flush](#) Der Ausgabe-Puffer wird geleert und alle Zeichen ausgegeben

[get_html_translation_table](#) Gibt die Übersetzungs-Tabelle zurück, die von [htmlspecialchars\(\)](#) und [htmlspecialchars_decode\(\)](#) benutzt wird

[get_meta_tags](#) Liest die Attribut-Inhalte aller Meta-Tags aus und gibt diese als Array zurück

[htmlentities](#) Wandelt alle Sonderzeichen in entsprechende HTML-Codes um

[htmlspecialchars](#) Diese Funktion wandelt Sonderzeichen in HTML-Codes um

[implode](#) Verbindet Array-Elemente zu einem String

[join](#) Join verbindet Array-Elemente zu einem String

[levenshtein](#) Berechnet den Levenshtein-Unterschied zweier Strings

[ltrim](#) Entfernt führende Leerraum eines Strings

[md5](#) Errechnet den MD5-Code eines Strings

[Metaphone](#) Berechnet den metaphone-Schlüssel eines Strings

[nl2br](#) Konvertiert Neue-Zeile-Textanweisungen in HTML-Zeilenumbrüche

[ob_start](#) Einschalten der Pufferung von Ausgaben

- [ob_get_contents](#) Gibt den Inhalt des Ausgabe-Puffers zurück
- [ob_end_flush](#) Leert (schickt/sendet) den Ausgabe-Puffer und deaktiviert die Ausgabe-Pufferung
- [ob_end_clean](#) Bereinigt (löscht) den Ausgabe-Puffer und deaktiviert die Ausgabe-Pufferung
- [ob_implicit_flush](#) Schaltet die implizite Ausgabe ein bzw. aus
- [Ord](#) Gibt den ASCII-Wert eines Zeichens zurück
- [parse_str](#) Überträgt einen String in Variable
- [print](#) Ausgabe eines Strings
- [printf](#) Gibt einen formatierten String aus
- [quoted_printable_decode](#) Konvertiert einen "quoted-printable"-String in einen 8-Bit-String
- [QuoteMeta](#) Quoten von Meta-Zeichen
- [rawurldecode](#) Dekodierung von URL-kodierten Strings
- [rawurlencode](#) URL-Kodierung nach RFC 1738
- [setlocale](#) Setzt lokale Informationen
- [similar_text](#) Berechnet die Ähnlichkeit zweier Zeichenketten
- [soundex](#) Berechnet die Laut-Ähnlichkeit eines Strings
- [sprintf](#) Gibt einen formatierten String zurück
- [strcasecmp](#) Binärer Vergleich von Zeichenketten
- [strchr](#) Findet das erste Vorkommen eines Zeichens
- [strcmp](#) Binärer Vergleich zweier Strings
- [strcspn](#) Ermittelt die Anzahl der nicht übereinstimmenden Zeichen
- [strip_tags](#) Entfernt HTML- und PHP-Tags aus einem String
- [StripCSlashes](#) Umkehrung / Gegenstück zu addslashes
- [StripSlashes](#) Umkehrung / Gegenstück zu addslashes
- [stristr](#) Wie [strstr\(\)](#) ohne Rücksicht auf Groß- bzw. Kleinschreibung
- [strlen](#) Ermitteln der String-Länge
- [str_pad](#) Erweitert einen String auf eine bestimmte Länge unter Verwendung eines anderen Strings
- [strpos](#) Ermitteln des ersten Vorkommens innerhalb eines Strings
- [strrchr](#) Findet das letzte Vorkommen eines Zeichens innerhalb einer Zeichenkette
- [str_repeat](#) Wiederholung eine String-Ausgabe
- [strrev](#) Umdrehen eines Strings
- [strrpos](#) Findet letztes Vorkommen eines Zeichens in einem String
- [strspn](#) Ermittelt die Länge der übereinstimmenden Zeichen
- [strstr](#) Findet das erste Vorkommen eines Strings

[strtok](#) Zerlegt einen String

[strtolower](#) Setzt einen String in Kleinbuchstaben um

[strtoupper](#) Setzt einen String in Großbuchstaben um

[str_replace](#) Ersetzt alle Vorkommen eines Strings in einem anderen String

[strtr](#) Tauscht bestimmte Zeichen aus

[substr](#) Gibt einen Teilstring zurück

[substr_count](#) Ermittelt die Häufigkeit des Vorkommens einer Zeichenkette in einem String

[substr_replace](#) Ersetzt Text in einer Zeichenkette

[trim](#) Entfernt überflüssige Zeichen ("whitespace" am Anfang und Ende eines Strings)

[ucfirst](#) Setzt das erste Zeichen eines Strings als Großbuchstabe um

[ucwords](#) Setzt die ersten Buchstaben aller Wörter eines Strings in die entsprechenden Großbuchstaben um

[Zurück](#)

[Anfang](#)

[Vor](#)

snmp_set_quick_print

[Hoch](#)

AddCSlashes

AddCSlashes

(unknown)

AddCSlashes -- Stellt Zeichen eines Strings ein "\" voran (wie in C)

Beschreibung:

string **addslashes** (string str, string charlist)

Es wird ein String zurück gegeben, in dem jedem Zeichen, das in *charlist* steht, ein "\" voran gestellt wurde. Teile wie \n, \r etc. werden wie in C übergangen, ASCII-Zeichen kleiner als 32 und grösser als 126 werden in ihren oktal-Wert umgesetzt. Seien sie bei der Umsetzung von alphanumerischen Zeichen vorsichtig. Sie können in *charlist* einen Bereich wie "\0..\37" angeben. Dadurch wird allen ASCII-Zeichen von 0 bis 31 ein "\" voran gestellt.

Beispiel 1 Addslashes() Beispiel:

```
$escaped = addslashes ($not_escaped, "\0..\37!@\\177..\377");
```

Anmerkung: Diese Funktion ist ab Version PHP4b3-dev verfügbar.

Siehe auch [stripslashes\(\)](#), [stripslashes\(\)](#), [htmlspecialchars\(\)](#), [htmlspecialchars\(\)](#) und [quotemeta\(\)](#).

AddSlashes

(unknown)

AddSlashes -- Stellt bestimmten Zeichen eines Strings einen Backslash "\" (Rückstrich) voran

Beschreibung:

string **addslashes** (string str)

Gibt einen String (Zeichenkette) zurück, in dem bestimmten Zeichen ein Backslash "\" voran gestellt wurde. Diese Funktion ist z.B. für Datenbankabfragen wichtig. Die behandelten Zeichen sind der einfache und der doppelte Anführungsstrich (' und "), der Rückstrich (\) selbst sowie NUL (das Null-Byte).

Siehe auch [stripslashes\(\)](#), [htmlspecialchars\(\)](#) und [quotemeta\(\)](#).

bin2hex

(PHP3 >= 3.0.9, PHP4)

bin2hex -- Wandelt Binär-Daten in ihre hexadezimale Entsprechung um

Beschreibung:

string **bin2hex** (string *str*)

Es wird ein ASCII-String mit der hexadezimalen Wiedergabe des Parameters *str* zurück gegeben. Die Umwandlung geschieht byteweise mit dem höchsten Anteil (4 Bit, "high-nibble") zuerst.

[Zurück](#)

AddSlashes

[Anfang](#)[Hoch](#)[Vor](#)

Chop

Chop

(unknown)

Chop -- Entfernt führende Leerzeichen

Beschreibung:

string **chop** (string *str*)

Gibt die in *str* enthaltene Zeichenkette ohne evtl. vorhandene führende Leerzeichen zurück. In dem String enthaltene Zeilenvorschübe werden entfernt.

Beispiel 1 Chop() Beispiel:

```
$string_mit_NL = "          10 führende Leerzeichen mit \n\n neuer Zeile";  
echo chop($string_mit_NL);
```

Die Ausgabe lautet:
10 führende Leerzeichen mit neuer Zeile

Siehe auch [trim\(\)](#).

Chr

(unknown)

Chr -- Gibt ein einzelnes Zeichen zurück

Beschreibung:

string **chr** (int *ascii*)

Gibt einen 1-Zeichen-String zurück, der dem mit *ascii* angegebenen Zeichen entspricht.

Beispiel 1 Chr() Beispiel:

```
$str .= chr (27); /* fügt an das Ende von $str ein Escape-Zeichen an */  
/* Oft wird diese Funktionalität gebraucht */  
$str = sprintf ("Dieser String endet mit einem Escape: %c", 27);
```

Diese Funktion ergänzt [ord\(\)](#). Siehe auch [sprintf\(\)](#) mit dem Formatierungs-String %C.

chunk_split

(PHP3 >= 3.0.6, PHP4)

chunk_split -- Zerlegt einen String in Teile gleicher Länge

Beschreibung:

string **chunk_split** (string string [, int chunklen [, string end]])

Kann für die Aufteilung eines Strings in kleinere Stücke verwendet werden. Dies ist z.B. hilfreich bei der Umwandlung von [base64_encode](#)-Ausgaben, die der RFC 2045 entsprechen müssen. Diese Funktion fügt nach per *chunklen* angegebene Zeichen (Vorgabe: alle 76 Zeichen) den mittels *end* angegebenen String (Vorgabe ist hierfür "\r\n") ein. Die Funktion gibt den neuen String zurück. Die Original-Zeichenkette bleibt unverändert.

Beispiel 1 Chunk_split() Beispiel:

```
# Formatiere den String $daten in die RFC 2045 Schreibweise  
$neuer_string = chunk_split (base64_encode($daten));
```

Diese Funktion arbeitet wesentlich schneller als [ereg_replace\(\)](#).

Anmerkung: **Chunk_split()** ist ab Version 3.0.6 vorhanden.

convert_cyr_string

(PHP3 >= 3.0.6, PHP4)

convert_cyr_string -- Setzt Strings von einem kyrillischen Zeichensatz in einen anderen um

Beschreibung:

string **convert_cyr_string** (string str, string from, string to)

Diese Funktion konvertiert den angegebenen String von einem kyrillischen Zeichensatz in einen anderen. Die Argumente *from* und *to* sind einzelne Zeichen die den Ausgangs- und Zielzeichensatz definieren. Die unterstützten Zeichensätze sind:

- k - koi8-r
- w - windows-1251
- i - iso8859-5
- a - x-cp866
- d - x-cp866
- m - x-mac-cyrillic

count_chars

(PHP4 >= 4.0b4)

count_chars -- Gibt Informationen über die in einem String enthaltenen Zeichen zurück

Beschreibung:

mixed **count_chars** (string string [, mode])

Zählt die Häufigkeit des Vorkommens jedes einzelnen Byte-Wertes (0..255) in *string* und gibt sie auf verschiedene Arten zurück. Der optionale Parameter *mode* ist voreingestellt auf 0. In Abhängigkeit des *mode*-Wertes gibt **count_chars()** das folgende zurück:

- 0 - ein Array mit Byte-Werten für die Häufigkeit (auch 0-Vorkommen) jedes Zeichens.
- 1 - wie 0, aber nur Häufigkeiten grösser 0.
- 2 - wie 0, aber nur Häufigkeiten gleich 0.
- 3 - ein String der überhaupt vorkommenden Zeichen enthält.
- 4 - ein String der alle nicht vorkommenden Zeichen enthält.

Anmerkung: Diese Funktion existiert ab PHP 4.0.

crypt

(PHP3 , PHP4)

crypt -- DES-Verschlüsselung eines Strings

Beschreibung:

string **crypt** (string str [, string salt])

Crypt() verschlüsselt einen String unter Verwendung der Standard-DES-Verschlüsselungs-Methode von Unix. Die zu übergebenden Argumente sind der zu verschlüsselnde String und optional ein 2-Zeichen-Salt-String, der die Schlüsselbasis bildet. Lesen sie die Man-Pages ihres Unix-Systems für mehr Informationen zu den Crypt-Funktionen.

Ist kein Salt-Argument angegeben, wird es von PHP nach dem Zufalls-Prinzip erzeugt.

Einige Betriebssysteme unterstützen mehr als eine Methode zur Verschlüsselung. So wird manchmal der DES- durch einen MD5-Algorithmus ersetzt. Der verwendete Schlüssel wird durch das Salt-Argument bestimmt. Zur Installations-Zeit untersucht PHP die vorhandenen Möglichkeiten und wird, abhängig vom Ergebnis dieser Prüfung, auch andere Schlüssel-Typen zulassen. Wird kein Salt unterstützt, erzeugt PHP per Voreinstellung einen 2-Zeichen DES-Salt, es sei denn auf ihrem System existiert MD5. Dann wird PHP einen zufälligen MD5-Salt generieren. PHP erzeugt eine Konstante namens CRYPT_SALT_LENGTH. Diese besagt, ob ihr System ein reguläres 2-Zeichen Salt oder das längere 12-Zeichen MD5-Salt unterstützt.

Die Standard-DES-Verschlüsselung **crypt()** enthält das Salt als erste 2 Zeichen der Ausgabe.

Auf Systemen, wo die crypt()-Funktion mehrere Verschlüsselungen unterstützt, werden die folgenden Konstanten auf 0 oder 1 gesetzt, je nachdem, ob der entsprechende Typ verfügbar ist:

- CRYPT_STD_DES - Standard DES-Schlüssel mit 2-Zeichen Salt
- CRYPT_EXT_DES - Erweiterter DES-Schlüssel mit einem 9-Zeichen Salt
- CRYPT_MD5 - MD5-Schlüssel mit 12-Zeichen Salt, beginnend mit \$1\$
- CRYPT_BLOWFISH - Erweiterter DES-Schlüssel, 16-Zeichen Salt, beginnend mit \$2\$

Der verschlüsselte String kann nicht entschlüsselt werden, da **crypt()** eine Einweg-Verschlüsselung ist.

echo

(unknown)

echo -- Gibt einen oder mehrere Strings aus

Beschreibung:

echo (string arg1, string [argn]...)

Gibt alle als Parameter angegebenen Strings aus.

Echo() ist keine Funktion, sonder ein Sprach-Konstrukt. Deshalb brauchen sie keine Klammern zu verwenden.

Beispiel 1 Echo()-Beispiel:

```
echo "Hallo Welt";

echo "Dieses
Beispiel
umfasst
mehrere
Zeilen.";
```

Anmerkung: Falls sie mehr als einen auszugebenden Parameter angeben, dürfen sie diese nicht in Klammern setzen.

Siehe auch: [print\(\)](#), [printf\(\)](#) und [flush\(\)](#).

explode

(PHP3 , PHP4)

explode -- Zerteilt einen String anhand eines Trennzeichens

Beschreibung:

array **explode** (string separator, string string [, int limit])

Gibt ein Array aus Strings zurück. Jeder String ist ein Teil von *string*. Die Trennung erfolgt an der mit *separator* angegebenen Zeichenkette (ein oder mehrere Zeichen). Ist der Parameter *limit* angegeben, werden maximal *limit* Teile zurück gegeben. Das letzte Element enthält dann den ganzen Rest von *string*.

Beispiel 1 Explode()-Beispiel:

```
$pizza = "Teil1 Teil2 Teil3 Teil4 Teil5 Teil6";  
$pieces = explode (" ", $pizza);
```

Siehe auch [split\(\)](#) und [implode\(\)](#).

flush

(PHP3 , PHP4)

flush -- Der Ausgabe-Puffer wird geleert und alle Zeichen ausgegeben

Beschreibung:

```
void flush(void);
```

Unabhängig davon, wie PHP läuft (CGI oder WEB-Server), wird der gesamte Ausgabe-Puffer geleert. Dabei wird versucht den ganzen Inhalt an den Browser des Benutzers zu schicken.

[Zurück](#)

explode

[Anfang](#)[Hoch](#)[Vor](#)

get_html_translation_table

get_html_translation_table

(PHP4 >= 4.0b4)

`get_html_translation_table` -- Gibt die Übersetzungs-Tabelle zurück, die von [htmlspecialchars\(\)](#) und [htmlentities\(\)](#) benutzt wird

Beschreibung:

string `get_html_translation_table` (int `table`)

`Get_html_translation_table()` ermittelt die intern benutzte Übersetzungs-Tabelle, die von den Funktionen [htmlspecialchars\(\)](#) und [htmlentities\(\)](#) benutzt wird. Es gibt zwei neue Parameter (*HTML_ENTITIES* und *HTML_SPECIALCHARS*), die es ihnen ermöglichen, die gewünschte Tabelle auszuwählen.

Beispiel 1 Beispiel zur Übersetzungs-Tabelle

```
$trans = get_html_translation_table (HTML_ENTITIES);  
$str = "Hallo & <Frau> & Krämer";  
$encoded = strtr ($str, $trans);
```

Die `$encoded`-Variable wird jetzt "Hallo & <Frau> & Krämer". enthalten.

Mittels [array_flip\(\)](#) können sie die Richtung der Übersetzung umkehren.

```
$trans = array_flip ($trans);  
$original = strtr ($str, $trans);
```

Der Inhalt von `$original` wird "Hallo & <Frau> & Krämer" sein.

Anmerkung: Diese Funktion wurde in PHP 4.0 neu hinzu gefügt.

Siehe auch [htmlspecialchars\(\)](#), [htmlentities\(\)](#), [strtr\(\)](#) und [array_flip\(\)](#).

htmlspecialchars

(PHP3 , PHP4)

htmlspecialchars -- Diese Funktion wandelt Sonderzeichen in HTML-Codes um

Beschreibung:

string **htmlspecialchars** (string string)

Bestimmte Zeichen haben eine spezielle Bedeutung in HTML und sollten im HTML-Code dargestellt werden, um ihre Bedeutung zu behalten. Diese Funktion liefert einen String zurück, der diese Bedingungen erfüllt.

Sinnvoll beim Schutz von benutzerabhängigen Texteingaben (z.B. in Gästebuch-Formularen).

Gegenwärtig werden folgende Umwandlungen vorgenommen:

- '&' (kaufmännisches und) wird zu '&'
- '"' (doppelte Anführungszeichen) werden zu '"'
- '<' (kleiner als) wird zu '<'
- '>' (größer als) wird zu '>'

Beachten sie, dass diese Funktion nur die o.a. Umwandlungen durchführt. Zur vollständigen Code-Umwandlung siehe [htmlentities\(\)](#).

Siehe auch [htmlentities\(\)](#) und [nl2br\(\)](#).

htmlentities

(PHP3 , PHP4)

htmlentities -- Wandelt alle Sonderzeichen in entsprechende HTML-Codes um

Beschreibung:

string **htmlentities** (string string)

Diese Funktion ist identisch zu **htmlspecialchars()**, ausser dass alle Zeichen, die HTML-Code-Entsprechungen haben in diese Codes umgewandelt werden.

Gegenwärtig wird der ISO-8859-1-Zeichensatz verwendet.

Siehe auch [htmlspecialchars\(\)](#) und [nl2br\(\)](#).

get_meta_tags

(PHP3 >= 3.0.4, PHP4)

`get_meta_tags` -- Liest die Attribut-Inhalte aller Meta-Tags aus und gibt diese als Array zurück

Beschreibung:

array **get_meta_tags** (string filename [, int use_include_path])

Öffnet *filename* und untersucht diese Zeile für Zeile auf das Vorkommen von <meta>-Tags der Form

Beispiel 1 Meta-Tags Beispiel:

```
<meta name="author" content="Name">
<meta name="tags" content="PHP3-Dokumentation">
</head> <!-- Ende der Untersuchung -->
```

(Eine auf einem MAC erzeugte Datei kann nicht unter UNIX auf Meta-Tags untersucht werden, da die Zeilenenden anders sind.)

Der Wert der name-Eigenschaft wird Schlüssel, der Wert der content-Eigenschaft wird Wert des zurück gegebenen Arrays. Dadurch können sie gewöhnliche Array-Funktionen nutzen, dieses zu durchlaufen oder einzelne Werte anzusprechen. Sonderzeichen innerhalb der name-Eigenschaft werden durch `'_'` ersetzt, der Rest wird in Kleinbuchstaben umgewandelt.

Ist *use_include_path* auf 1 gesetzt, wird PHP versuchen, die Datei anhand des Standard-Include-Pfades zu öffnen.

implode

(PHP3 , PHP4)

implode -- Verbindet Array-Elemente zu einem String

Beschreibung:

string **implode** (string glue, array pieces)

Gibt einen String zurück, dessen Inhalt allen Array-Elementen in der fortlaufenden Reihenfolge ihres Vorkommens im Array entspricht. Die Stringteile sind durch den im Parameter glue enthaltenen String verbunden.

Beispiel 1 Implode()-Beispiel:

```
$Trennung_durch_Doppelpunkt = implode (":", $array);
```

Siehe auch [explode\(\)](#), [join\(\)](#) und [split\(\)](#).

join

(PHP3 , PHP4)

join -- Join verbindet Array-Elemente zu einem String

Beschreibung:

string **join** (string glue, array pieces)

Join() ist ein Alias von [implode\(\)](#) (die Funktionalität ist also vollkommen identisch).

Siehe auch [explode\(\)](#), [implode\(\)](#) und [split\(\)](#).

levenshtein

(PHP3 CVS only, PHP4 >= 4.0.1)

levenshtein -- Berechnet den Levenshtein-Unterschied zweier Strings

Beschreibung:

int **levenshtein** (string *str1*, string *str2*)

Diese Funktion errechnet die Differenz zwischen den als Argumente übergebenen zwei Strings. Ist einer der Strings länger als die zulässigen 255 Zeichen, wird -1 zurück gegeben.

Die Levenshtein-Differenz ist definiert als die minimale Anzahl an Zeichen, die ersetzt, eingefügt oder gelöscht werden müssen um den *str1* nach *str2* umzusetzen. Die Komplexität des Algorithmus ist $O(m * n)$, wobei n und m der Länge der Strings *str1* und *str2* (viel besser als bei der Funktion [similar_text\(\)](#), wo sie $O(\max(n,m)**3)$ beträgt).

Siehe auch [soundex\(\)](#), [similar_text\(\)](#) und [metaphone\(\)](#).

ltrim

(PHP3 , PHP4)

ltrim -- Entfernt führende Leerraum eines Strings

Beschreibung:

string **ltrim** (string str)

Es wird Leerraum ab Stringanfang entfernt und der um diesen gekürzte String zurück gegeben. Als Leerraum gelten die Zeichen "\n", "\r", "\t", "\v", "\0" und das Leerzeichen.

Siehe auch [chop\(\)](#) und [trim\(\)](#).

md5

(PHP3 , PHP4)

md5 -- Errechnet den MD5-Code eines Strings

Beschreibung:

string **md5** (string *str*)

Berechnet den MD5-Code von *str* unter Verwendung des [RSA Data Security, Inc. MD5 Message-Digest Algorithmus](#).

Metaphone

(unknown)

Metaphone -- Berechnet den metaphone-Schlüssel eines Strings

Beschreibung:

string **metaphone** (string str)

Errechnet den metaphone-Schlüssel von *str*.

Genau wie [soundex\(\)](#) berechnet metaphone den gleichen Schlüssel für ähnlich klingende Wörter. Die Metaphone-Funktion arbeitet genauer als [soundex\(\)](#), da sie die Grundregeln der englischen Aussprache kennt. Die durch metaphone erzeugten Schlüssel sind von variabler Länge.

Der Metaphone-Algorithmus wurde entwickelt von Lawrence Philips <lphilips@verity.com> und ist beschrieben in ["Practical Algorithms for Programmers", Binstock & Rex, Addison Wesley, 1995].

Anmerkung: Diese Funktion ist ab PHP 4.0 verfügbar.

nl2br

(PHP3 , PHP4)

nl2br -- Konvertiert Neue-Zeile-Textanweisungen in HTML-Zeilenumbrüche

Beschreibung:

string **nl2br** (string string)

Gibt *string* mit '
' an Stelle von Neue-Zeile zurück.

Siehe auch [htmlspecialchars\(\)](#) und [htmlentities\(\)](#).

ob_start

(PHP4)

ob_start -- Einschalten der Pufferung von Ausgaben

Beschreibung:

```
voidob_start(void);
```

Diese Funktion schaltet die Pufferung von Ausgaben ein. Solange die Ausgabe-Pufferung aktiv ist, erfolgt keine Ausgabe bei der Skript-Ausführung. Statt dessen werden alle Ausgaben in einem internen Puffer gespeichert.

Der Inhalt diese internen Puffers kann in eine String-Variable kopiert werden (per [ob_get_contents\(\)](#)). Die Ausgabe erfolgt erst bei Aufruf von [ob_end_flush\(\)](#) und sobald der Puffer-Inhalt per [ob_end_clean\(\)](#) verworfen wird.

Siehe auch [ob_get_contents\(\)](#), [ob_end_flush\(\)](#), [ob_end_clean\(\)](#) und [ob_implicit_flush\(\)](#).

ob_get_contents

(PHP4)

ob_get_contents -- Gibt den Inhalt des Ausgabe-Puffers zurück

Beschreibung:

```
string ob_get_contents(void);
```

Damit wird der Inhalt des Ausgabe-Puffers zurück gegeben (oder False, wenn die Ausgabe-Pufferung nicht aktiv ist).

Siehe auch [ob_start\(\)](#), [ob_end_flush\(\)](#) und [ob_end_clean\(\)](#).

ob_end_flush

(PHP4)

ob_end_flush -- Leert (schickt/sendet) den Ausgabe-Puffer und deaktiviert die Ausgabe-Pufferung

Beschreibung:

```
void ob_end_flush(void);
```

Der Inhalt des Ausgabe-Puffers (sofern vorhanden) wird abgeschickt und die Pufferung der Ausgabe deaktiviert. Falls sie mit dem Puffer-Inhalt weiter arbeiten möchten, müssen sie diesen erst per [ob_get_contents\(\)](#) zwischen speichern bevor sie **ob_end_flush()** aufrufen, da dadurch der Puffer geleert wird.

Siehe auch [ob_start\(\)](#), [ob_get_contents\(\)](#) und [ob_end_clean\(\)](#).

ob_end_clean

(PHP4)

ob_end_clean -- Bereinigt (löscht) den Ausgabe-Puffer und deaktiviert die Ausgabe-Pufferung

Beschreibung:

```
void ob_end_clean(void);
```

Diese Funktion verwirft den Inhalt des Ausgabe-Puffers und deaktiviert die Pufferung.

Siehe auch [ob_start\(\)](#) und [ob_end_flush\(\)](#).

ob_implicit_flush

(PHP4 >= 4.0b4)

ob_implicit_flush -- Schaltet die implizite Ausgabe ein bzw. aus

Beschreibung:

void **ob_implicit_flush** ([int *flag*])

Ob_implicit_flush() schaltet die implizite Ausgabe an oder aus (wenn kein *flag* angegeben wurde, wird per default "Ein" angenommen). Die implizite Puffer-Ausgabe erzeugt eine Ausgabe nach jedem Ausgabe-Befehl, so dass keine Extra-Aufrufe von [flush\(\)](#) mehr erforderlich sind.

Das Einschalten der impliziten Ausgabe beendet die Pufferung der Ausgabe. Der aktuelle Puffer-Inhalt wird ausgegeben, so als ob die Funktion [ob_end_flush\(\)](#) aufgerufen wurde.

Siehe auch [flush\(\)](#), [ob_start\(\)](#) und [ob_end_flush\(\)](#).

Ord

(unknown)

Ord -- Gibt den ASCII-Wert eines Zeichens zurück

Beschreibung:

int **ord** (string string)

Es wird der ASCII-Wert des ersten Zeichens von *string* zurück gegeben. Diese Funktion steht in Zusammenhang mit [chr\(\)](#).

Beispiel 1 Ord()-Beispiel:

```
if (ord ($str) == 10) {  
    echo "Das erste Zeichen von \"$str\" ist ein Zeilenvorschub.\n";  
}
```

Siehe auch [chr\(\)](#).

parse_str

(PHP3 , PHP4)

parse_str -- Überträgt einen String in Variable

Beschreibung:

void **parse_str** (string str)

Durchsucht *str* als ob er der Query-String einer URL-Übergabe via GET wäre und gibt die Werte als Werte eines Arrays zurück.

Beispiel 1 Der Gebrauch von parse_str():

```
$str = "erster= nichts & zweiter[]=etwas & zweiter[]=nochwas";
parse_str($str);
echo $erster;          /* gibt "nichts" aus */
echo $zweiter[0];      /* gibt "etwas" aus */
echo $zweiter[1];      /* gibt "nochwas" aus */
```

print

(unknown)

print -- Ausgabe eines Strings

Beschreibung:

print (string arg)

Gibt *arg* aus.

Siehe auch [echo\(\)](#), [printf\(\)](#) und [flush\(\)](#).

printf

(PHP3 , PHP4)

printf -- Gibt einen formatierten String aus

Beschreibung:

int **printf** (string format [, mixed args...])

Erzeugt eine Ausgabe abhängig von *format*. *Format* wird in der Dokumentation zu [sprintf\(\)](#) näher beschrieben.

Siehe auch [print\(\)](#), [sprintf\(\)](#) und [flush\(\)](#).

quoted_printable_decode

(PHP3 >= 3.0.6, PHP4)

quoted_printable_decode -- Konvertiert einen "quoted-printable"-String in einen 8-Bit-String

Beschreibung:

string **quoted_printable_decode** (string str)

Diese Funktion gibt einen 8-Bit-String zurück, der der Konvertierung aus einem "quoted-printable"-String entspricht. Diese Funktion ist ähnlich wie [imap_qprint\(\)](#) bis auf die Tatsache, dass sie kein IMAP-Module erfordert.

QuoteMeta

(unknown)

QuoteMeta -- Quoten von Meta-Zeichen

Beschreibung:

string **quotemeta** (string *str*)

Der Inhalt von *str* wird mit einem Rückstrich ("Backslash", \) vor jedem Vorkommen von

. \ \ + * ? [^] (\$)

versehen ("quoten") und zurück gegeben.

Siehe auch [addslashes\(\)](#), [htmlentities\(\)](#), [htmlspecialchars\(\)](#), [nl2br\(\)](#) und [stripslashes\(\)](#).

rawurldecode

(PHP3 , PHP4)

rawurldecode -- Dekodierung von URL-kodierten Strings

Beschreibung:

string **rawurldecode** (string str)

Gibt einen String zurück, in dem die Abschnitte, die mit einem Prozent-Zeichen (%) beginnen (dem ein 2-stelliger hexadezimal-Wert folgt), ersetzt wurden durch Buchstaben. So wird z.B. die Zeichenkette

```
foo%20bar%40baz
```

umgewandelt in

```
foo bar@baz
```

.

Siehe auch [rawurlencode\(\)](#).

rawurlencode

(PHP3 , PHP4)

rawurlencode -- URL-Kodierung nach RFC 1738

Beschreibung:

string **rawurlencode** (string str)

Gibt eine Zeichenkette zurück, in der alle Sonderzeichen ausser

`-, _ , .`

durch ein Prozent-Zeichen (%), gefolgt von einer 2-stelligen hexadezimalen Zahl, ersetzt worden sind. Dies ist die Kodierung nach RFC 1738 zum Schutz von Zeichen, die als spezielle URL-Trennzeichen zu interpretieren sind oder die vor einer Umwandlung durch Übertragungs-Medien mit Zeichen-Konvertierung (z.B. einige E-Mail-Systeme) bewahrt werden sollen. Falls sie z.B. ein Passwort in eine FTP-URL einbinden wollen:

Beispiel 1 Rawurlencode()-Beispiel 1:

```
echo '<A HREF="ftp://user:', rawurlencode ('foo @+%/'),  
    '@ftp.mein.de/x.txt">';
```

Oder sie wollen Informationen in der Pfadangabe der URL übertragen:

Beispiel 2 Rawurlencode()-Beispiel 2:

```
echo '<A HREF="http://x.de/leitung_list_skript/',  
    rawurlencode ('verkauf und marketing/Hamburg'), '">';
```

Siehe auch [rawurldecode\(\)](#).

setlocale

(PHP3 , PHP4)

setlocale -- Setzt lokale Informationen

Beschreibung:

string **setlocale** (string category, string locale)

Der String *category* enthält die Kategorie der zu ändernden Lokalisitäts-Informationen:

- LC_ALL - alle folgenden Werte
- LC_COLLATE - für String-Vergleiche - z.Zt. noch nicht in PHP implementiert
- LC_CTYPE - betrifft Klassifizierungen und Umwandlung, z.B. bei [strtoupper\(\)](#)
- LC_MONETARY - betrifft localeconv() - z.Zt. noch nicht in PHP implementiert
- LC_NUMERIC - für das Dezimal-Trennzeichen
- LC_TIME - betrifft Zeit und Datums-Formatierungen durch [strftime\(\)](#)

Ist *locale* als Leer-String definiert, also "", werden die Werte an Hand der entsprechenden Umgebungs-Variablen gleichen Namens oder von "LANG" bestimmt.

Ist *locale* Null oder "0", werden keine Einstellungen verändert sondern als Ergebnis zurück gegeben.

Setlocale() gibt die neue Einstellung oder false (wenn die locale-Funktionalität auf der System-Plattform nicht unterstützt wird oder der Kategorie-Wert ungültig ist) zurück. Ein ungültiger Kategorie-Name erzeugt auch noch eine Warnmeldung.

similar_text

(PHP3 >= 3.0.7, PHP4 >= 4.0b2)

similar_text -- Berechnet die Ähnlichkeit zweier Zeichenketten

Beschreibung:

`int similar_text (string first, string second [, double percent])`

Die Berechnung der Ähnlichkeit zweier Strings erfolgt nach der in "Oliver" [1993] beschriebenen Methode. Beachten sie, dass diese Implementierung keinen Stack verwendet wie in Olivers Pseude-Code, dafür aber rekursive Aufrufe, die die Performance erhöhen können (aber nicht müssen). Beachten sie auch, dass die Komplexität dieses Algorithmus $O(N^3)$ beträgt, wobei N die Länge der längsten Zeichenkette ist.

Falls sie als Referenz ein drittes Argument angeben, wird **similar_text()** ihnen die Ähnlichkeit als Prozentwert errechnen. Es wird die Anzahl der in beiden Strings übereinstimmenden Zeichen zurück gegeben.

soundex

(PHP3 , PHP4)

soundex -- Berechnet die Laut-Ähnlichkeit eines Strings

Beschreibung:

string **soundex** (string str)

Errechnet den Wert der Laut-Ähnlichkeit von *str*.

Soundex-Werte haben die Eigenschaft, dass ähnlich ausgesprochene Wörter den gleichen Soundex-Wert erzeugen. Dies kann zur Suche in Datenbanken verwendet werden, wenn sie zwar die Aussprache aber nicht die genaue Schreibweise kennen. **Soundex()** liefert einen String aus 4 Zeichen, beginnend mit einem Buchstaben zurück.

Diese besondere **soundex()**-Funktion ist u.a. beschrieben in "The Art Of Computer Programming, vol. 3: Sorting And Searching", von Donald Knuth, Addison-Wesley-Verlag (1973), Seiten 391 bis 392.

Beispiel 1 Soundex-Beispiel:

```
soundex ("Euler") == soundex ("Ellery") == 'E460';
soundex ("Gauss") == soundex ("Ghosh") == 'G200';
soundex ("Knuth") == soundex ("Kant") == 'H416';
soundex ("Lloyd") == soundex ("Ladd") == 'L300';
soundex ("Lukasiewicz") == soundex ("Lissajous") == 'L222';
```

sprintf

(PHP3 , PHP4)

sprintf -- Gibt einen formatierten String zurück

Beschreibung:

string **sprintf** (string format [, mixed args...])

Gibt einen an Hand des Formatierungs-Strings *format* gebildeten String zurück.

Der Formatierungs-String wird aus keiner oder mehreren Anweisungen gebildet: Gewöhnliche Zeichen (ausser %), die direkt in das Ergebnis übertragen werden und *Umsetzungs-Anweisungen*. Jede dieser Anweisungen resultiert ihre eigene Auswirkung. Diese Anweisungen gelten sowohl für die **sprintf()**- als auch für die [printf\(\)](#)-Funktion.

Jede der Formatierungs-Anweisungen besteht aus den folgenden Elementen (in alphabetischer Reihenfolge):

1. Einem optionalen *Füll-Zeichen*, das den String bis rechts auffüllt. Dies kann ein Leerzeichen oder eine 0 (das Null-Zeichen) sein. Als Füllzeichen voreingestellt ist das Leerzeichen. Andere Füllzeichen können durch Voranstellung eines einfachen Anführungszeichens (') definiert werden. Beachten sie die folgenden Beispiele.
2. Einer optionalen Angabe der *Ausrichtung*, ob die Ausgabe also nach links oder nach rechts (default) ausgerichtet werden soll. Um den Text links auszurichten müssen sie das Zeichen - angeben.
3. Optional einem Wert für die *Längen-Angabe*, also der Wert für die mindestenst zurückzugebenden Zeichen.
4. Optional einem Wert für die Anzahl der auszugebenden *Nachkomma-Stellen*. Dieser Wert hat nur Einfluss auf Werte vom Typ "double" (eine weitere hilfreiche Funktion für die Formatierung von Zahlen ist [number_format\(\)](#)).
5. Einer *Typ-Angabe* welche den Type enthält, als die das Argument angesehen werden soll. Mögliche Angaben sind:
 - % - ein Prozent-Zeichen ("%"). Es ist kein Argument erforderlich.
 - b - das Argument wird als Integer angesehen und als Binär-Wert ausgegeben.
 - c - das Argument wird als Integer angesehen und das entsprechende ASCII-Zeichen wird ausgegeben.

d - das Argument wird als Integer angesehen und ein Dezimalwert ausgegeben.

f - das Argument wird als "double" angesehen und eine Fließkomma-Zahl ausgegeben.

o - das Argument wird als Integer angesehen und als Oktalwert ausgegeben.

s - das Argument wird als String angesehen und als solcher ausgegeben.

x - das Argument wird als Integer angesehen und als Hexadezimal-Wert ausgegeben (mit Kleinbuchstaben).

X - das Argument wird als Integer angesehen und als Hexadezimal-Wert ausgegeben (mit Großbuchstaben).

Siehe auch: [printf\(\)](#) und [number_format\(\)](#).

Beispiele:

Beispiel 1 Sprintf(): Null-gefüllte Integer-Werte

```
$isodatum = sprintf ("%04d-%02d-%02d", $jahr, $monat, $tag);
```

Beispiel 2 Sprintf(): Integer- zu Binärwert

```
$wert = 155;  
$binaer = sprintf ("%b", $wert);  
echo ("<br> $binaer <br>");  
// gibt 10011011 aus
```

Beispiel 3 Sprintf(): Währungsformatierung

```
$money1 = 68.75;  
$money2 = 54.35;  
$money = $money1 + $money2;  
// echo $money wird "123.1" ausgegeben  
$formatiert = sprintf ("%01.2f", $money);  
// echo $formatiert wird "123.10" ausgegeben
```

[Zurück](#)

soundex

[Anfang](#)

[Hoch](#)

[Vor](#)

strcasecmp

strcasecmp

(PHP3 >= 3.0.2, PHP4)

strcasecmp -- Binärer Vergleich von Zeichenketten

Beschreibung:

`int strcasecmp (string str1, string str2)`

Ist *str1* kleiner als *str2* wird < 0, ist *str1* grösser als *str2* wird > 0 und bei Gleichheit wird 0 zurück gegeben.

Beispiel 1 Strcasecmp()-Beispiel:

```
$var1 = "Hallo";  
$var2 = "hallo";  
if (!strcasecmp($var1, $var2) ) {  
    echo '$var1 ist wie $var2 bei schreibweise-unabhängigem Vergleich';  
}
```

Beachten sie, dass nicht zwischen Groß- und Kleinschreibung unterschieden wird.

Siehe auch [ereg\(\)](#), [strcmp\(\)](#), [substr\(\)](#), [stristr\(\)](#) und [strstr\(\)](#).

strchr

(PHP3 , PHP4)

strchr -- Findet das erste Vorkommen eines Zeichens

Beschreibung:

string **strchr** (string haystack, string needle)

Diese Funktion ist ein Alias zu [strstr\(\)](#) und funktioniert genau wie dort beschrieben.

strcmp

(PHP3 , PHP4)

strcmp -- Binärer Vergleich zweier Strings

Beschreibung:

```
int strcmp (string str1, string str2)
```

Ist *str1* kleiner als *str2* wird < 0 , ist *str1* grösser als *str2* wird > 0 und bei Gleichheit wird 0 zurück gegeben.

Beachten sie, dass zwischen Groß- und Kleinschreibung unterschieden wird.

Siehe auch [ereg\(\)](#), [strcasecmp\(\)](#), [substr\(\)](#), [stristr\(\)](#) und [strstr\(\)](#).

strcspn

(PHP3 >= 3.0.3, PHP4)

strcspn -- Ermittelt die Anzahl der nicht übereinstimmenden Zeichen

Beschreibung:

int strcspn (string *str1*, string *str2*)

Die Länge des Strings *str1* bis zum ersten Erscheinen eines Buchstabens (aus *str1*) in *str2* wird zurück gegeben.

Beispiel 1 Strcspn()-Beispiel:

```
$s1 = "dasrktuvs";  
$s2 = "iklp";           // "k" ist der "Kasus-Knacktus"  
$wert = strcspn ($s1, $s2);  
echo "<br> $wert <br>"; // gibt also 4 zurück
```

Siehe auch [strspn\(\)](#).

strip_tags

(PHP3 >= 3.0.8, PHP4 >= 4.0b2)

strip_tags -- Entfernt HTML- und PHP-Tags aus einem String

Beschreibung:

string **strip_tags** (string *str* [, string *allowable_tags*])

Diese Funktion versucht, alle HTML- und PHP-Tags aus *str* zu entfernen. Bei unvollständigen oder sinnlosen Tags werden Fehler ausgegeben. Es wird die gleiche Art der Tag-Engine wie bei der [fgetss\(\)](#)-Funktion benutzt.

Sie können den optionalen Parameter *allowable_tags* benutzen, um nicht zu entfernende Tags zu definieren.

Anmerkung: *allowable_tags* steht ab PHP 3.0.13 bzw. PHP4B3 zur Verfügung.

Bad Request

Your browser sent a request that this server could not understand.

Invalid URI in request HTTP/1.1 200 OK

Apache/1.3.6 Server at xweb Port 80

StripSlashes

(unknown)

StripSlashes -- Umkehrung / Gegenstück zu addslashes

Beschreibung:

string **stripslashes** (string str)

Gibt einen String ohne evtl. vorhandene "\" (Backslash) zurück (\ ' wird zu ' usw.). Doppelte Rückstriche ("\\") werden zu "\" umgesetzt.

Siehe auch [addslashes\(\)](#).

stristr

(PHP3 >= 3.0.6, PHP4)

stristr -- Wie [strstr\(\)](#) ohne Rücksicht auf Groß- bzw. Kleinschreibung

Beschreibung:

string **stristr** (string haystack, string needle)

Findet alle Vorkommen von *needle* innerhalb von *haystack* vom Anfang bis zum Ende (ohne Rücksicht auf die Schreibweise beider Parameter).

Wird *needle* nicht gefunden, wird false zurück gegeben.

Ist *needle* kein String, wird er in einen Integer-Wert umgesetzt. Dies ASCII-Entsprechung dieses Wertes wird dann als zu suchendes Zeichen angesehen.

Siehe auch [strchr\(\)](#), [strrchr\(\)](#), [substr\(\)](#) und [ereg\(\)](#).

strstr

(PHP3 , PHP4)

strstr -- Findet das erste Vorkommen eines Strings

Beschreibung:

string **strstr** (string haystack, string needle)

Es wird alles von *haystack* zurück gegeben ab der ersten Fundstelle von *needle* bis zum Ende.

Falls *needle* nicht gefunden wird, ist das Ergebnis false.

Ist *needle* kein String, wird er zu einem Integer-Wert umgesetzt und als das diesem Wert entsprechende ASCII-Zeichen angesehen.

Beispiel 1 Strstr()-Beispiel:

```
$email = 'sterling@designmultimedia.com';  
$domain = strstr ($email, '@');  
print $domain; // gibt @designmultimedia.com zurück
```

Siehe auch [stristr\(\)](#), [strrchr\(\)](#), [substr\(\)](#) und [ereg\(\)](#).

strlen

(PHP3 , PHP4)

strlen -- Ermitteln der String-Länge

Beschreibung:

int **strlen** (string str)

Gibt die Länge der Zeichenkette *string* zurück.

[Zurück](#)[Anfang](#)[Vor](#)

stristr

[Hoch](#)

str_pad

str_pad

(PHP4 >= 4.0.1)

str_pad -- Erweitert einen String auf eine bestimmte Länge unter Verwendung eines anderen Strings

Beschreibung:

```
string str_pad (string input, int pad_length [, string pad_string [, int pad_type]])
```

Mit dieser Funktion erweitern sie den String *input* auf der linken, rechten oder beiden Seiten auf die mit dem optionalen Parameter *pad_type* angegebene Länge. Wird das optionale Argument *pad_string* nicht angegeben, erfolgt die Erweiterung mit Leerzeichen, ansonsten wird hierzu der Inhalt von *pad_string* verwendet.

Das optionale Argument *pad_type* kann STR_PAD_RIGHT, STR_PAD_LEFT oder STR_PAD_BOTH sein. Wird *pad_type* nicht angegeben, so wird per default von STR_PAD_RIGHT ausgegangen.

Ist der Wert von *pad_length* negativ oder kleiner als die Länge des Input-Strings, wird keine Erweiterung vorgenommen.

Beispiel 1 Str_pad()-Beispiel:

```
$input = "Alien";  
print str_pad ($input, 10); // Ergebnis "Alien      "  
print str_pad ($input, 10, "-=", STR_PAD_LEFT); // Ergebnis "-=-=-Alien"  
print str_pad ($input, 10, "_", STR_PAD_BOTH); // Ergebnis "__Alien__"
```

strpos

(PHP3 , PHP4)

strpos -- Ermitteln des ersten Vorkommens innerhalb eines Strings

Beschreibung:

int **strpos** (string haystack, string needle [, int offset])

Gibt als numerischen Wert die Position des ersten Vorkommens von *needle* innerhalb der Zeichenkette *haystack* zurück. Anders als bei [strstr\(\)](#) kann diese Funktion eine komplette Zeichenkette unterstützen.

Wurde *needle* nicht gefunden, wird false zurück gegeben.

Anmerkung: Die Meldungen "Zeichen bei Position 0 gefunden (character found at position 0)" und "Zeichen nicht gefunden (character not found)" werden oft falsch ausgelegt. Hier die Unterscheidung:

```
// in PHP 4.0b3 und danach:
$pos = strpos ($mystring, "b");
if ($pos === false) { // Achtung: 3 Gleichheits-Zeichen
    // nicht gefunden...
}

// in Versionen vor 4.0b3:
$pos = strpos ($mystring, "b");
if (is_string ($pos) && !$pos) {
    // nicht gefunden...
}
```

Ist *needle* kein String, wird er zu einem Integer-Wert umgesetzt und als das diesem Wert entsprechende ASCII-Zeichen angesehen.

Der optionale Parameter *offset* ermöglicht es, den Startwert für die Suche nach *needle* innerhalb von *haystack* anzugeben. Die zurück gegebene Positions-Angabe ist dann relativ.

Siehe auch [strstr\(\)](#), [strchr\(\)](#), [substr\(\)](#), [stristr\(\)](#) und [strtr\(\)](#).

strpos

[Zurück](#)

str_pad

[Anfang](#)

[Hoch](#)

[Vor](#)

strchr

strchr

(PHP3 , PHP4)

strchr -- Findet das letzte Vorkommen eines Zeichens innerhalb einer Zeichenkette

Beschreibung:

string **strchr** (string haystack, string needle)

Diese Funktion gibt den letzten Teil des Strings *haystack* zurück, der ab dem letzten Vorkommen von *needle* steht (bis zum Ende).

Wird *needle* nicht gefunden, wird false zurück gegeben.

Enthält *needle* mehr als ein Zeichen, wird nur das erste Zeichen genommen.

Ist *needle* kein String, wird er zu einem Integer-Wert umgesetzt und als das diesem Wert entsprechende ASCII-Zeichen angesehen.

Beispiel 1 Strchr()-Beispiel:

```
// hole das letzte Verzeichnis aus $PATH
$dir = substr (strchr ($PATH, ":"), 1);

// hole alles nach dem letzten "Neue-Zeile"
$text = "Zeile 1\nZeile 2\nZeile 3";
$last = substr (strchr ($text, 10), 1 );
```

Siehe auch [substr\(\)](#), [stristr\(\)](#) und [strstr\(\)](#).

str_repeat

(PHP4 >= 4.0b4)

str_repeat -- Wiederholung eine String-Ausgabe

Beschreibung:

string **str_repeat** (string input, int multiplifier)

Die Ausgabe von *input_str* wird *multiplier* mal wiederholt. *Multiplier* muss grösser als 0 sein.

Beispiel 1 Str_repeat()-Beispiel:

```
echo str_repeat ("-= ", 10);
```

Es wird "-=-=-=-=-=-=" ausgegeben.

Anmerkung: Verfügbar ab PHP 4.0.

strrev

(PHP3 , PHP4)

strrev -- Umdrehen eines Strings

Beschreibung:

string **strrev** (string string)

Dreht die Schreibweise von *string* um.

strrpos

(PHP3 , PHP4)

strrpos -- Findet letztes Vorkommen eines Zeichens in einem String

Beschreibung:

int **strrpos** (string haystack, char needle)

Der Wert des letzten Vorkommens von *needle* im String *haystack* wird ermittelt. *Needle* darf nur ein einzelnes Zeichen sein. Wird als *needle* eine Zeichenkette übergeben, wird nur dessen erster Buchstabe ausgewertet.

Wurde *needle* nicht gefunden, wird false zurück gegeben.

Ist *needle* kein String, wird er zu einem Integer-Wert umgesetzt und als das diesem Wert entsprechende ASCII-Zeichen angesehen.

Siehe auch [strpos\(\)](#), [strchr\(\)](#), [substr\(\)](#), [stristr\(\)](#) und [strstr\(\)](#).

strspn

(PHP3 >= 3.0.3, PHP4)

strspn -- Ermittelt die Länge der übereinstimmenden Zeichen

Beschreibung:

int **strspn** (string str1, string str2)

Ermittelt die Anzahl der Zeichen am Anfang von *str1*, die aus dem in *str2* übergebenen Zeichenvorrat stammen.

So ergibt z.B.

```
strspn("42 ist die Antwort, was ist die Frage ...","1234567890");
```

als Ergebnis 2.

Siehe auch [strcspn\(\)](#).

strtok

(PHP3 , PHP4)

strtok -- Zerlegt einen String

Beschreibung:

string **strtok** (string arg1, string arg2)

Strtok() wird benutzt, um einen String (*arg1*) zu teilen. Den Satz "Das ist ein Beispiel" können sie in seine einzelnen Wörter zerlegen, in dem sie das Leerzeichen als Trennzeichen (*arg2*) angeben.

Beispiel 1 Strtok()-Beispiel:

```
$string = "Das ist noch ein Beispiel";  
$tok = strtok ($string, " ");  
while ($tok) {  
    echo "Wort=$tok<br>";  
    $tok = strtok (" ");  
}
```

Beachten sie, dass der zu zerlegende String nur beim ersten Aufruf von **strtok()** angegeben werden muss. Die nachfolgenden Aufrufe brauchen nur das Trennzeichen als Parameter. Falls sie eine neue Zeichenkette zerlegen wollen, rufen sie **strtok()** wieder mit dem Parameter *arg1* auf. Das initialisiert die Funktion. Sie können auch mehrere Trennzeichen in *arg2* angeben. Die Trennung erfolgt beim ersten Auftreten eines beliebigen Zeichens aus *arg2*.

Beachten sie, dass die Teile auch "0" sein können. Dies entspricht in bedingten Ausdrücken dem Wert false.

Siehe auch [split\(\)](#) und [explode\(\)](#).

strtolower

(PHP3 , PHP4)

strtolower -- Setzt einen String in Kleinbuchstaben um

Beschreibung:

string **strtolower** (string str)

Gibt *string* zurück, in dem alle Buchstaben in Kleinbuchstaben umgewandelt wurden.

Beachten sie, dass 'Buchstaben' vom Wert locale abhängig ist. Ist z.B. die Voreinstellung für locale "C", werden keine Umlaute (Ä, Ü, Ö) umgewandelt.

Beispiel 1 Strtolower()-Beispiel:

```
$str = "Dieter B. (M.T.) Ist Ein Schaf und Er LIEBT Sich So";  
$str = strtolower($str);  
print $str; // Gibt "dieter b. (m.t.) ist ein schaf und er liebt sich  
           // so" aus.
```

Siehe auch [strtoupper\(\)](#) und [ucfirst\(\)](#).

strtoupper

(PHP3 , PHP4)

strtoupper -- Setzt einen String in Großbuchstaben um

Beschreibung:

string **strtoupper** (string string)

Gibt *string* zurück, in dem alle Buchstaben in Großbuchstaben umgewandelt wurden.

Beachten sie, dass 'Buchstaben' vom Wert locale abhängig ist. Ist z.B. die Voreinstellung für locale "C", werden keine Umlaute (ä, ü, ö) umgewandelt.

Beispiel 1 Strtoupper()-Beispiel:

```
$str = "Dieter B. (M.T.) Ist Ein Schaf und Er LIEBT Sich So";  
$str = strtoupper($str);  
print $str; // Gibt "DIETER B. (M.T.) IST EIN SCHAF UND ER LIEBT SICH  
           // SO" aus.
```

Siehe auch [strtolower\(\)](#) und [ucfirst\(\)](#).

str_replace

(PHP3 >= 3.0.6, PHP4)

str_replace -- Ersetzt alle Vorkommen eines Strings in einem anderen String

Beschreibung:

string **str_replace** (string needle, string str, string haystack)

Diese Funktion ersetzt alle Vorkommen von *needle* innerhalb der Zeichenkette *haystack* durch den String *str*. Falls sie keine ausgefallenen String-Ersetzungen brauchen, sollten sie immer dieser Funktion den Vorzug vor [ereg_replace\(\)](#) geben.

Beispiel 1 Str_replace()-Beispiel:

```
$bodytag = str_replace ("%body%", "black", "<body text=%body%>");
```

Diese Funktion beruht auf einem Binär-Vergleich.

Anmerkung: Str_replace() ist seit PHP 3.0.6 verfügbar, war aber bis PHP 3.0.8 fehlerhaft.

Siehe auch [ereg_replace\(\)](#) und [strtr\(\)](#).

strtr

(PHP3 , PHP4)

strtr -- Tauscht bestimmte Zeichen aus

Beschreibung:

string **strtr** (string *str*, string *from*, string *to*)

Diese Funktion bearbeitet *str*, in dem alle Vorkommen von Zeichen aus *from* in die entsprechenden Zeichen aus *to* umgesetzt werden. Zurück gegeben wird der bearbeitete String.

Sind *from* und *to* von unterschiedlicher Länge werden die überzähligen Zeichen ignoriert.

Beispiel 1 Strtr()-Beispiel:

```
$addr = strtr($addr, "ääö", "aao");
```

Strtr() kann auch mit nur 2 Argumenten aufgerufen werden. Die Funktionsweise ist dann anders: *From* muss dann ein Array sein, welches String -> String Paare enthält, die innerhalb des Quellstrings ausgetauscht werden sollen. *Strtr* sucht immer erst die längste Entsprechung und wird keine Auswechselungen vornehmen, die bereits bearbeitet wurden.

Beispiele dazu:

```
$trans = array ("hallo" => "hi", "hi" => "hallo");  
echo strtr("hi alle, ich sage hallo", $trans) . "\n";
```

Die Ausgabe ist: "hallo alle, ich sage hi",

Anmerkung: Die Möglichkeit mit 2 Parametern wurde in PHP 4.0 hinzugefügt.

Siehe auch [ereg_replace\(\)](#).

substr

(PHP3 , PHP4)

substr -- Gibt einen Teilstring zurück

Beschreibung:

string **substr** (string string, int start [, int length])

Substr() gibt den Teil von *string* zurück, der durch die *start* und *length* Parameter definiert wurde.

Hat *start* einen positiven Wert, beginnt der zurück gegebene String an dieser Position von *string*.

Beispiele:

```
$rest = substr ("abcdef", 1);    // gibt "bcdef" zurück  
$rest = substr ("abcdef", 1, 3); // gibt "bcd" zurück
```

Ist *start* negativ, gibt er den Beginn-Wert ab Ende des Strings an (also rückwärts gezählt).

Beispiele:

```
$rest = substr ("abcdef", -1);    // gibt "f" zurück  
$rest = substr ("abcdef", -2);    // gibt "ef" zurück  
$rest = substr ("abcdef", -3, 1); // gibt "d" zurück  
$rest = substr ("abcdef", 1, -1); // gibt "bcde" zurück  
$rest = substr ("abcdef", 4, -4); // gibt "" zurück (nichts)  
$rest = substr ("abcdef", 7, 1); // gibt "" zurück (nichts)  
$rest = substr ("abcdef", 7, -4); // gibt "" zurück (nichts)  
$rest = substr ("abcdef", 4, 4); // gibt "ef" zurück  
$rest = substr ("abcdef", -2, 4); // gibt "ef" zurück  
$rest = substr ("abcdef", -5, -4); // gibt "b" zurück
```

Siehe auch [strrchr\(\)](#) und [ereg\(\)](#).

substr_count

(PHP4 >= 4.0RC2)

substr_count -- Ermittelt die Häufigkeit des Vorkommens einer Zeichenkette in einem String

Beschreibung:

int **substr_count** (string haystack, string needle)

Die Funktion **substr_count()** ermittelt, wie oft *needle* in dem String *haystack* vorkommt.

Beispiel 1 Substr_count()-Beispiel:

```
print substr_count("Dieser Test ist Mist", "ist"); // ergibt 2
```

Diese Funktion steht ab PHP 4.0.0 zur Verfügung.

substr_replace

(PHP4 >= 4.0b4)

substr_replace -- Ersetzt Text in einer Zeichenkette

Beschreibung:

string **substr_replace** (string string, string replacement, int start [, int length])

Substr_replace() ersetzt den Teil von *string*, der durch die Parameter *start* und (optional) *length* begrenzt wird mit der in *replacement* angegebenen Zeichnekette. Das Ergebnis wird zurück gegeben.

Ist *start* positiv, beginnt der Austausch am Offset *start* innerhalb von *string*.

Ist *start* negativ, wird vom Ende der Zeichenkette *string* bis zum Wert von *start* rückwärts gezählt und dort mit dem Austausch begonnen.

Ist *length* angegeben und positive, stellt dieser Parameter die Länge des auszuwechselnden Teilstrings dar. Ist *length* angegeben und negativ, gibt er die Zeichenzahl an um die ab Ende von *string* rückwärts gezählt wird. Bis zu dieser Stelle erfolgt dann der Austausch. Ist *length* nicht angegeben, wird unterstellt, dass sie die Ersetzung bis zum String-Ende (strlen(*string*)) bezwecken.

Beispiel 1 Substr_replace()-Beispiel:

```
<?php
$var = 'ABCDEFGH:/MNRPQR/';
echo "Original: $var<hr>\n";

/* Diese 2 Beispiele ersetzen alles in $var mit 'bob'. */
echo substr_replace ($var, 'bob', 0) . "<br>\n";
echo substr_replace ($var, 'bob', 0, strlen ($var)) . "<br>\n";

/* Fügt 'bob' am Anfang von $var ein . */
echo substr_replace ($var, 'bob', 0, 0) . "<br>\n";

/* Die nächsten beiden ersetzen 'MNRPQR' in $var durch 'bob'. */
echo substr_replace ($var, 'bob', 10, -1) . "<br>\n";
echo substr_replace ($var, 'bob', -7, -1) . "<br>\n";

/* Löscht 'MNRPQR' aus $var. */
```

```
echo substr_replace ($var, '', 10, -1) . "<br>\n";  
?>
```

Siehe auch [str_replace\(\)](#) und [substr\(\)](#).

Anmerkung: `Substr_replace()` existiert ab PHP 4.0.

[Zurück](#)[substr_count](#)[Anfang](#)[Hoch](#)[Vor](#)[trim](#)

trim

(PHP3 , PHP4)

trim -- Entfernt überflüssige Zeichen ("whitespace" am Anfang und Ende eines Strings

Beschreibung:

string **trim** (string str)

Diese Funktion entfernt Leerraum am Anfang und Ende einer Zeichenkette. Der gekürzte String wird zurück gegeben. Als überflüssige Zeichen werden z.Zt. angesehen: "\n", "\r", "\t", "\v", "\0" und normale Leerzeichen.

Siehe auch [chop\(\)](#) und [ltrim\(\)](#).

ucfirst

(PHP3 , PHP4)

ucfirst -- Setzt das erste Zeichen eines Strings als Großbuchstabe um

Beschreibung:

string **ucfirst** (string str)

Setzt das erste Zeichen von *str* in einen Großbuchstaben um, wenn es ein Zeichen des Alphabets ist.

Beachten sie, dass die Zeichen des Alphabets abhängig sind vom Wert des gesetzten locale. Ist dieser (Voreinstellung) "C", werden die deutschen Umlaute nicht erkannt.

Beispiel 1 ucfirst()-Beispiel:

```
$text = 'markus ist ein kleines schaf.';
$text = ucfirst ($text); // $text ist nun "Markus ist ein kleines
                        // schaf."
```

Siehe auch [strtoupper\(\)](#) und [strtolower\(\)](#).

ucwords

(PHP3 >= 3.0.3, PHP4)

ucwords -- Setzt die ersten Buchstaben aller Wörter eines Strings in die entsprechenden Großbuchstaben um

Beschreibung:

string **ucwords** (string *str*)

Setzt die ersten Buchstaben aller Wörter innerhalb von *str* in die entsprechenden Großbuchstaben um, sofern es sich um alphabetische Zeichen handelt.

Beachten sie, dass die Zeichen des Alphabets abhängig sind vom Wert des gesetzten locale. Ist dieser (Voreinstellung) "C", werden die deutschen Umlaute nicht erkannt.

Beispiel 1 ucwords()-Beispiel:

```
$text = "markus ist ein kleines schaf.";
$text = ucwords ($text); // $text ist jetzt: "Markus Ist Ein Kleines
                        // Schaf."
```

Siehe auch [strtoupper\(\)](#), [strtolower\(\)](#) und [ucfirst\(\)](#).

LVIII. Sybase functions

Inhaltsverzeichnis

[sybase_affected_rows](#) get number of affected rows in last query

[sybase_close](#) close Sybase connection

[sybase_connect](#) open Sybase server connection

[sybase_data_seek](#) move internal row pointer

[sybase_fetch_array](#) fetch row as array

[sybase_fetch_field](#) get field information

[sybase_fetch_object](#) fetch row as object

[sybase_fetch_row](#) get row as enumerated array

[sybase_field_seek](#) set field offset

[sybase_free_result](#) free result memory

[sybase_num_fields](#) get number of fields in result

[sybase_num_rows](#) get number of rows in result

[sybase_pconnect](#) open persistent Sybase connection

[sybase_query](#) send Sybase query

[sybase_result](#) get result data

[sybase_select_db](#) select Sybase database

sybase_affected_rows

(PHP3 >= 3.0.6, PHP4)

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\(\)](#).

Anmerkung: This function is only available using the CT library interface to Sybase, and not the DB library.

sybase_close

(PHP3 , PHP4)

sybase_close -- close Sybase connection

Description

int 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

(PHP3 , PHP4)

sybase_connect -- open Sybase server connection

Description

int **sybase_connect** (string servername, string username, string password)

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

(PHP3 , PHP4)

sybase_data_seek -- move internal row pointer

Description

int **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 specified row number. The next call to [sybase_fetch_row\(\)](#) would return that row.

See also: **sybase_data_seek()**.

sybase_fetch_array

(PHP3 , PHP4)

sybase_fetch_array -- fetch row as array

Description

int 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

(PHP3 , PHP4)

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 retrieved by sybase_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 [sybase_field_seek\(\)](#)

sybase_fetch_object

(PHP3 , PHP4)

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

(PHP3 , PHP4)

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_rows() 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

(PHP3 , PHP4)

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

(PHP3 , PHP4)

sybase_free_result -- free result memory

Description

int **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, you may call **sybase_free_result()** with the result identifier as an argument and the associated result memory will be freed.

[Zurück](#)[Anfang](#)[Vor](#)

sybase_field_seek

[Hoch](#)

sybase_num_fields

sybase_num_fields

(PHP3 , PHP4)

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

(PHP3 , PHP4)

sybase_num_rows -- get number of rows in result

Description

int **sybase_num_rows** (string 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

(PHP3 , PHP4)

sybase_pconnect -- open persistent Sybase connection

Description

`int sybase_pconnect (string servername, string username, string password)`

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

(PHP3 , PHP4)

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

(PHP3 , PHP4)

sybase_result -- get result data

Description

`int 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

(PHP3 , PHP4)

sybase_select_db -- select Sybase database

Description

int **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\(\)](#)

LIX. URL functions

Inhaltsverzeichnis

[base64_decode](#) dekodiert Daten mit MIME base64

[base64_encode](#) Kodiert Daten MIME base64

[parse_url](#) Analysiert eine URL und zerlegt diese in ihre Bestandteile.

[urldecode](#) Dekodiert eine URL-kodierte Zeile.

[urlencode](#) URL-kodiert eine Zeile

base64_decode

(PHP3 , PHP4)

base64_decode -- dekodiert Daten mit MIME base64

Beschreibung

string **base64_decode** (string encoded_data)

base64_decode() dekodiert *encoded_data* und bringt die Originaldaten wieder. Die ausgegebenen Dateien können im Binärformat sein.

Siehe auch: [base64_encode\(\)](#), RFC-2045 Sektion 6.8.

base64_encode

(PHP3 , PHP4)

base64_encode -- Kodiert Daten MIME base64

Beschreibung

string **base64_encode** (string data)

base64_encode() kodiert *data* mit base64. Diese Kodierung ist dafür da, Binärdaten über Technologien, die 8-bit nicht sauber übertragen können, wie z.B. Emails.

Base64-kodierte Daten benötigen ungefähr 33% mehr Speicher als die Originaldaten.

Siehe auch: [base64_decode\(\)](#), [chunk_split\(\)](#), RFC-2045 Sektion 6.8.

parse_url

(PHP3 , PHP4)

parse_url -- Analysiert eine URL und zerlegt diese in ihre Bestandteile.

Beschreibung

array **parse_url** (string url)

Diese Funktion erzeugt ein assziatives Array aus den verschiedenen Teilen einer URL. Diese umfassen "scheme", "host", "port", "user", "pass", "path", "query", und "fragment".

urlencode

(PHP3 , PHP4)

urlencode -- Dekodiert eine URL-kodierte Zeile.

Beschreibung

string **urlencode** (string str)

Dekodiert jede %## Kodierung einer Zeile. Die dekodierte Zeile wird zurückgegeben.

Beispiel 1 urlencode() Beispiel

```
$a = split ('&', $querystring);
$i = 0;
while ($i < count ($a)) {
    $b = split ('=', $a [$i]);
    echo 'Value for parameter ', htmlspecialchars (urlencode ($b [0])),
        ' is ', htmlspecialchars (urlencode ($b [1])), "<BR>";
    $i++;
}
```

Siehe auch [urlencode\(\)](#)

urlencode

(PHP3 , PHP4)

urlencode -- URL-kodiert eine Zeile

Beschreibung

string **urlencode** (string str)

Zeigt eine Zeile an, bei der alle nicht alphanumerischen Zeichen bis auf - _ . durch das Prozentzeichen (%) ersetzt werden, gefolgt von zwei Hexzeichen und Leerzeichen durch ein Pluszeichen (+) ersetzt werden. Genauso werden auch Daten kodiert, die über ein WWW-Formularat gesendet werden und auf die gleiche Weise mit `application/x-www-form-urlencoded` kodiert werden. Dies unterscheidet sich von der Kodierung in RFC1738 (siehe [rawurlencode\(\)](#)) bei dem aus historischen Gründen Leerzeichen wie Pluszeichen (+) kodiert werden. Diese Funktion ist sehr nützlich, wenn eine Zeichenkette kodiert werden soll und die Variablen an die nächste Seite weitergegeben werden sollen:

Beispiel 1 urlencode() Beispiel

```
echo '<A HREF="mycgi?foo=', urlencode ($userinput), '>';'
```

Siehe auch [urldecode\(\)](#).

LX. Variablen-Functions

Inhaltsverzeichnis

- [call_user_func](#) Aufruf einer benutzerdefinierten Funktion
- [call_user_method](#) Call a user method on an specific object
- [doubleval](#) Konvertiert einen Wert nach double
- [empty](#) Prüft, ob eine Variable einen Wert enthält
- [gettype](#) Liefert den Datentyp einer Variablen
- [intval](#) Konvertiert einen Wert nach integer
- [is_array](#) Prüft, ob Variable ein Array ist
- [is_bool](#) Prüft, ob eine Variable vom Typ bool ist
- [is_double](#) Prüft, ob eine Variable vom Typ double ist
- [is_float](#) Prüft, ob eine Variable vom Typ float ist
- [is_int](#) Prüft, ob eine Variable vom Typ int ist
- [is_integer](#) Prüft, ob eine Variable vom Typ integer ist
- [is_long](#) Prüft, ob eine Variable vom Typ long ist
- [is_object](#) Prüft, ob eine Variable vom Typ object ist
- [is_real](#) Prüft, ob Variable vom Typ real ist
- [is_resource](#) Prüft, ob eine Variable vom Typ resource ist
- [is_string](#) Prüft, ob Variable vom Typ string ist
- [isset](#) Prüft die Existenz einer Variablen
- [print_r](#) Gibt Variablen-Informationen in lesbarer Form aus
- [settype](#) Legt den Typ einer Variablen fest
- [strval](#) Konvertierung zum string
- [unset](#) Löschen einer Variablen
- [var_dump](#) Gibt alle Informationen zu einer Variablen aus

call_user_func

(PHP3 >= 3.0.3, PHP4)

call_user_func -- Aufruf einer benutzerdefinierten Funktion

Description

mixed **call_user_func** (string function_name [, mixed parameter [, mixed ...]])

Ruft die Funktion mit Namen *function_name* mit den Parametern *parameters* auf.

```
function friseur ($type) {  
    print "Sie möchten einen $type-Kopf, kein Problem<br>";  
}  
call_user_func ('friseur', "Bubi");  
call_user_func ('friseur', "Pilz");
```

call_user_method

(PHP3 >= 3.0.3, PHP4)

`call_user_method` -- Call a user method on an specific object

Description

mixed **call_user_method** (string *method_name*, object *obj* [, mixed *parameter* [, mixed ...]])

Calls a 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");
?>
```


[Zurück](#)

call_user_func

[Anfang](#)

[Hoch](#)

[Vor](#)

doubleval

doubleval

(PHP3 , PHP4)

doubleval -- Konvertiert einen Wert nach double

Beschreibung

double **doubleval** (mixed var)

Diese Funktion interpretiert ihr Argument als Fließkommazahl und liefert den Wert als double zurück.

Als *var* können beliebige skalare Ausdrücke, aber keine Arrays oder Objekte übergeben werden.

```
$var = '122.34343The';  
$double_value_of_var = doubleval ($var);  
print $double_value_of_var; // Ausgabe: 122.34343
```

Siehe auch [intval\(\)](#), [strval\(\)](#), [settype\(\)](#) und [Typ-Veränderung](#).

empty

(unknown)

empty -- Prüft, ob eine Variable einen Wert enthält

Beschreibung

int **empty** (mixed var)

Diese Funktion liefert true, wenn eine Variable nicht definiert, leer oder gleich 0 ist, ansonsten false.

```
$var = 0;
if (empty ($var)) { #evaluates true
    print '$var is either 0 or not at all set';
}
if (!isset ($var)) { // evaluates false
    print 'The $var is not set at all';
}
```

empty() kann nur auf Variablen, nicht aber auf Ausdrücke angewandt werden.

Siehe auch [isset\(\)](#) und [unset\(\)](#).

gettype

(PHP3 , PHP4)

gettype -- Liefert den Datentyp einer Variablen

Beschreibung

string **gettype** (mixed var)

Diese Funktion liefert den Namen des Typs der Variablen *var* zurück.

Mögliche Ergebnisse sind:

- "integer"
- "double"
- "string"
- "array"
- "object"
- "unknown type"

Siehe auch [settype\(\)](#).

intval

(PHP3 , PHP4)

intval -- Konvertiert einen Wert nach integer

Beschreibung

int **intval** (mixed var [, int base])

Diese Funktion konvertiert den Parameter *var* zu einem integer, wobei *base* als Basis benutzt wird (Default ist *base*=10).

var kann ein Ausdruck oder eine einfache Variable sein, nicht jedoch ein Array oder Objekt.

Siehe auch [doubleval\(\)](#), [strval\(\)](#), [settype\(\)](#) und [Typ-Veränderungen](#).

is_array

(PHP3 , PHP4)

is_array -- Prüft, ob Variable ein Array ist

Beschreibung

int **is_array** (mixed var)

Liefert true, wenn *var* ein Array ist, sonst false.

Siehe auch [is_double\(\)](#), [is_float\(\)](#), [is_int\(\)](#), [is_integer\(\)](#), [is_real\(\)](#), [is_string\(\)](#), [is_long\(\)](#), und [is_object\(\)](#).

is_bool

(PHP4 >= 4.0b4)

is_bool -- Prüft, ob eine Variable vom Typ bool ist

Beschreibung

int **is_bool** (mixed *var*)

Liefert true, wenn *var* vom Typ bool ist.

[Zurück](#)[Anfang](#)[Vor](#)

is_array

[Hoch](#)

is_double

is_double

(PHP3 , PHP4)

is_double -- Prüft, ob eine Variable vom Typ double ist

Beschreibung

int **is_double** (mixed var)

Liefert true, wenn *var* vom Ty double ist, sonst false.

Siehe auch [is_array\(\)](#), [is_float\(\)](#), [is_int\(\)](#), [is_integer\(\)](#), [is_real\(\)](#), [is_string\(\)](#), [is_long\(\)](#) und [is_object\(\)](#).

is_float

(PHP3 , PHP4)

is_float -- Prüft, ob eine Variable vom Typ float ist

Beschreibung

int **is_float** (mixed var)

Diese Funktion ist ein Synonym für [is_double\(\)](#).

Siehe auch [is_double\(\)](#), [is_real\(\)](#), [is_int\(\)](#), [is_integer\(\)](#), [is_string\(\)](#), [is_object\(\)](#), [is_array\(\)](#) und [is_long\(\)](#).

is_int

(PHP3 , PHP4)

is_int -- Prüft, ob eine Variable vom Typ int ist

Beschreibung

int **is_int** (mixed var)

Diese Funktion ist ein Synonym für **is_int()**.

Siehe auch [is_double\(\)](#), [is_float\(\)](#), [is_integer\(\)](#), [is_string\(\)](#), [is_real\(\)](#), [is_object\(\)](#), [is_array\(\)](#) und [is_long\(\)](#).

is_integer

(PHP3 , PHP4)

is_integer -- Prüft, ob eine Variable vom Typ integer ist

Beschreibung

int **is_integer** (mixed var)

Diese Funktion liefert true, wenn *var* vom Typ integer ist, sonst false.

Siehe auch [is_double\(\)](#), [is_float\(\)](#), [is_int\(\)](#), [is_string\(\)](#), [is_real\(\)](#), [is_object\(\)](#), [is_array\(\)](#) und [is_long\(\)](#).

is_long

(PHP3 , PHP4)

is_long -- Prüft, ob eine Variable vom Typ long ist

Beschreibung

int **is_long** (mixed var)

Diese Funktion ist ein Synonym für [is_integer\(\)](#).

Siehe auch [is_double\(\)](#), [is_float\(\)](#), [is_int\(\)](#), [is_real\(\)](#), [is_string\(\)](#), [is_object\(\)](#), [is_array\(\)](#) und [is_integer\(\)](#).

is_object

(PHP3 , PHP4)

is_object -- Prüft, ob eine Variable vom Typ object ist

Beschreibung

int **is_object** (mixed var)

Liefert true, wenn *var* ein Objekt enthält, sonst false.

Siehe auch [is_long\(\)](#), [is_int\(\)](#), [is_integer\(\)](#), [is_float\(\)](#), [is_double\(\)](#), [is_real\(\)](#), [is_string\(\)](#) und [is_array\(\)](#).

is_real

(PHP3 , PHP4)

is_real -- Prüft, ob Variable vom Typ real ist

Beschreibung

int **is_real** (mixed var)

Diese Funktion ist ein Synonym für [is_double\(\)](#).

Siehe auch [is_long\(\)](#), [is_int\(\)](#), [is_integer\(\)](#), [is_float\(\)](#), [is_double\(\)](#), [is_object\(\)](#), [is_string\(\)](#) und [is_array\(\)](#).

is_resource

(PHP4 >= 4.0b4)

is_resource -- Prüft, ob eine Variable vom Typ resource ist

Beschreibung

```
int is_resource (mixed var)
```

is_resource() liefert true, wenn es sich bei der Variable *var* um eine PHP-Resource wie z.B. ein Datei- oder Result-Handle handelt, ansonsten false. PHP-Resourcen werden von PHP-internen Funktionen erzeugt und freigegeben. Unterläßt der Benutzer die Freigabe in seinem Code, so kann diese durch den speziellen Datentyp resource dennoch angestossen werden, sobald die Resource-Variable nicht mehr referenziert wird.

is_string

(PHP3 , PHP4)

is_string -- Prüft, ob Variable vom Typ string ist

Beschreibung

int **is_string** (mixed var)

Liefert true, wenn *var* vom Typ string ist, sonst false.

Siehe auch [is_long\(\)](#), [is_int\(\)](#), [is_integer\(\)](#), [is_float\(\)](#), [is_double\(\)](#), [is_real\(\)](#), [is_object\(\)](#) und [is_array\(\)](#).

isset

(unknown)

isset -- Prüft die Existenz einer Variablen

Beschreibung

int **isset** (mixed var)

Diese Funktion liefert true, wenn die Variable oder das Array-Element *var* existiert, sonst false.

Wird eine Variable oder ein Array-Element mit [unset\(\)](#) freigegeben so liefert **isset()** anschließend ebenfalls false.

```
echo isset ($a); // false
$a = "test";
echo isset ($a); // true
unset ($a);
echo isset ($a); // false
```

Siehe auch [empty\(\)](#) und [unset\(\)](#).

print_r

(PHP4)

print_r -- Gibt Variablen-Informationen in lesbarer Form aus

Beschreibung

void **print_r** (mixed expression)

Diese Funktion zeigt Informationen über eine Variable oder einen Ausdruck in lesbarer Form an. Einfache Werte werden direkt ausgegeben, Arrays und Objekte werden dagegen als eingerückte Schlüssel/Wert Paare ausgegeben.

print_r() ähnelt [var_dump\(\)](#), gibt aber Typinformationen nur bei Arrays und Objekten aus.

```
<?php
$a = array (1, 2, array ("a", "b", "c"));
print_r ($a);
?>
```

Warnung

Diese Funktion ist nicht in der Lage, zyklische Referenzen aufzulösen, d.h. die Ausgabe eines Arrays oder Objektes, das sich selbst enthält, wie z.B. `print_r ($GLOBALS)`, wird niemals beendet.

settype

(PHP3 , PHP4)

settype -- Legt den Typ einer Variablen fest

Beschreibung

int **settype** (string var, string type)

Der Typ der Variable *var* wird festgelegt als *type*.

Mögliche Werte für *type* sind:

- "integer"
- "double"
- "string"
- "array"
- "object"

Liefert true bei Erfolg, sonst false.

Siehe auch [gettype\(\)](#).

strval

(PHP3 , PHP4)

strval -- Konvertierung zum string

Beschreibung

string **strval** (mixed *var*)

Diese Funktion liefert den übergebenen Ausdruck *var* als String-Repräsentation zurück.

var darf veon jedem einfache Datentyp, aber kein Array oder Objekt sein.

Siehe auch [doubleval\(\)](#), [intval\(\)](#), [settype\(\)](#) und [Typ-Veränderung](#).

unset

(unknown)

unset -- Löschen einer Variablen

Beschreibung

int **unset** (mixed var)

unset() gibt eine Variable oder ein Array-Element vollständig frei.

Beispiel 1 Unset() example

```
unset ($foo);  
unset ($bar['quux']);
```

Siehe auch [isset\(\)](#) und [empty\(\)](#).

var_dump

(PHP3 >= 3.0.5, PHP4)

var_dump -- Gibt alle Informationen zu einer Variablen aus

Beschreibung

void **var_dump** (mixed expression)

Diese Fuktion liefert Informationen über Typ und Wert des Parameter *expression* zurück. Arrays und Objekte werden rekursiv durchlaufen und mit entsprechender Einrückung dargestellt

Siehe auch [print_r\(\)](#).

```
<pre>
<?php
    $a = array (1, 2, array ("a", "b", "c"));
    var_dump ($a);
?>
</pre>
```

LXI. Vmailmgr functions

These functions require [qmail](#) and the [vmailmgr package](#) by Bruce Guenter.

For all functions, the following two variables are defined as: string vdomain the domain name of your virtual domain (vdomain.com) string basepwd the password of the 'real' user that holds the virtual users

Only up to 8 characters are recognized in passwords for virtual users

Return status for all functions matches response in response.h

0 ok

1 bad

2 error

3 error connecting

Known problems: [vm_deluser\(\)](#) does not delete the user directory as it should. [vm_addalias\(\)](#) currently does not work correctly.

```
<?php
dl("php3_vmailmgr.so"); //load the shared library
$vdomain="vdomain.com";
$basepwd="password";
?>
```

Inhaltsverzeichnis

[vm_adduser](#) Add a new virtual user with a password

[vm_addalias](#) Add an alias to a virtual user

[vm_passwd](#) Changes a virtual users password

[vm_delalias](#) Removes an alias

[vm_deluser](#) Removes a virtual user

vm_adduser

(PHP3 , PHP4 <= 4.0.0)

vm_adduser -- Add a new virtual user with a password

Description

int **vm_adduser** (string vdomain, string basepwd, string newusername, string newuserpassword)

Add a new virtual user with a password. *newusername* is the email login name and *newuserpassword* the password for this user.

vm_addalias

(PHP3 , PHP4 <= 4.0.0)

vm_addalias -- Add an alias to a virtual user

Description

int **vm_addalias** (string vdomain, string basepwd, string username, string alias)

Add an alias to a virtual user. *username* is the email login name and *alias* is an alias for this vuser.

vm_passwd

(PHP3 , PHP4 <= 4.0.0)

vm_passwd -- Changes a virtual users password

Description

```
int vm_passwd (string vdomain, string username, string password,  
string newpassword)
```

Changes a virtual users password. *username* is the email login name, *password* the old password for the vuser, and *newpassword* the new password.

vm_delalias

(PHP3 , PHP4 <= 4.0.0)

vm_delalias -- Removes an alias

Description

int **vm_delalias** (string vdomain, string basepwd, string alias)

Removes an alias.

vm_deluser

(PHP3 , PHP4 <= 4.0.0)

vm_deluser -- Removes a virtual user

Description

```
int vm_deluser (string vdomain, string username)
```

Removes a virtual user..

LXII. WDDX functions

These functions are intended for work with [WDDX](#).

Note that 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.

Beispiel 1 Serializing a single value

```
<?php
print wddx_serialize_value("PHP to WDDX packet example", "PHP packet");
?>
```

This example will produce:

```
<wddxPacket version='0.9'><header comment='PHP packet' /><data>
<string>PHP to WDDX packet example</string></data></wddxPacket>
```

Beispiel 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='0.9'><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>
```

Inhaltsverzeichnis

[wddx_serialize_value](#) Serialize a single value into a WDDX packet

[wddx_serialize_vars](#) Serialize variables into a WDDX packet

[wddx_packet_start](#) Starts a new WDDX packet with structure inside it

[wddx_packet_end](#) Ends a WDDX packet with the specified ID

[wddx_add_vars](#) Ends a WDDX packet with the specified ID

[wddx_deserialize](#) Deserializes a WDDX packet

[Zurück](#)

vm_deluser

[Anfang](#)

[Hoch](#)

[Vor](#)

wddx_serialize_value

wddx_serialize_value

(PHP3 >= 3.0.7, PHP4 >= 4.0b2)

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

(PHP3 >= 3.0.7, PHP4 >= 4.0b2)

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.

Beispiel 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='0.9'><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>
```


wddx_packet_start

(PHP3 >= 3.0.7, PHP4 >= 4.0b2)

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.

[Zurück](#)[Anfang](#)[Vor](#)[wddx_serialize_vars](#)[Hoch](#)[wddx_packet_end](#)

wddx_packet_end

(PHP3 >= 3.0.7, PHP4 >= 4.0b2)

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.

[Zurück](#)[Anfang](#)[Vor](#)

wddx_packet_start

[Hoch](#)

wddx_add_vars

wddx_add_vars

(PHP3 >= 3.0.7, PHP4 >= 4.0b2)

wddx_add_vars -- Ends a WDDX packet with the specified ID

Description

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

(PHP3 >= 3.0.7, PHP4 >= 4.0b2)

wddx_deserialize -- Deserializes a WDDX packet

Description

mixed **wddx_deserialize** (string packet)

wddx_deserialized() 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.

[Zurück](#)

wddx_add_vars

[Anfang](#)[Hoch](#)[Vor](#)

XML parser functions

LXIII. XML parser functions

Introduction

About XML

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/>.

Installation

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://www.guardian.no/~ssb/phpxml.html>.

Note that if you are using Apache-1.3.7 or later, you already have the required expat library. Simply configure PHP using `--with-xml` (without any additional path) and it will automatically use the expat library built into Apache.

On UNIX, run **configure** with the `--with-xml` option. The expat library should be installed somewhere your compiler can find it. If you compile PHP as a module for Apache 1.3.9 or later, PHP will automatically use the bundled expat library from Apache. You may need to set `CPPFLAGS` and `LDFLAGS` in your environment before running configure if you have installed expat somewhere exotic.

Build PHP. *Tada!* That should be it.

About This Extension

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.

The XML event handlers defined are:

Tabelle 1 Supported XML handlers

PHP function to set handler	Event description

<u>xml_set_element_handler()</u>	Element events are issued whenever the XML parser encounters start or end tags. There are separate handlers for start tags and end tags.
<u>xml_set_character_data_handler()</u>	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.
<u>xml_set_processing_instruction_handler()</u>	PHP programmers should be familiar with processing instructions (PIs) already. <code><?php ?></code> is a processing instruction, where <i>php</i> is called the "PI target". The handling of these are application-specific, except that all PI targets starting with "XML" are reserved.
<u>xml_set_default_handler()</u>	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.
<u>xml_set_unparsed_entity_decl_handler()</u>	This handler will be called for declaration of an unparsed (NDATA) entity.
<u>xml_set_notation_decl_handler()</u>	This handler is called for declaration of a notation.
<u>xml_set_external_entity_ref_handler()</u>	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 <u>the external entity example</u> 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.

Some Examples

Here are some example PHP scripts parsing XML documents.

XML Element Structure Example

This first example displays the structure of the start elements in a document with indentation.

Beispiel 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

Beispiel 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`.)

Beispiel 3 External Entity Example

```

$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);

?>

```

Beispiel 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;
  <sect1 id="about">
    <title>About this Document</title>
    <para>
      <!-- this is a comment -->
      <?php print 'Hi! This is PHP version '.phpversion(); ?>
    </para>
  </sect1>
</chapter>

```

This file is included from xmltest.xml:

Beispiel 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>
```

Inhaltsverzeichnis[xml_parser_create](#) create an XML parser[xml_set_object](#) Use XML Parser withing an object[xml_set_element_handler](#) set up start and end element handlers[xml_set_character_data_handler](#) set up character data handler[xml_set_processing_instruction_handler](#) Set up processing instruction (PI) handler[xml_set_default_handler](#) set up default handler[xml_set_unparsed_entity_decl_handler](#) Set up unparsed entity declaration handler[xml_set_notation_decl_handler](#) set up notation declaration handler[xml_set_external_entity_ref_handler](#) set up external entity reference handler[xml_parse](#) start parsing an XML document[xml_get_error_code](#) get XML parser error code[xml_error_string](#) get XML parser error string[xml_get_current_line_number](#) get current line number for an XML parser[xml_get_current_column_number](#) Get current column number for an XML parser[xml_get_current_byte_index](#) get current byte index for an XML parser[xml_parser_free](#) Free an XML parser[xml_parser_set_option](#) set options in an XML parser[xml_parser_get_option](#) get options from an XML parser[utf8_decode](#) Converts a string with ISO-8859-1 characters encoded with UTF-8 to single-byte ISO-8859-1.[utf8_encode](#) encodes an ISO-8859-1 string to UTF-8[Zurück](#)[Anfang](#)[Vor](#)[wddx_deserialize](#)[Hoch](#)[xml_parser_create](#)

xml_parser_create

(PHP3 >= 3.0.6, PHP4)

xml_parser_create -- create an XML parser

Description

int 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_set_object

(PHP4 >= 4.0b4)

xml_set_object -- Use XML Parser withing an object

Description

`void xml_set_object (int parser, object &object)`

This function makes *parser* useable from within *object*. All callback functions settet via [xml_set_element_handler\(\)](#) etc are 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>");  
?>
```

Anmerkung: `xml_set_object()` handling was added in PHP 4.0.

[Zurück](#)[xml_parser_create](#)[Anfang](#)[Hoch](#)[Vor](#)[xml_set_element_handler](#)

xml_set_element_handler

(PHP3 >= 3.0.6, PHP4)

xml_set_element_handler -- set up start and end element handlers

Description

int xml_set_element_handler (int parser, string startElementHandler, string endElementHandler)

Sets the element handler functions for the XML parser *parser*. *startElementHandler* and *endElementHandler* are strings containing the names of functions that must exist when [xml_parse\(\)](#) is called for *parser*.

The function named by *startElementHandler* must accept three parameters:

startElementHandler (int 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 *endElementHandler* must accept two parameters:

endElementHandler (int 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.

There is currently no support for object/method handlers.

[Zurück](#)

xml_set_object

[Anfang](#)

[Hoch](#)

[Vor](#)

xml_set_character_data_handler

xml_set_character_data_handler

(PHP3 >= 3.0.6, PHP4)

xml_set_character_data_handler -- set up character data handler

Description

int xml_set_character_data_handler (int *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 (int *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.

There is currently no support for object/method handlers.

xml_set_processing_instruction_handler

(PHP3 >= 3.0.6, PHP4)

xml_set_processing_instruction_handler -- Set up processing instruction (PI) handler

Description

`int xml_set_processing_instruction_handler (int 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 (int 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.

There is currently no support for object/method handlers.

[Zurück](#)

xml_set_character_data_handler

[Anfang](#)

[Hoch](#)

[Vor](#)

xml_set_default_handler

xml_set_default_handler

(PHP3 >= 3.0.6, PHP4)

xml_set_default_handler -- set up default handler

Description

int xml_set_default_handler (int *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 (int *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.

There is currently no support for object/method handlers.

xml_set_unparsed_entity_decl_handler

(PHP3 >= 3.0.6, PHP4)

xml_set_unparsed_entity_decl_handler -- Set up unparsed entity declaration handler

Description

int xml_set_unparsed_entity_decl_handler (int 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 (int parser, string entityName, string base, string systemId,
string publicId, string notationName)

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

entityName

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.

systemId

System identifier for the external entity.

publicId

Public identifier for the external entity.

notationName

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.

There is currently no support for object/method handlers.

[Zurück](#)[Anfang](#)[Vor](#)[xml_set_default_handler](#)[Hoch](#)[xml_set_notation_decl_handler](#)

xml_set_notation_decl_handler

(PHP3 >= 3.0.6, PHP4)

xml_set_notation_decl_handler -- set up notation declaration handler

Description

int xml_set_notation_decl_handler (int 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 {systemId |
  publicId}>
```

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 (int parser, string notationName, string base, string
systemId, string publicId)

parser

The first parameter, *parser*, is a reference to the XML parser calling the handler.

notationName

This is the notation's *name*, as per the notation format described above.

base

This is the base for resolving the system identifier (*systemId*) of the notation declaration. Currently this parameter will always be set to an empty string.

systemId

System identifier of the external notation declaration.

publicId

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.

There is currently no support for object/method handlers.

[Zurück](#)

xml_set_unparsed_entity_decl_handler

[Anfang](#)

[Hoch](#)

[Vor](#)

xml_set_external_entity_ref_handler

xml_set_external_entity_ref_handler

(PHP3 >= 3.0.6, PHP4)

xml_set_external_entity_ref_handler -- set up external entity reference handler

Description

`int xml_set_external_entity_ref_handler (int 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*.

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.

`int handler (int parser, string openEntityNames, string base, string
systemId, string publicId)
parser`

The first parameter, *parser*, is a reference to the XML parser calling the handler.

openEntityNames

The second parameter, *openEntityNames*, 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 (*systemid*) of the external entity. Currently this parameter will always be set to an empty string.

systemId

The fourth parameter, *systemId*, is the system identifier as specified in the entity declaration.

publicId

The fifth parameter, *publicId*, 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.

There is currently no support for object/method handlers.

[Zurück](#)

xml_set_notation_decl_handler

[Anfang](#)

[Hoch](#)

[Vor](#)

xml_parse

xml_parse

(PHP3 >= 3.0.6, PHP4)

xml_parse -- start parsing an XML document

Description

int **xml_parse** (int parser, string data [, int isFinal])

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 *isFinal* parameter is set and true when the last data is parsed.

isFinal (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_get_error_code

(PHP3 >= 3.0.6, PHP4)

xml_get_error_code -- get XML parser error code

Description

```
int xml_get_error_code (int 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_error_string

(PHP3 >= 3.0.6, PHP4)

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_line_number

(PHP3 >= 3.0.6, PHP4)

xml_get_current_line_number -- get current line number for an XML parser

Description

```
int xml_get_current_line_number (int 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.

[Zurück](#)[Anfang](#)[Vor](#)[xml_error_string](#)[Hoch](#)[xml_get_current_column_number](#)

xml_get_current_column_number

(PHP3 >= 3.0.6, PHP4)

xml_get_current_column_number -- Get current column number for an XML parser

Description

```
int xml_get_current_column_number (int 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_byte_index

(PHP3 >= 3.0.6, PHP4)

xml_get_current_byte_index -- get current byte index for an XML parser

Description

```
int xml_get_current_byte_index (int 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).

[Zurück](#)[Anfang](#)[Vor](#)

xml_get_current_column_number

[Hoch](#)

xml_parser_free

xml_parser_free

(PHP3 >= 3.0.6, PHP4)

xml_parser_free -- Free an XML parser

Description

```
string xml_parser_free (int 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_set_option

(PHP3 >= 3.0.6, PHP4)

xml_parser_set_option -- set options in an XML parser

Description

int xml_parser_set_option (int 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:

Tabelle 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_parser_get_option

(PHP3 >= 3.0.6, PHP4)

xml_parser_get_option -- get options from an XML parser

Description

mixed **xml_parser_get_option** (int *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.

utf8_decode

(PHP3 >= 3.0.6, PHP4)

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 explanation of UTF-8 encoding.

utf8_encode

(PHP3 >= 3.0.6, PHP4)

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:

Tabelle 1 UTF-8 encoding

bytes	bits	representation
1	7	0bbbbbbb
2	11	110bbbb 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.

LXIV. NIS functions

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 [Introduction to YP/NIS](#). There is also a book called [Managing NFS and NIS](#) by Hal Stern.

To get these functions to work, you have to configure PHP with `--with-yp`.

Inhaltsverzeichnis

[yp_get_default_domain](#) Fetches the machine's default NIS domain.

[yp_order](#) Returns the order number for a map.

[yp_master](#) Returns the machine name of the master NIS server for a map.

[yp_match](#) Returns the matched line.

[yp_first](#) Returns the first key-value pair from the named map.

[yp_next](#) Returns the next key-value pair in the named map.

yp_get_default_domain

(PHP3 >= 3.0.7, PHP4)

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.

Beispiel 1 Example for the default domain

```
<?php
    $domain = yp_get_default_domain();
    echo "Default NIS domain is: " . $domain;
?>
```

yp_order

(PHP3 >= 3.0.7, PHP4)

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.

Beispiel 1 Example for the NIS order

```
<?php
    $number = yp_order($domain,$mapname);
    echo "Order number for this map is: " . $order;
?>
```

See also [yp-get-default-domain\(\)](#).

yp_master

(PHP3 >= 3.0.7, PHP4)

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.

Beispiel 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

(PHP3 >= 3.0.7, PHP4)

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.

Beispiel 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_first

(PHP3 >= 3.0.7, PHP4)

yp_first -- Returns the first key-value pair from the named map.

Description

string[] **yp_first** (string domain, string map)

yp_first() returns the first key-value pair from the named map in the named domain, otherwise FALSE.

Beispiel 1 Example for the NIS first

```
<?php
    $entry = yp_first($domain, "passwd.byname");
    $key = key($entry);
    echo "First entry in this map has key " . $key
        . " and value " . $entry[$key];
?>
```

See also [yp-get-default-domain\(\)](#)

yp_next

(PHP3 >= 3.0.7, PHP4)

yp_next -- Returns the next key-value pair in the named map.

Description

string[] **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.

Beispiel 1 Example for NIS next

```
<?php
    $entry = yp_next($domain, "passwd.byname", "joe");

    if(!$entry) {
        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\(\)](#).

LXV. Compression functions

This module uses the functions of [zlib](#) by Jean-loup Gailly and Mark Adler to transparently read and write gzip (.gz) compressed files. You have to use a zlib version $\geq 1.0.9$ with this module.

This module contains versions of most of the [filesystem](#) functions which work with gzip-compressed files (and uncompressed files, too, but not with sockets).

Small code example

Opens a temporary file and writes a test string to it, then it prints out the content of this file twice.

Beispiel 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, 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";
?>
```

Inhaltsverzeichnis

[gzclose](#) close an open gz-file pointer

[gzeof](#) test for end-of-file on a gz-file pointer

[gzfile](#) read entire gz-file into an array

[gzgetc](#) get character from gz-file pointer

[gzgets](#) get line from file pointer

[gzgetss](#) get line from gz-file pointer and strip HTML tags

[gzopen](#) open gz-file

[gzpassthru](#) output all remaining data on a gz-file pointer

[gzputs](#) write to a gz-file pointer

[gzread](#) Binary-safe gz-file read

[gzrewind](#) rewind the position of a gz-file pointer

[gzseek](#) seek on a gz-file pointer

[gztell](#) tell gz-file pointer read/write position

[gzwrite](#) Binary-safe gz-file write

[readgzfile](#) output a gz-file

Zurück	Anfang	Vor
yp_next	Hoch	gzclose

gzclose

(PHP3 , PHP4)

gzclose -- close an open gz-file pointer

Description

int **gzclose** (int zp)

The gz-file pointed to by zp is closed.

Returns true on success and false on failure.

The gz-file pointer must be valid, and must point to a file successfully opened by [gzopen\(\)](#).

gzeof

(PHP3 , PHP4)

gzeof -- test for end-of-file on a gz-file pointer

Description

```
int gzeof (int 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

(PHP3 , PHP4)

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

(PHP3 , PHP4)

gzgetc -- get character from gz-file pointer

Description

string **gzgetc** (int 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

(PHP3 , PHP4)

gzgets -- get line from file pointer

Description

string **gzgets** (int 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

(PHP3 , PHP4)

gzgetss -- get line from gz-file pointer and strip HTML tags

Description

string **gzgetss** (int 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.

Anmerkung: *allowable_tags* was added in PHP 3.0.13, PHP4B3.

See also [gzgets\(\)](#), [gzopen\(\)](#), and [strip_tags\(\)](#).

gzopen

(PHP3 , PHP4)

gzopen -- open gz-file

Description

int 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.

Beispiel 1 gzopen() example

```
$fp = gzopen("/tmp/file.gz", "r");
```

See also [gzclose\(\)](#).

gzpassthru

(PHP3 , PHP4)

gzpassthru -- output all remaining data on a gz-file pointer

Description

`int gzpassthru (int 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

(PHP3 , PHP4)

gzputs -- write to a gz-file pointer

Description

```
int gzputs (int zp, string str [, int length])
```

gzputs() is an alias to [gzwrite\(\)](#), and is identical in every way.

gzread

(PHP3 , PHP4)

gzread -- Binary-safe gz-file read

Description

string **gzread** (int *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

(PHP3 , PHP4)

gzrewind -- rewind the position of a gz-file pointer

Description

```
int gzrewind (int 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

(PHP3 , PHP4)

gzseek -- seek on a gz-file pointer

Description

```
int gzseek (int 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

(PHP3 , PHP4)

gztell -- tell gz-file pointer read/write position

Description

`int gztell (int 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\(\)](#).

gzwrite

(PHP3 , PHP4)

gzwrite -- Binary-safe gz-file write

Description

`int gzwrite (int 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.

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

(PHP3 , PHP4)

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 Anhang

Inhaltsverzeichnis

A [Migrating from PHP/FI 2.0 to PHP 3.0](#)

B [PHP development](#)

C [The PHP Debugger](#)

[Zurück](#)[Anfang](#)[Vor](#)

readgzfile

Migrating from PHP/FI 2.0 to
PHP 3.0

Anhang A Migrating from PHP/FI 2.0 to PHP 3.0

Inhaltsverzeichnis

[About the incompatibilities in 3.0](#)[Start/end tags](#)[if..endif syntax](#)[while syntax](#)[Expression types](#)[Error messages have changed](#)[Short-circuited boolean evaluation](#)[Function true/false return values](#)[Other incompatibilities](#)

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 `converter` 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:

Beispiel A-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:

Beispiel A-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:

Beispiel A-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:

Beispiel A-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:

Beispiel A-5 Migration: old if..endif syntax

```
if ($foo);  
    echo "yep\n";  
elseif ($bar);  
    echo "almost\n";  
else;  
    echo "nope\n";  
endif;
```

Beispiel A-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:

Beispiel A-7 Migration: old while..endwhile syntax

```
while ($more_to_come);  
    ...  
endwhile;
```

Beispiel A-8 Migration: new while..endwhile syntax

```
while ($more_to_come):  
    ...  
endwhile;
```

Warnung
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.

[Zurück](#)

Expression types

[Anfang](#)

[Hoch](#)

[Vor](#)

Short-circuited boolean evaluation

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.

Beispiel A-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;
```

Beispiel A-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.

Beispiel A-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.

[Zurück](#)

Function true/false return values

[Anfang](#)

[Hoch](#)

[Vor](#)

PHP development

Anhang B PHP development

Inhaltsverzeichnis

[Adding functions to PHP3](#)[Calling User Functions](#)[Reporting Errors](#)

Adding functions to PHP3

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:

Beispiel B-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:

Beispiel B-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:

Tabelle B-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", PHP3 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) [\[1\]](#)
- SET_VAR_DOUBLE(name,value)
- SET_VAR_LONG(name,value)

[\[1\]](#)

Symbol tables in PHP 3.0 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.

Beispiel B-3 Checking whether \$foo exists in a symbol table

```
if (hash_exists(active_symbol_table, "foo", sizeof("foo"))) { exists... }
else { doesn't exist }
```

Beispiel B-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 3.0 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:

Beispiel B-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:

Beispiel B-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 strdup() 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 3.0 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:

Beispiel B-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;
```

Beispiel B-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... */
```

Beispiel B-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 3.0 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

destructotion 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 PHP3 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`.

Fußnoten

- [1] Be careful here. 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`.

[Zurück](#)[Other incompatibilities](#)[Anfang](#)[Hoch](#)[Vor](#)[Calling User Functions](#)

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.

[Zurück](#)[PHP development](#)[Anfang](#)[Hoch](#)[Vor](#)[Reporting Errors](#)

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.

[Zurück](#)[Calling User Functions](#)[Anfang](#)[Hoch](#)[Vor](#)[The PHP Debugger](#)

Anhang C The PHP Debugger

Inhaltsverzeichnis

[Using the Debugger](#)

[Debugger Protocol](#)

Using the Debugger

PHP's internal debugger is useful for tracking down evasive bugs. The debugger works by connecting to a TCP port for every time PHP 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 them turned off with [error_reporting\(\)](#)*.

[Zurück](#)

Reporting Errors

[Anfang](#)[Hoch](#)[Vor](#)

Debugger Protocol

Debugger Protocol

The 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 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 script that generated this error.

type

Type of line. Tells the receiving program about what it should treat the following data as:

Tabelle C-1 Debugger Line Types

Name	Meaning
<code>start</code>	Tells the receiving program that a debugger message starts here. The contents of <i>data</i> will be the type of error message, listed below.
<code>message</code>	The PHP error message.
<code>location</code>	File name and line number where the error occurred. The first <code>location</code> line will always contain the top-level location. <i>data</i> will contain <i>file: line</i> . There will always be a <code>location</code> line after <code>message</code> and after every <code>function</code> .

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.

Tabelle C-2 Debugger Error Types

Debugger	PHP 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)

Beispiel C-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

```

[Zurück](#)

[Anfang](#)

The PHP Debugger

[Hoch](#)