Date and Time

Introduction

These functions allow you to get the date and time from the server where your PHP scripts are running. You can use these functions to format the date and time in many different ways.

Note
Please keep in mind that these functions are dependent on the locale settings of your server. Make sure to take daylight saving time (use e.g. *$date = strtotime('+7 days', $date)* and not *$date += 7*24*60*60*) and leap years into consideration when working with these functions.

Note
The timezones referenced in this section can be found in the List of Supported Timezones.

Installing/Configuring

Requirements

No external libraries are needed to build this extension.

Installation

There is no installation needed to use these functions; they are part of the PHP core.

Note
Getting the latest timezone database The latest version of the timezone database can be installed via PECL's » timezonedb. For Windows users, a pre-compiled DLL can be downloaded from the PECL4Win site: » php_timezonedb.dll.

Note
Experimental DateTime support in PHP 5.1.x Although the DateTime class (and related functions) are enabled by default since PHP 5.2.0, it is possible to add experimental support into PHP 5.1.x by using the following flag before configure/compile: *CFLAGS=-DEXPERIMENTAL_DATE_SUPPORT=1*

Runtime Configuration

The behaviour of these functions is affected by settings in *php.ini*.

Date/Time Configuration Options

Name	Default	Changeable	Changelog
date.default_latitude	"31.7667"	PHP_INI_ALL	Available since PHP 5.0.0.
date.default_longitude	"35.2333"	PHP_INI_ALL	Available since PHP 5.0.0.

date.sunrise_zenith	"90.583333"	PHP_INI_ALL	Available since PHP 5.0.0.
date.sunset_zenith	"90.583333"	PHP_INI_ALL	Available since PHP 5.0.0.
date.timezone	""	PHP_INI_ALL	Available since PHP 5.1.0.

For further details and definitions of the PHP_INI_* constants, see the php.ini directives.

Here's a short explanation of the configuration directives.

date.default_latitude float
The default latitude.

date.default_longitude float
The default longitude.

date.sunrise_zenith float
The default sunrise zenith.

date.sunset_zenith float
The default sunset zenith.

date.timezone string
The default timezone used by all date/time functions if the *TZ* environment variable isn't set. The precedence order is described in the date_default_timezone_get() page. See List of Supported Timezones for a list of supported timezones.

Note

The first four configuration options are currently only used by date_sunrise() and date_sunset().

Resource Types

This extension has no resource types defined.

Predefined Constants

The following constants are defined since PHP 5.1.1 and they offer standard date representations, which can be used along with the date format functions (like date()).

DATE_ATOM (string)
Atom (example: 2005-08-15T15:52:01+00:00)

DATE_COOKIE (string)
HTTP Cookies (example: Monday, 15-Aug-05 15:52:01 UTC)

DATE_ISO8601 (string)
ISO-8601 (example: 2005-08-15T15:52:01+0000)

DATE_RFC822 (string)
RFC 822 (example: Mon, 15 Aug 05 15:52:01 +0000)

DATE_RFC850 (string)
RFC 850 (example: Monday, 15-Aug-05 15:52:01 UTC)

DATE_RFC1036 (string)
RFC 1036 (example: Mon, 15 Aug 05 15:52:01 +0000)

DATE_RFC1123 (string)
RFC 1123 (example: Mon, 15 Aug 2005 15:52:01 +0000)

DATE_RFC2822 (string)
RFC 2822 (Mon, 15 Aug 2005 15:52:01 +0000)

DATE_RFC3339 (string)
Same as **DATE_ATOM** (since PHP 5.1.3)

DATE_RSS (string)
RSS (Mon, 15 Aug 2005 15:52:01 +0000)

DATE_W3C (string)
World Wide Web Consortium (example: 2005-08-15T15:52:01+00:00)

Following constants exists since PHP 5.1.2 and specify a format returned by functions date_sunrise() and date_sunset().

SUNFUNCS_RET_TIMESTAMP (integer)
Timestamp

SUNFUNCS_RET_STRING (integer)
Hours:minutes (example: 08:02)

SUNFUNCS_RET_DOUBLE (integer)
Hours as floating point number (example 8.75)

List of Supported Timezones

Here you'll find the complete list of timezones supported by PHP, which are meant to be used with e.g. date_default_timezone_set().

Note
The latest version of the timezone database can be installed via PECL's » timezonedb. For Windows users, a pre-compiled DLL can be downloaded from the PECL4Win site: » php_timezonedb.dll.

Africa

Africa

Africa/Abidjan	Africa/Accra	Africa/Addis_Ababa	Africa/Algiers	Africa/Asmara
Africa/Asmera	Africa/Bamako	Africa/Bangui	Africa/Banjul	Africa/Bissau
Africa/Blantyre	Africa/Brazzaville	Africa/Bujumbura	Africa/Cairo	Africa/Casablanca
Africa/Ceuta	Africa/Conakry	Africa/Dakar	Africa/Dar_es_Salaam	Africa/Djibouti
Africa/Douala	Africa/El_Aaiun	Africa/Freetown	Africa/Gaborone	Africa/Harare
Africa/Johannesburg	Africa/Kampala	Africa/Khartoum	Africa/Kigali	Africa/Kinshasa
Africa/Lagos	Africa/Libreville	Africa/Lome	Africa/Luanda	Africa/Lubumbashi
Africa/Lusaka	Africa/Malabo	Africa/Maputo	Africa/Maseru	Africa/Mbabane
Africa/Mogadishu	Africa/Monrovia	Africa/Nairobi	Africa/Ndjamena	Africa/Niamey
Africa/Nouakchott	Africa/Ouagadougou	Africa/Porto-Novo	Africa/Sao_Tome	Africa/Timbuktu
Africa/Tripoli	Africa/Tunis	Africa/Windhoek		

America

America

America/Adak	America/Anchorage	America/Anguilla	America/Antigua	America/Araguaina
America/Argentina/Buenos_Aires	America/Argentina/Catamarca	America/Argentina/ComodRivadavia	America/Argentina/Cordoba	America/Argentina/Jujuy
America/Argentina/La_Rioja	America/Argentina/Mendoza	America/Argentina/Rio_Gallegos	America/Argentina/San_Juan	America/Argentina/San_Luis
America/Argentina/Tucuman	America/Argentina/Ushuaia	America/Aruba	America/Asuncion	America/Atikokan
America/Atka	America/Bahia	America/Barbados	America/Belem	America/Belize
America/Blanc-Sablon	America/Boa_Vista	America/Bogota	America/Boise	America/Buenos_Aires
America/Cambridge_Bay	America/Campo_Grande	America/Cancun	America/Caracas	America/Catamarca
America/Cayenne	America/Cayman	America/Chicago	America/Chihuahua	America/Coral_Harbour
America/Cordoba	America/Costa_Rica	America/Cuiaba	America/Curacao	America/Danmarkshavn
America/Dawson	America/Dawson_Creek	America/Denver	America/Detroit	America/Dominica
America/Edmonton	America/Eirunepe	America/El_Salvador	America/Ensenada	America/Fort_Wayne
America/Fortaleza	America/Glace_Bay	America/Godthab	America/Goose_Bay	America/Grand_Turk
America/Grenada	America/Guadeloupe	America/Guatemala	America/Guayaquil	America/Guyana
America/Halifax	America/Havana	America/Hermosillo	America/Indiana/Indianapolis	America/Indiana/Knox
America/Indiana/Marengo	America/Indiana/Petersburg	America/Indiana/Tell_City	America/Indiana/Vevay	America/Indiana/Vincennes

America/Indiana/Winamac	America/Indianapolis	America/Inuvik	America/Iqaluit	America/Jamaica
America/Jujuy	America/Juneau	America/Kentucky/Louisville	America/Kentucky/Monticello	America/Knox_IN
America/La_Paz	America/Lima	America/Los_Angeles	America/Louisville	America/Maceio
America/Managua	America/Manaus	America/Marigot	America/Martinique	America/Mazatlan
America/Mendoza	America/Menominee	America/Merida	America/Mexico_City	America/Miquelon
America/Moncton	America/Monterrey	America/Montevideo	America/Montreal	America/Montserrat
America/Nassau	America/New_York	America/Nipigon	America/Nome	America/Noronha
America/North_Dakota/Center	America/North_Dakota/New_Salem	America/Panama	America/Pangnirtung	America/Paramaribo
America/Phoenix	America/Port-au-Prince	America/Port_of_Spain	America/Porto_Acre	America/Porto_Velho
America/Puerto_Rico	America/Rainy_River	America/Rankin_Inlet	America/Recife	America/Regina
America/Resolute	America/Rio_Branco	America/Rosario	America/Santiago	America/Santo_Domingo
America/Sao_Paulo	America/Scoresbysund	America/Shiprock	America/St_Barthelemy	America/St_Johns
America/St_Kitts	America/St_Lucia	America/St_Thomas	America/St_Vincent	America/Swift_Current
America/Tegucigalpa	America/Thule	America/Thunder_Bay	America/Tijuana	America/Toronto
America/Tortola	America/Vancouver	America/Virgin	America/Whitehorse	America/Winnipeg
America/Yakutat	America/Yellowknife			

Antarctica

Antarctica

Antarctica/Casey	Antarctica/Davis	Antarctica/DumontDUrville	Antarctica/Mawson	Antarctica/McMurdo
Antarctica/Palmer	Antarctica/Rothera	Antarctica/South_Pole	Antarctica/Syowa	Antarctica/Vostok

Arctic

Arctic

Arctic/Longyearbyen

Asia

Asia

Asia/Aden	Asia/Almaty	Asia/Amman	Asia/Anadyr	Asia/Aqtau
Asia/Aqtobe	Asia/Ashgabat	Asia/Ashkhabad	Asia/Baghdad	Asia/Bahrain
Asia/Baku	Asia/Bangkok	Asia/Beirut	Asia/Bishkek	Asia/Brunei
Asia/Calcutta	Asia/Choibalsan	Asia/Chongqing	Asia/Chungking	Asia/Colombo
Asia/Dacca	Asia/Damascus	Asia/Dhaka	Asia/Dili	Asia/Dubai
Asia/Dushanbe	Asia/Gaza	Asia/Harbin	Asia/Ho_Chi_Minh	Asia/Hong_Kong
Asia/Hovd	Asia/Irkutsk	Asia/Istanbul	Asia/Jakarta	Asia/Jayapura
Asia/Jerusalem	Asia/Kabul	Asia/Kamchatka	Asia/Karachi	Asia/Kashgar
Asia/Katmandu	Asia/Kolkata	Asia/Krasnoyarsk	Asia/Kuala_Lumpur	Asia/Kuching
Asia/Kuwait	Asia/Macao	Asia/Macau	Asia/Magadan	Asia/Makassar
Asia/Manila	Asia/Muscat	Asia/Nicosia	Asia/Novosibirsk	Asia/Omsk
Asia/Oral	Asia/Phnom_Penh	Asia/Pontianak	Asia/Pyongyang	Asia/Qatar

Asia/Qyzylorda	Asia/Rangoon	Asia/Riyadh	Asia/Saigon	Asia/Sakhalin
Asia/Samarkand	Asia/Seoul	Asia/Shanghai	Asia/Singapore	Asia/Taipei
Asia/Tashkent	Asia/Tbilisi	Asia/Tehran	Asia/Tel_Aviv	Asia/Thimbu
Asia/Thimphu	Asia/Tokyo	Asia/Ujung_Pandang	Asia/Ulaanbaatar	Asia/Ulan_Bator
Asia/Urumqi	Asia/Vientiane	Asia/Vladivostok	Asia/Yakutsk	Asia/Yekaterinburg
Asia/Yerevan				

Atlantic

Atlantic

Atlantic/Azores	Atlantic/Bermuda	Atlantic/Canary	Atlantic/Cape_Verde	Atlantic/Faeroe
Atlantic/Faroe	Atlantic/Jan_Mayen	Atlantic/Madeira	Atlantic/Reykjavik	Atlantic/South_Georgia
Atlantic/St_Helena	Atlantic/Stanley			

Australia

Australia

Australia/ACT	Australia/Adelaide	Australia/Brisbane	Australia/Broken_Hill	Australia/Canberra
Australia/Currie	Australia/Darwin	Australia/Eucla	Australia/Hobart	Australia/LHI
Australia/Lindeman	Australia/Lord_Howe	Australia/Melbourne	Australia/North	Australia/NSW
Australia/Perth	Australia/Queensland	Australia/South	Australia/Sydney	Australia/Tasmania
Australia/Victoria	Australia/West	Australia/Yancowinna		

Europe

Europe

Europe/Amsterdam	Europe/Andorra	Europe/Athens	Europe/Belfast	Europe/Belgrade
Europe/Berlin	Europe/Bratislava	Europe/Brussels	Europe/Bucharest	Europe/Budapest
Europe/Chisinau	Europe/Copenhagen	Europe/Dublin	Europe/Gibraltar	Europe/Guernsey
Europe/Helsinki	Europe/Isle_of_Man	Europe/Istanbul	Europe/Jersey	Europe/Kaliningrad
Europe/Kiev	Europe/Lisbon	Europe/Ljubljana	Europe/London	Europe/Luxembourg
Europe/Madrid	Europe/Malta	Europe/Mariehamn	Europe/Minsk	Europe/Monaco
Europe/Moscow	Europe/Nicosia	Europe/Oslo	Europe/Paris	Europe/Podgorica
Europe/Prague	Europe/Riga	Europe/Rome	Europe/Samara	Europe/San_Marino
Europe/Sarajevo	Europe/Simferopol	Europe/Skopje	Europe/Sofia	Europe/Stockholm
Europe/Tallinn	Europe/Tirane	Europe/Tiraspol	Europe/Uzhgorod	Europe/Vaduz
Europe/Vatican	Europe/Vienna	Europe/Vilnius	Europe/Volgograd	Europe/Warsaw
Europe/Zagreb	Europe/Zaporozhye	Europe/Zurich		

Indian

Indian

Indian/Antananarivo	Indian/Chagos	Indian/Christmas	Indian/Cocos	Indian/Comoro

Indian/Kerguelen	Indian/Mahe	Indian/Maldives	Indian/Mauritius	Indian/Mayotte
Indian/Reunion				

Pacific

Pacific

Pacific/Apia	Pacific/Auckland	Pacific/Chatham	Pacific/Easter	Pacific/Efate
Pacific/Enderbury	Pacific/Fakaofo	Pacific/Fiji	Pacific/Funafuti	Pacific/Galapagos
Pacific/Gambier	Pacific/Guadalcanal	Pacific/Guam	Pacific/Honolulu	Pacific/Johnston
Pacific/Kiritimati	Pacific/Kosrae	Pacific/Kwajalein	Pacific/Majuro	Pacific/Marquesas
Pacific/Midway	Pacific/Nauru	Pacific/Niue	Pacific/Norfolk	Pacific/Noumea
Pacific/Pago_Pago	Pacific/Palau	Pacific/Pitcairn	Pacific/Ponape	Pacific/Port_Moresby
Pacific/Rarotonga	Pacific/Saipan	Pacific/Samoa	Pacific/Tahiti	Pacific/Tarawa
Pacific/Tongatapu	Pacific/Truk	Pacific/Wake	Pacific/Wallis	Pacific/Yap

Others

Others

Brazil/Acre	Brazil/DeNoronha	Brazil/East	Brazil/West	Canada/Atlantic
Canada/Central	Canada/East-Saskatchewan	Canada/Eastern	Canada/Mountain	Canada/Newfoundland
Canada/Pacific	Canada/Saskatchewan	Canada/Yukon	CET	Chile/Continental
Chile/EasterIsland	CST6CDT	Cuba	EET	Egypt

Eire	EST	EST5EDT	Etc/GMT	Etc/GMT+0
Etc/GMT+1	Etc/GMT+10	Etc/GMT+11	Etc/GMT+12	Etc/GMT+2
Etc/GMT+3	Etc/GMT+4	Etc/GMT+5	Etc/GMT+6	Etc/GMT+7
Etc/GMT+8	Etc/GMT+9	Etc/GMT-0	Etc/GMT-1	Etc/GMT-10
Etc/GMT-11	Etc/GMT-12	Etc/GMT-13	Etc/GMT-14	Etc/GMT-2
Etc/GMT-3	Etc/GMT-4	Etc/GMT-5	Etc/GMT-6	Etc/GMT-7
Etc/GMT-8	Etc/GMT-9	Etc/GMT0	Etc/Greenwich	Etc/UCT
Etc/Universal	Etc/UTC	Etc/Zulu	Factory	GB
GB-Eire	GMT	GMT+0	GMT-0	GMT0
Greenwich	Hongkong	HST	Iceland	Iran
Israel	Jamaica	Japan	Kwajalein	Libya
MET	Mexico/BajaNorte	Mexico/BajaSur	Mexico/General	MST
MST7MDT	Navajo	NZ	NZ-CHAT	Poland
Portugal	PRC	PST8PDT	ROC	ROK
Singapore	Turkey	UCT	Universal	US/Alaska
US/Aleutian	US/Arizona	US/Central	US/East-Indiana	US/Eastern
US/Hawaii	US/Indiana-Starke	US/Michigan	US/Mountain	US/Pacific
US/Pacific-New	US/Samoa	UTC	W-SU	WET
Zulu				

Warning
Please do not use any of the timezones listed here (besides UTC), they only exist for backward compatible reasons.

Date/Time Functions

checkdate

checkdate -- Validate a Gregorian date

Description

bool **checkdate** (int `$month`, int `$day`, int `$year`)

Checks the validity of the date formed by the arguments. A date is considered valid if each parameter is properly defined.

Parameters

`month`
The month is between 1 and 12 inclusive.

`day`
The day is within the allowed number of days for the given *`month`*. Leap *`year`* s are taken into consideration.

`year`
The year is between 1 and 32767 inclusive.

Return Values

Returns **TRUE** if the date given is valid; otherwise returns **FALSE**.

Examples

Example #1 - checkdate() example

```
<?php
var_dump(checkdate(12, 31, 2000));
var_dump(checkdate(2, 29, 2001));
?>
```

The above example will output:

```
bool(true)
bool(false)
```

See Also

- mktime()
- strtotime()

date_create

date_create -- Returns new DateTime object

Description

DateTime **date_create** ([string `$time` [, DateTimeZone `$timezone`]])

DateTime **DateTime::__construct** ([string `$time` [, DateTimeZone `$timezone`]])

Parameters

`time`
String in a format accepted by strtotime(), defaults to "now".

`timezone`
Time zone of the time.

Return Values

Returns DateTime object on success or **FALSE** on failure.

date_date_set

date_date_set -- Sets the date

Description

void **date_date_set** (DateTime `$object`, int `$year`, int `$month`, int `$day`)

void **DateTime::setDate** (int `$year`, int `$month`, int `$day`)

Parameters

`object`
: DateTime object.

`year`
: Year of the date.

`month`
: Month of the date.

`day`
: Day of the date.

Return Values

Returns **NULL** on success or **FALSE** on failure.

See Also

- date_isodate_set()
- date_time_set()

date_default_timezone_get

date_default_timezone_get -- Gets the default timezone used by all date/time functions in a script

Description

string **date_default_timezone_get** (void)

This functions returns the default timezone, using the following "guess" order:

- The timezone set using the date_default_timezone_set() function (if any)
- The *TZ* environment variable (if non empty)
- The date.timezone ini option (if set)
- "magical" guess (if the operating system supports it)
- If none of the above options succeeds, return *UTC*

Return Values

Returns a string.

See Also

- date_default_timezone_set()

date_default_timezone_set

date_default_timezone_set -- Sets the default timezone used by all date/time functions in a script

Description

bool **date_default_timezone_set** (string `$timezone_identifier`)

date_default_timezone_set() sets the default timezone used by all date/time functions.

Note
Since PHP 5.1.0 (when the date/time functions were rewritten), every call to a date/time function will generate a **E_NOTICE** if the timezone isn't valid, and/or a **E_STRICT** message if using the system settings or the *TZ* environment variable.

Instead of using this function to set the default timezone in your script, you can also use the INI setting date.timezone to set the default timezone.

Parameters

`timezone_identifier`
: The timezone identifier, like *UTC* or *Europe/Lisbon*. The list of valid identifiers is available in the List of Supported Timezones.

Return Values

This function returns **FALSE** if the *`timezone_identifier`* isn't valid, or **TRUE** otherwise.

ChangeLog

Version	Description
5.1.2	The function started to validate the *`timezone_identifier`* parameter.

See Also

- date_default_timezone_get()

date_format

date_format -- Returns date formatted according to given format

Description

string **date_format** (DateTime `$object`, string `$format`)

string **DateTime::format** (string `$format`)

Parameters

`object`
: DateTime object.

`format`
: Format accepted by date().

Return Values

Returns formatted date on success or **FALSE** on failure.

See Also

- date()

date_isodate_set

date_isodate_set -- Sets the ISO date

Description

void **date_isodate_set** (DateTime `$object`, int `$year`, int `$week` [, int `$day`])

void **DateTime::setISODate** (int `$year`, int `$week` [, int `$day`])

Parameters

`object`
: DateTime object.

`year`
: Year of the date.

`week`
: Week of the date.

`day`
: Day of the date.

Return Values

Returns **NULL** on success or **FALSE** on failure.

See Also

- date_date_set()

date_modify

date_modify -- Alters the timestamp

Description

void **date_modify** (DateTime `$object`, string `$modify`)

void **DateTime::modify** (string `$modify`)

Parameters

`object`
DateTime object.

`modify`
String in a relative format accepted by strtotime().

Return Values

Returns **NULL** on success or **FALSE** on failure.

Examples

Example #2 - A date_modify() example

```
<?php
$date = new DateTime("2006-12-12");
$date->modify("+1 day");
echo $date->format("Y-m-d");
?>
```

The above example will output:

```
2006-12-13
```

See Also

- strtotime()

date_offset_get

date_offset_get -- Returns the daylight saving time offset

Description

int **date_offset_get** (DateTime `$object`)

int **DateTime::getOffset** (void)

Parameters

`object`
DateTime object.

Return Values

Returns DST offset in seconds on success or **FALSE** on failure.

date_parse

date_parse -- Returns associative array with detailed info about given date

Description

array **date_parse** (string `$date`)

Parameters

`date`
Date in format accepted by strtotime().

Return Values

Returns array with information about the parsed date on success, or **FALSE** on failure.

Errors/Exceptions

In case the date format has an error, the element 'errors' will contains the error messages.

Examples

Example #3 - A date_parse() example

```
<?php
print_r(date_parse("2006-12-12 10:00:00.5"));
?>
```

The above example will output:

```
Array
(
   [year] => 2006
   [month] => 12
   [day] => 12
   [hour] => 10
   [minute] => 0
   [second] => 0
   [fraction] => 0.5
   [warning_count] => 0
   [warnings] => Array()
   [error_count] => 0
   [errors] => Array()
   [is_localtime] =>
)
```

See Also

- getdate()

date_sun_info

date_sun_info -- Returns an array with information about sunset/sunrise and twilight begin/end

Description

array **date_sun_info** (int `$time`, float `$latitude`, float `$longitude`)

Parameters

`time`
Timestamp.

`latitude`
Latitude in degrees.

`longitude`
Longitude in degrees.

Return Values

Returns array on success or **FALSE** on failure.

Examples

Example #4 - A date_sun_info() example

```
<?php
$sun_info = date_sun_info(strtotime("2006-12-12"), 31.7667, 35.2333);
foreach ($sun_info as $key => $val) {
   echo "$key: " . date("H:i:s", $val) . "\n";
}
?>
```

The above example will output:

```
sunrise: 05:52:11
sunset: 15:41:21
transit: 10:46:46
civil_twilight_begin: 05:24:08
civil_twilight_end: 16:09:24
nautical_twilight_begin: 04:52:25
nautical_twilight_end: 16:41:06
astronomical_twilight_begin: 04:21:32
astronomical_twilight_end: 17:12:00
```

See Also

- date_sunrise()
- date_sunset()

date_sunrise

date_sunrise -- Returns time of sunrise for a given day and location

Description

mixed **date_sunrise** (int `$timestamp` [, int `$format` [, float `$latitude` [, float `$longitude` [, float `$zenith` [, float `$gmt_offset`]]]]])

date_sunrise() returns the sunrise time for a given day (specified as a *`timestamp`*) and location.

Parameters

`timestamp`
The *`timestamp`* of the day from which the sunrise time is taken.

`format`

`format` **constants**

constant	description	example
SUNFUNCS_RET_STRING	returns the result as string	16:46
SUNFUNCS_RET_DOUBLE	returns the result as float	16.782431
SUNFUNCS_RET_TIMESTAMP	returns the result as integer (timestamp)	10950346

`latitude`
Defaults to North, pass in a negative value for South. See also: *date.default_latitude*

`longitude`
Defaults to East, pass in a negative value for West. See also: *date.default_longitude*

`zenith`
Default: *date.sunrise_zenith*

`gmtoffset`
Specified in hours.

Return Values

Returns the sunrise time in a specified *`format`* on success, or **FALSE** on failure.

Errors/Exceptions

Every call to a date/time function will generate a **E_NOTICE** if the time zone is not valid, and/or a **E_STRICT** message if using the system settings or the *TZ* environment variable. See also date_default_timezone_set()

ChangeLog

Version	Description
5.1.0	Now issues the **E_STRICT** and **E_NOTICE** time zone errors.

Examples

Example #5 - date_sunrise() example

```
<?php

/* calculate the sunrise time for Lisbon, Portugal
Latitude: 38.4 North
Longitude: 9 West
Zenith ~= 90
offset: +1 GMT
*/

echo date("D M d Y"). ', sunrise time : ' .date_sunrise(time(),
SUNFUNCS_RET_STRING, 38.4, -9, 90, 1);

?>
```

The above example will output something similar to:

```
Mon Dec 20 2004, sunrise time : 08:54
```

See Also

- date_sunset()

date_sunset

date_sunset -- Returns time of sunset for a given day and location

Description

mixed **date_sunset** (int `$timestamp` [, int `$format` [, float `$latitude` [, float `$longitude` [, float `$zenith` [, float `$gmt_offset`]]]]])

date_sunset() returns the sunset time for a given day (specified as a *`timestamp`*) and location.

Parameters

`timestamp`
The *`timestamp`* of the day from which the sunset time is taken.

`format`

`format` **constants**

constant	description	example
SUNFUNCS_RET_STRING	returns the result as string	16:46
SUNFUNCS_RET_DOUBLE	returns the result as float	16.782431
SUNFUNCS_RET_TIMESTAMP	returns the result as integer (timestamp)	10950346

`latitude`
Defaults to North, pass in a negative value for South. See also: *date.default_latitude*

`longitude`
Defaults to East, pass in a negative value for West. See also: *date.default_longitude*

`zenith`
Default: *date.sunrise_zenith*

`gmtoffset`
Specified in hours.

Errors/Exceptions

Every call to a date/time function will generate a **E_NOTICE** if the time zone is not valid,

and/or a **E_STRICT** message if using the system settings or the *TZ* environment variable. See also date_default_timezone_set()

ChangeLog

Version	Description
5.1.0	Now issues the **E_STRICT** and **E_NOTICE** time zone errors.

Return Values

Returns the sunset time in a specified *format* on success, or **FALSE** on failure.

Examples

Example #6 - date_sunset() example

```
<?php

/* calculate the sunset time for Lisbon, Portugal
Latitude: 38.4 North
Longitude: 9 West
Zenith ~= 90
offset: +1 GMT
*/

echo date("D M d Y"). ', sunset time : ' .date_sunset(time(),
SUNFUNCS_RET_STRING, 38.4, -9, 90, 1);

?>
```

The above example will output something similar to:

```
Mon Dec 20 2004, sunset time : 18:13
```

See Also

- date_sunrise()

date_time_set

date_time_set -- Sets the time

Description

void **date_time_set** (DateTime `$object`, int `$hour`, int `$minute` [, int `$second`])

void **DateTime::setTime** (int `$hour`, int `$minute` [, int `$second`])

Parameters

`object`
: DateTime object.

`hour`
: Hour of the time.

`minute`
: Minute of the time.

`second`
: Second of the time.

Return Values

Returns **NULL** on success or **FALSE** on failure.

See Also

- date_date_set()

date_timezone_get

date_timezone_get -- Return time zone relative to given DateTime

Description

DateTimeZone **date_timezone_get** (DateTime `$object`)

DateTimeZone **DateTime::getTimezone** (void)

Parameters

`object`
DateTime object.

Return Values

Returns DateTimeZone object on success or **FALSE** on failure.

See Also

- date_timezone_set()

date_timezone_set

date_timezone_set -- Sets the time zone for the DateTime object

Description

void **date_timezone_set** (DateTime `$object`, DateTimeZone `$timezone`)

void **DateTime::setTimezone** (DateTimeZone `$timezone`)

Parameters

`object`
DateTime object.

`timezone`
Desired time zone.

Return Values

Returns **NULL** on success or **FALSE** on failure.

See Also

- date_timezone_get()

date

date -- Format a local time/date

Description

string **date** (string `$format` [, int `$timestamp`])

Returns a string formatted according to the given format string using the given integer *`timestamp`* or the current time if no timestamp is given. In other words, *`timestamp`* is optional and defaults to the value of time().

Parameters

`format`
The format of the outputted date string. See the formatting options below.

The following characters are recognized in the *`format`* parameter string

`format` **character**	**Description**	**Example returned values**
Day	---	---
d	Day of the month, 2 digits with leading zeros	*01* to *31*
D	A textual representation of a day, three letters	*Mon* through *Sun*
j	Day of the month without leading zeros	*1* to *31*
l (lowercase 'L')	A full textual representation of the day of the week	*Sunday* through *Saturday*
N	ISO-8601 numeric representation of the day of the week (added in PHP 5.1.0)	*1* (for Monday) through *7* (for Sunday)
S	English ordinal suffix for the day of the month, 2 characters	*st*, *nd*, *rd* or *th*. Works well with *j*
w	Numeric representation of the day of the week	*0* (for Sunday) through *6* (for Saturday)
z	The day of the year (starting	*0* through *365*

	from 0)	
Week	---	---
W	ISO-8601 week number of year, weeks starting on Monday (added in PHP 4.1.0)	Example: *42* (the 42nd week in the year)
Month	---	---
F	A full textual representation of a month, such as January or March	*January* through *December*
m	Numeric representation of a month, with leading zeros	*01* through *12*
M	A short textual representation of a month, three letters	*Jan* through *Dec*
n	Numeric representation of a month, without leading zeros	*1* through *12*
t	Number of days in the given month	*28* through *31*
Year	---	---
L	Whether it's a leap year	*1* if it is a leap year, *0* otherwise.
o	ISO-8601 year number. This has the same value as *Y*, except that if the ISO week number (*W*) belongs to the previous or next year, that year is used instead. (added in PHP 5.1.0)	Examples: *1999* or *2003*
Y	A full numeric representation of a year, 4 digits	Examples: *1999* or *2003*
y	A two digit representation of a year	Examples: *99* or *03*
Time	---	---
a	Lowercase Ante meridiem and Post meridiem	*am* or *pm*

A	Uppercase Ante meridiem and Post meridiem	*AM* or *PM*
B	Swatch Internet time	*000* through *999*
g	12-hour format of an hour without leading zeros	*1* through *12*
G	24-hour format of an hour without leading zeros	*0* through *23*
h	12-hour format of an hour with leading zeros	*01* through *12*
H	24-hour format of an hour with leading zeros	*00* through *23*
i	Minutes with leading zeros	*00* to *59*
s	Seconds, with leading zeros	*00* through *59*
u	Milliseconds (added in PHP 5.2.2)	Example: *54321*
Timezone	---	---
e	Timezone identifier (added in PHP 5.1.0)	Examples: *UTC*, *GMT*, *Atlantic/Azores*
I (capital i)	Whether or not the date is in daylight saving time	*1* if Daylight Saving Time, *0* otherwise.
O	Difference to Greenwich time (GMT) in hours	Example: *+0200*
P	Difference to Greenwich time (GMT) with colon between hours and minutes (added in PHP 5.1.3)	Example: *+02:00*
T	Timezone abbreviation	Examples: *EST*, *MDT*...
Z	Timezone offset in seconds. The offset for timezones west of UTC is always negative, and for those east of UTC is always positive.	*-43200* through *50400*
Full Date/Time	---	---
c	ISO 8601 date (added in PHP 5)	2004-02-12T15:19:21+00:00

r	» RFC 2822 formatted date	Example: *Thu, 21 Dec 2000 16:01:07 +0200*
U	Seconds since the Unix Epoch (January 1 1970 00:00:00 GMT)	See also time()

Unrecognized characters in the format string will be printed as-is. The *Z* format will always return *0* when using gmdate().

Note
Since this function only accepts integer timestamps the *u* format character is only useful when using the date_format() function with user based timestamps created with date_create().

`timestamp`
The optional `timestamp` parameter is an integer Unix timestamp that defaults to the current local time if a `timestamp` is not given. In other words, it defaults to the value of time().

Return Values

Returns a formatted date string. If a non-numeric value is used for `timestamp`, **FALSE** is returned and an *E_WARNING* level error is emitted.

Errors/Exceptions

Every call to a date/time function will generate a **E_NOTICE** if the time zone is not valid, and/or a **E_STRICT** message if using the system settings or the *TZ* environment variable. See also date_default_timezone_set()

ChangeLog

Version	Description
5.1.0	The valid range of a timestamp is typically from Fri, 13 Dec 1901 20:45:54 GMT to Tue, 19 Jan 2038 03:14:07 GMT. (These are the dates that correspond to the minimum and maximum values for a 32-bit signed integer). However, before PHP 5.1.0 this range was limited from 01-01-1970 to 19-01-2038 on some systems (e.g. Windows).
5.1.0	Now issues the **E_STRICT** and **E_NOTICE**

	time zone errors.
5.1.1	There are useful constants of standard date/time formats that can be used to specify the *format* parameter.

Examples

Example #7 - date() examples

```
<?php
// set the default timezone to use. Available since PHP 5.1
date_default_timezone_set('UTC');


// Prints something like: Monday
echo date("l");

// Prints something like: Monday 8th of August 2005 03:12:46 PM
echo date('l jS \of F Y h:i:s A');

// Prints: July 1, 2000 is on a Saturday
echo "July 1, 2000 is on a " . date("l", mktime(0, 0, 0, 7, 1, 2000));

/* use the constants in the format parameter */
// prints something like: Mon, 15 Aug 2005 15:12:46 UTC
echo date(DATE_RFC822);

// prints something like: 2000-07-01T00:00:00+00:00
echo date(DATE_ATOM, mktime(0, 0, 0, 7, 1, 2000));
?>
```

You can prevent a recognized character in the format string from being expanded by escaping it with a preceding backslash. If the character with a backslash is already a special sequence, you may need to also escape the backslash.

Example #8 - Escaping characters in date()

```
<?php
// prints something like: Wednesday the 15th
echo date("l \\t\h\e jS");
?>
```

It is possible to use date() and mktime() together to find dates in the future or the past.

Example #9 - date() and mktime() example

```
<?php
$tomorrow  = mktime(0, 0, 0, date("m")  , date("d")+1, date("Y"));
$lastmonth = mktime(0, 0, 0, date("m")-1, date("d"),   date("Y"));
$nextyear  = mktime(0, 0, 0, date("m"),   date("d"),   date("Y")+1);
?>
```

Note

This can be more reliable than simply adding or subtracting the number of seconds in a day or month to a timestamp because of daylight saving time.

Some examples of date() formatting. Note that you should escape any other characters, as any which currently have a special meaning will produce undesirable results, and other characters may be assigned meaning in future PHP versions. When escaping, be sure to use single quotes to prevent characters like \n from becoming newlines.

Example #10 - date() Formatting

```
<?php
// Assuming today is: March 10th, 2001, 5:16:18 pm

$today = date("F j, Y, g:i a");                 // March 10, 2001, 5:16 pm
$today = date("m.d.y");                         // 03.10.01
$today = date("j, n, Y");                       // 10, 3, 2001
$today = date("Ymd");                           // 20010310
$today = date('h-i-s, j-m-y, it is w Day z ');  // 05-16-17, 10-03-01, 1631
1618 6 Fripm01
$today = date('\i\t \i\s \t\h\e jS \d\a\y.');   // It is the 10th day.
$today = date("D M j G:i:s T Y");               // Sat Mar 10 15:16:08 MST
2001
$today = date('H:m:s \m \i\s\ \m\o\n\t\h');     // 17:03:17 m is month
$today = date("H:i:s");                         // 17:16:17
?>
```

To format dates in other languages, you should use the setlocale() and strftime() functions instead of date().

Notes

Note

To generate a timestamp from a string representation of the date, you may be able to use strtotime(). Additionally, some databases have functions to convert their date formats into timestamps (such as MySQL's » UNIX_TIMESTAMP function).

Tip
Timestamp of the start of the request is available in *$_SERVER['REQUEST_TIME']* since PHP 5.1.

See Also

- getlastmod()
- gmdate()
- mktime()
- strftime()
- time()

getdate

getdate -- Get date/time information

Description

array **getdate** ([int `$timestamp`])

Returns an associative array containing the date information of the *`timestamp`*, or the current local time if no *`timestamp`* is given.

Parameters

`timestamp`
: The optional *`timestamp`* parameter is an integer Unix timestamp that defaults to the current local time if a *`timestamp`* is not given. In other words, it defaults to the value of time().

Return Values

Returns an associative array of information related to the *`timestamp`*. Elements from the returned associative array are as follows:

Key elements of the returned associative array

Key	Description	Example returned values
"seconds"	Numeric representation of seconds	*0* to *59*
"minutes"	Numeric representation of minutes	*0* to *59*
"hours"	Numeric representation of hours	*0* to *23*
"mday"	Numeric representation of the day of the month	*1* to *31*
"wday"	Numeric representation of the day of the week	*0* (for Sunday) through *6* (for Saturday)
"mon"	Numeric representation of a month	*1* through *12*

"year"	A full numeric representation of a year, 4 digits	Examples: *1999* or *2003*
"yday"	Numeric representation of the day of the year	*0* through *365*
"weekday"	A full textual representation of the day of the week	*Sunday* through *Saturday*
"month"	A full textual representation of a month, such as January or March	*January* through *December*
0	Seconds since the Unix Epoch, similar to the values returned by time() and used by date().	System Dependent, typically *-2147483648* through *2147483647*.

Examples

Example #11 - getdate() example

```
<?php
$today = getdate();
print_r($today);
?>
```

The above example will output something similar to:

```
Array
(
    [seconds] => 40
    [minutes] => 58
    [hours]   => 21
    [mday]    => 17
    [wday]    => 2
    [mon]     => 6
    [year]    => 2003
    [yday]    => 167
    [weekday] => Tuesday
    [month]   => June
    [0]       => 1055901520
)
```

See Also

- date()
- time()
- setlocale()

gettimeofday

gettimeofday -- Get current time

Description

mixed **gettimeofday** ([bool `$return_float`])

This is an interface to gettimeofday(2). It returns an associative array containing the data returned from the system call.

Parameters

`return_float`
When set to **TRUE**, a float instead of an array is returned.

Return Values

By default an array is returned. If *`return_float`* is set, then a float is returned.

Array keys:

- "sec" - seconds since the Unix Epoch
- "usec" - microseconds
- "minuteswest" - minutes west of Greenwich
- "dsttime" - type of dst correction

ChangeLog

Version	Description
5.1.0	The *`return_float`* parameter was added.

Examples

Example #12 - gettimeofday() example

```
<?php
print_r(gettimeofday());

echo gettimeofday(true);
?>
```

The above example will output something similar to:

```
Array
(
   [sec] => 1073504408
   [usec] => 238215
   [minuteswest] => 0
   [dsttime] => 1
)

1073504408.23910
```

gmdate

gmdate -- Format a GMT/UTC date/time

Description

string **gmdate** (string `$format` [, int `$timestamp`])

Identical to the date() function except that the time returned is Greenwich Mean Time (GMT).

Parameters

`format`
The format of the outputted date string. See the formatting options for the date() function.

`timestamp`
The optional *`timestamp`* parameter is an integer Unix timestamp that defaults to the current local time if a *`timestamp`* is not given. In other words, it defaults to the value of time().

Return Values

Returns a formatted date string. If a non-numeric value is used for *`timestamp`*, **FALSE** is returned and an *E_WARNING* level error is emitted.

ChangeLog

Version	Description
5.1.0	The valid range of a timestamp is typically from Fri, 13 Dec 1901 20:45:54 GMT to Tue, 19 Jan 2038 03:14:07 GMT. (These are the dates that correspond to the minimum and maximum values for a 32-bit signed integer). However, before PHP 5.1.0 this range was limited from 01-01-1970 to 19-01-2038 on some systems (e.g. Windows).
5.1.1	There are useful constants of standard date/time formats that can be used to specify the *`format`* parameter.

Examples

Example #13 - gmdate() example

When run in Finland (GMT +0200), the first line below prints "Jan 01 1998 00:00:00", while the second prints "Dec 31 1997 22:00:00".

```
<?php
echo date("M d Y H:i:s", mktime(0, 0, 0, 1, 1, 1998));
echo gmdate("M d Y H:i:s", mktime(0, 0, 0, 1, 1, 1998));
?>
```

See Also

- date()
- mktime()
- gmmktime()
- strftime()

gmmktime

gmmktime -- Get Unix timestamp for a GMT date

Description

int **gmmktime** ([int `$hour` [, int `$minute` [, int `$second` [, int `$month` [, int `$day` [, int `$year` [, int `$is_dst`]]]]]]])

Identical to mktime() except the passed parameters represents a GMT date. gmmktime() internally uses mktime() so only times valid in derived local time can be used.

Like mktime(), arguments may be left out in order from right to left, with any omitted arguments being set to the current corresponding GMT value.

Parameters

`hour`
The hour

`minute`
The minute

`second`
The second

`month`
The month

`day`
The day

`year`
The year

`is_dst`
Parameters always represent a GMT date so *`is_dst`* doesn't influence the result.

Return Values

Returns a integer Unix timestamp.

ChangeLog

Version	Description
5.1.0	As of PHP 5.1.0, the *is_dst* parameter became deprecated. As a result, the new timezone handling features should be used instead.

Examples

Example #14 - gmmktime() on Windows boundary

```
<?php
gmmktime(0, 0, 0, 1, 1, 1970); // valid in GMT and west, invalid in east
?>
```

See Also

- mktime()
- date()
- time()

gmstrftime

gmstrftime -- Format a GMT/UTC time/date according to locale settings

Description

string **gmstrftime** (string `$format` [, int `$timestamp`])

Behaves the same as strftime() except that the time returned is Greenwich Mean Time (GMT). For example, when run in Eastern Standard Time (GMT -0500), the first line below prints "Dec 31 1998 20:00:00", while the second prints "Jan 01 1999 01:00:00".

Parameters

`format`
: See description in strftime().

`timestamp`
: The optional *`timestamp`* parameter is an integer Unix timestamp that defaults to the current local time if a *`timestamp`* is not given. In other words, it defaults to the value of time().

Return Values

Returns a string formatted according to the given format string using the given *`timestamp`* or the current local time if no timestamp is given. Month and weekday names and other language dependent strings respect the current locale set with setlocale().

Examples

Example #15 - gmstrftime() example

```
<?php
setlocale(LC_TIME, 'en_US');
echo strftime("%b %d %Y %H:%M:%S", mktime(20, 0, 0, 12, 31, 98)) . "\n";
echo gmstrftime("%b %d %Y %H:%M:%S", mktime(20, 0, 0, 12, 31, 98)) . "\n";
?>
```

See Also

- strftime()

idate

idate -- Format a local time/date as integer

Description

int **idate** (string `$format` [, int `$timestamp`])

Returns a number formatted according to the given format string using the given integer *`timestamp`* or the current local time if no timestamp is given. In other words, *`timestamp`* is optional and defaults to the value of time().

Unlike the function date(), idate() accepts just one char in the *`format`* parameter.

Parameters

`format`

The following characters are recognized in the *`format`* parameter string

`format` **character**	**Description**
B	Swatch Beat/Internet Time
d	Day of the month
h	Hour (12 hour format)
H	Hour (24 hour format)
i	Minutes
I (uppercase i)	returns *1* if DST is activated, *0* otherwise
L (uppercase l)	returns *1* for leap year, *0* otherwise
m	Month number
s	Seconds
t	Days in current month
U	Seconds since the Unix Epoch - January 1 1970 00:00:00 UTC - this is the same as time()
w	Day of the week (*0* on Sunday)

W	ISO-8601 week number of year, weeks starting on Monday
y	Year (1 or 2 digits - check note below)
Y	Year (4 digits)
z	Day of the year
Z	Timezone offset in seconds

`timestamp`
The optional `timestamp` parameter is an integer Unix timestamp that defaults to the current local time if a `timestamp` is not given. In other words, it defaults to the value of time().

Return Values

Returns an integer.

As idate() always returns an integer and as they can't start with a "0", idate() may return fewer digits than you would expect. See the example below.

Errors/Exceptions

Every call to a date/time function will generate a **E_NOTICE** if the time zone is not valid, and/or a **E_STRICT** message if using the system settings or the *TZ* environment variable. See also date_default_timezone_set()

ChangeLog

Version	Description
5.1.0	Now issues the **E_STRICT** and **E_NOTICE** time zone errors.

Examples

Example #16 - idate() example

```
<?php
```

```
$timestamp = strtotime('1st January 2004'); //1072915200

// this prints the year in a two digit format
// however, as this would start with a "0", it
// only prints "4"
echo idate('y', $timestamp);
?>
```

See Also

- date()
- time()

localtime

localtime -- Get the local time

Description

array **localtime** ([int `$timestamp` [, bool `$is_associative`]])

The localtime() function returns an array identical to that of the structure returned by the C function call.

Parameters

`timestamp`
: The optional *`timestamp`* parameter is an integer Unix timestamp that defaults to the current local time if a *`timestamp`* is not given. In other words, it defaults to the value of time().

`is_associative`
: If set to **FALSE** or not supplied than the array is returned as a regular, numerically indexed array. If the argument is set to **TRUE** then localtime() is an associative array containing all the different elements of the structure returned by the C function call to localtime. The names of the different keys of the associative array are as follows:

- "tm_sec" - seconds
- "tm_min" - minutes
- "tm_hour" - hour
- "tm_mday" - day of the month Months are from 0 (Jan) to 11 (Dec) and days of the week are from 0 (Sun) to 6 (Sat).
- "tm_mon" - month of the year, starting with 0 for January
- "tm_year" - Years since 1900
- "tm_wday" - Day of the week
- "tm_yday" - Day of the year
- "tm_isdst" - Is daylight savings time in effect

Errors/Exceptions

Every call to a date/time function will generate a **E_NOTICE** if the time zone is not valid, and/or a **E_STRICT** message if using the system settings or the *TZ* environment variable. See also date_default_timezone_set()

ChangeLog

Version	Description
5.1.0	Now issues the **E_STRICT** and **E_NOTICE** time zone errors.

Examples

Example #17 - localtime() example

```
<?php
$localtime = localtime();
$localtime_assoc = localtime(time(), true);
print_r($localtime);
print_r($localtime_assoc);
?>
```

The above example will output something similar to:

```
Array
(
    [0] => 24
    [1] => 3
    [2] => 19
    [3] => 3
    [4] => 3
    [5] => 105
    [6] => 0
    [7] => 92
    [8] => 1
)

Array
(
    [tm_sec] => 24
    [tm_min] => 3
    [tm_hour] => 19
    [tm_mday] => 3
    [tm_mon] => 3
    [tm_year] => 105
    [tm_wday] => 0
    [tm_yday] => 92
    [tm_isdst] => 1
)
```

microtime

microtime -- Return current Unix timestamp with microseconds

Description

mixed **microtime** ([bool `$get_as_float`])

microtime() returns the current Unix timestamp with microseconds. This function is only available on operating systems that support the gettimeofday() system call.

Parameters

`get_as_float`
When called without the optional argument, this function returns the string "msec sec" where sec is the current time measured in the number of seconds since the Unix Epoch (0:00:00 January 1, 1970 GMT), and msec is the microseconds part. Both portions of the string are returned in units of seconds. If the optional *`get_as_float`* is set to **TRUE** then a float (in seconds) is returned.

ChangeLog

Version	Description
5.0.0	The *`get_as_float`* parameter was added.

Examples

Example #18 - Timing script execution with microtime()

```
<?php
/**
* Simple function to replicate PHP 5 behaviour
*/
function microtime_float()
{
   list($usec, $sec) = explode(" ", microtime());
   return ((float)$usec + (float)$sec);
}

$time_start = microtime_float();
```

```
// Sleep for a while
usleep(100);

$time_end = microtime_float();
$time = $time_end - $time_start;

echo "Did nothing in $time seconds\n";
?>
```

Example #19 - Timing script execution in PHP 5

```
<?php
$time_start = microtime(true);

// Sleep for a while
usleep(100);

$time_end = microtime(true);
$time = $time_end - $time_start;

echo "Did nothing in $time seconds\n";
?>
```

See Also

- time()

mktime

mktime -- Get Unix timestamp for a date

Description

int **mktime** ([int `$hour` [, int `$minute` [, int `$second` [, int `$month` [, int `$day` [, int `$year` [, int `$is_dst`]]]]]]])

Returns the Unix timestamp corresponding to the arguments given. This timestamp is a long integer containing the number of seconds between the Unix Epoch (January 1 1970 00:00:00 GMT) and the time specified.

Arguments may be left out in order from right to left; any arguments thus omitted will be set to the current value according to the local date and time.

Parameters

hour
: The number of the hour.

minute
: The number of the minute.

second
: The number of seconds past the minute.

month
: The number of the month.

day
: The number of the day.

year
: The number of the year, may be a two or four digit value, with values between 0-69 mapping to 2000-2069 and 70-100 to 1970-2000. On systems where time_t is a 32bit signed integer, as most common today, the valid range for *year* is somewhere between 1901 and 2038. However, before PHP 5.1.0 this range was limited from 1970 to 2038 on some systems (e.g. Windows).

is_dst
: This parameter can be set to 1 if the time is during daylight savings time (DST), 0 if it is not, or -1 (the default) if it is unknown whether the time is within daylight savings time or not. If it's unknown, PHP tries to figure it out itself. This can cause unexpected (but not incorrect) results. Some times are invalid if DST is enabled on the system PHP is running on or *is_dst* is set to 1. If DST is enabled in e.g. 2:00, all times between 2:00 and 3:00 are invalid and mktime() returns an undefined (usually negative) value. Some systems (e.g. Solaris 8) enable DST at midnight so time 0:30 of the day when DST is

enabled is evaluated as 23:30 of the previous day.

Note
As of PHP 5.1.0, this parameter became deprecated. As a result, the new timezone handling features should be used instead.

Return Values

mktime() returns the Unix timestamp of the arguments given. If the arguments are invalid, the function returns **FALSE** (before PHP 5.1 it returned *-1*).

Errors/Exceptions

Every call to a date/time function will generate a **E_NOTICE** if the time zone is not valid, and/or a **E_STRICT** message if using the system settings or the *TZ* environment variable. See also date_default_timezone_set()

ChangeLog

Version	Description
3.0.10	Added `is_dst` parameter
5.1.0	The `is_dst` parameter became deprecated. Made the function return **FALSE** on error, instead of *-1*. Fixed the function to accept the year, month and day to be all passed as zero.
5.1.0	Now issues the **E_STRICT** and **E_NOTICE** time zone errors.

Examples

Example #20 - mktime() example
mktime() is useful for doing date arithmetic and validation, as it will automatically calculate the correct value for out-of-range input. For example, each of the following lines produces the string "Jan-01-1998".

```
<?php
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));
?>
```

Example #21 - Last day of next month

The last day of any given month can be expressed as the "0" day of the next month, not the -1 day. Both of the following examples will produce the string "The last day in Feb 2000 is: 29".

```
<?php
$lastday = mktime(0, 0, 0, 3, 0, 2000);
echo strftime("Last day in Feb 2000 is: %d", $lastday);
$lastday = mktime(0, 0, 0, 4, -31, 2000);
echo strftime("Last day in Feb 2000 is: %d", $lastday);
?>
```

Notes

Caution

Before PHP 5.1.0, negative timestamps were not supported under any known version of Windows and some other systems as well. Therefore the range of valid years was limited to 1970 through 2038.

See Also

- gmmktime()
- date()
- time()

strftime

strftime -- Format a local time/date according to locale settings

Description

string **strftime** (string `$format` [, int `$timestamp`])

Format a local time/date according to locale settings. Month and weekday names and other language dependent strings respect the current locale set with setlocale().

Not all conversion specifiers may be supported by your C library, in which case they will not be supported by PHP's strftime(). Additionally, not all platforms support negative timestamps, therefore your date range may be limited to no earlier than the Unix epoch. This means that e.g. %e, %T, %R and %D (there might be more) and dates prior to *Jan 1, 1970* will not work on Windows, some Linux distributions, and a few other operating systems. For Windows systems a complete overview of supported conversion specifiers can be found at this MSDN » website.

Parameters

`format`

The following conversion specifiers are recognized in the format string:

- %a - abbreviated weekday name according to the current locale
- %A - full weekday name according to the current locale
- %b - abbreviated month name according to the current locale
- %B - full month name according to the current locale
- %c - preferred date and time representation for the current locale
- %C - century number (the year divided by 100 and truncated to an integer, range 00 to 99)
- %d - day of the month as a decimal number (range 01 to 31)
- %D - same as %m/%d/%y
- %e - day of the month as a decimal number, a single digit is preceded by a space (range ' 1' to '31')
- %g - like %G, but without the century.
- %G - The 4-digit year corresponding to the ISO week number (see %V). This has the same format and value as %Y, except that if the ISO week number belongs to the previous or next year, that year is used instead.
- %h - same as %b
- %H - hour as a decimal number using a 24-hour clock (range 00 to 23)

- %I - hour as a decimal number using a 12-hour clock (range 01 to 12)
- %j - day of the year as a decimal number (range 001 to 366)
- %m - month as a decimal number (range 01 to 12)
- %M - minute as a decimal number
- %n - newline character
- %p - either `am' or `pm' according to the given time value, or the corresponding strings for the current locale
- %r - time in a.m. and p.m. notation
- %R - time in 24 hour notation
- %S - second as a decimal number
- %t - tab character
- %T - current time, equal to %H:%M:%S
- %u - weekday as a decimal number [1,7], with 1 representing Monday

Warning
Sun Solaris seems to start with Sunday as 1 although ISO 9889:1999 (the current C standard) clearly specifies that it should be Monday.

- %U - week number of the current year as a decimal number, starting with the first Sunday as the first day of the first week
- %V - The ISO 8601:1988 week number of the current year as a decimal number, range 01 to 53, where week 1 is the first week that has at least 4 days in the current year, and with Monday as the first day of the week. (Use %G or %g for the year component that corresponds to the week number for the specified timestamp.)
- %W - week number of the current year as a decimal number, starting with the first Monday as the first day of the first week
- %w - day of the week as a decimal, Sunday being 0
- %x - preferred date representation for the current locale without the time
- %X - preferred time representation for the current locale without the date
- %y - year as a decimal number without a century (range 00 to 99)
- %Y - year as a decimal number including the century
- %Z or %z - time zone or name or abbreviation
- %% - a literal `%' character

Maximum length of this parameter is 1023 characters.

`timestamp`

The optional *`timestamp`* parameter is an integer Unix timestamp that defaults to the current local time if a *`timestamp`* is not given. In other words, it defaults to the value of time().

Return Values

Returns a string formatted according to the given format string using the given *timestamp* or the current local time if no timestamp is given. Month and weekday names and other language dependent strings respect the current locale set with setlocale().

Errors/Exceptions

Every call to a date/time function will generate a **E_NOTICE** if the time zone is not valid, and/or a **E_STRICT** message if using the system settings or the *TZ* environment variable. See also date_default_timezone_set()

ChangeLog

Version	Description
5.1.0	Now issues the **E_STRICT** and **E_NOTICE** time zone errors.

Examples

This example works if you have the respective locales installed in your system.

Example #22 - strftime() locale examples

```
<?php
setlocale(LC_TIME, "C");
echo strftime("%A");
setlocale(LC_TIME, "fi_FI");
echo strftime(" in Finnish is %A,");
setlocale(LC_TIME, "fr_FR");
echo strftime(" in French %A and");
setlocale(LC_TIME, "de_DE");
echo strftime(" in German %A.\n");
?>
```

Example #23 - ISO 8601:1988 week number example

```
<?php
/*     December 2002 / January 2003
ISOWk  M   Tu  W   Thu F   Sa  Su
-----  ----------------------------
51     16  17  18  19  20  21  22
52     23  24  25  26  27  28  29
1      30  31   1   2   3   4   5
```

```
2       6   7   8   9  10  11  12
3      13  14  15  16  17  18  19   */

// Outputs: 12/28/2002 - %V,%G,%Y = 52,2002,2002
echo "12/28/2002 - %V,%G,%Y = " . strftime("%V,%G,%Y",
strtotime("12/28/2002")) . "\n";

// Outputs: 12/30/2002 - %V,%G,%Y = 1,2003,2002
echo "12/30/2002 - %V,%G,%Y = " . strftime("%V,%G,%Y",
strtotime("12/30/2002")) . "\n";

// Outputs: 1/3/2003 - %V,%G,%Y = 1,2003,2003
echo "1/3/2003 - %V,%G,%Y = " . strftime("%V,%G,%Y",strtotime("1/3/2003")) .
"\n";

// Outputs: 1/10/2003 - %V,%G,%Y = 2,2003,2003
echo "1/10/2003 - %V,%G,%Y = " . strftime("%V,%G,%Y",strtotime("1/10/2003"))
. "\n";



/*     December 2004 / January 2005
ISOWk  M   Tu  W   Thu F   Sa  Su
----- ----------------------------
51     13  14  15  16  17  18  19
52     20  21  22  23  24  25  26
53     27  28  29  30  31   1   2
1       3   4   5   6   7   8   9
2      10  11  12  13  14  15  16   */

// Outputs: 12/23/2004 - %V,%G,%Y = 52,2004,2004
echo "12/23/2004 - %V,%G,%Y = " .
strftime("%V,%G,%Y",strtotime("12/23/2004")) . "\n";

// Outputs: 12/31/2004 - %V,%G,%Y = 53,2004,2004
echo "12/31/2004 - %V,%G,%Y = " .
strftime("%V,%G,%Y",strtotime("12/31/2004")) . "\n";

// Outputs: 1/2/2005 - %V,%G,%Y = 53,2004,2005
echo "1/2/2005 - %V,%G,%Y = " . strftime("%V,%G,%Y",strtotime("1/2/2005")) .
"\n";

// Outputs: 1/3/2005 - %V,%G,%Y = 1,2005,2005
echo "1/3/2005 - %V,%G,%Y = " . strftime("%V,%G,%Y",strtotime("1/3/2005")) .
"\n";

?>
```

Notes

Note
%G and %V, which are based on ISO 8601:1988 week numbers can give unexpected (albeit correct) results if the numbering system is not thoroughly understood. See %V examples in this manual page.

See Also

- setlocale()
- mktime()
- strptime()
- gmstrftime()
- » Open Group specification of strftime()

strptime

strptime -- Parse a time/date generated with strftime()

Description

array **strptime** (string `$date`, string `$format`)

strptime() returns an array with the *`date`* parsed, or **FALSE** on error.

Month and weekday names and other language dependent strings respect the current locale set with setlocale() (LC_TIME).

Parameters

`date` (string)
The string to parse (e.g. returned from strftime())

`format` (string)
The format used in *`date`* (e.g. the same as used in strftime()). For more information about the format options, read the strftime() page.

Return Values

Returns an array, or **FALSE** on failure.

The following parameters are returned in the array

parameters	**Description**
tm_sec	Seconds after the minute (0-61)
tm_min	Minutes after the hour (0-59)
tm_hour	Hour since midnight (0-23)
tm_mday	Day of the month (1-31)
tm_mon	Months since January (0-11)
tm_year	Years since 1900
tm_wday	Days since Sunday (0-6)
tm_yday	Days since January 1 (0-365)

unparsed	the *date* part which was not recognized using the specified *format*

Examples

Example #24 - strptime() example

```
<?php
$format = '%d/%m/%Y %H:%M:%S';
$strf = strftime($format);

echo "$strf\n";

print_r(strptime($strf, $format));
?>
```

The above example will output something similar to:

```
03/10/2004 15:54:19

Array
(
   [tm_sec] => 19
   [tm_min] => 54
   [tm_hour] => 15
   [tm_mday] => 3
   [tm_mon] => 9
   [tm_year] => 104
   [tm_wday] => 0
   [tm_yday] => 276
   [unparsed] =>
)
```

Notes

Note

This function is not implemented on Windows platforms.

See Also

- strftime()

strtotime

strtotime -- Parse about any English textual datetime description into a Unix timestamp

Description

int **strtotime** (string `$time` [, int `$now`])

The function expects to be given a string containing a US English date format and will try to parse that format into a Unix timestamp (the number of seconds since January 1 1970 00:00:00 GMT), relative to the timestamp given in *now*, or the current time if *now* is not supplied.

This function will use the TZ environment variable (if available) to calculate the timestamp. Since PHP 5.1.0 there are easier ways to define the timezone that is used across all date/time functions. That process is explained in the date_default_timezone_get() function page.

Note
If the number of the year is specified in a two digit format, the values between 00-69 are mapped to 2000-2069 and 70-99 to 1970-1999.

Parameters

time
: The string to parse, according to the GNU » Date Input Formats syntax. Before PHP 5.0.0, microseconds weren't allowed in the time, since PHP 5.0.0 they are allowed but ignored.

now
: The timestamp used to calculate the returned value.

Return Values

Returns a timestamp on success, **FALSE** otherwise. Previous to PHP 5.1.0, this function would return *-1* on failure.

Errors/Exceptions

Every call to a date/time function will generate a **E_NOTICE** if the time zone is not valid, and/or a **E_STRICT** message if using the system settings or the *TZ* environment variable. See also date_default_timezone_set()

ChangeLog

Version	Description
5.1.0	It now returns **FALSE** on failure, instead of *-1*.
5.1.0	Now issues the **E_STRICT** and **E_NOTICE** time zone errors.

Examples

Example #25 - A strtotime() example

```
<?php
echo strtotime("now"), "\n";
echo strtotime("10 September 2000"), "\n";
echo strtotime("+1 day"), "\n";
echo strtotime("+1 week"), "\n";
echo strtotime("+1 week 2 days 4 hours 2 seconds"), "\n";
echo strtotime("next Thursday"), "\n";
echo strtotime("last Monday"), "\n";
?>
```

Example #26 - Checking for failure

```
<?php
$str = 'Not Good';

// previous to PHP 5.1.0 you would compare with -1, instead of false
if (($timestamp = strtotime($str)) === false) {
   echo "The string ($str) is bogus";
} else {
   echo "$str == " . date('l dS \o\f F Y h:i:s A', $timestamp);
}
?>
```

Notes

Warning

In PHP 5 up to 5.0.2, *"now"* and other relative times are wrongly computed from today's midnight. It differs from other versions where it is correctly computed from current time.

Warning

In PHP versions prior to 4.4.0, *"next"* is incorrectly computed as +2. A typical solution to this is to use *"+1"*.

Note

The valid range of a timestamp is typically from Fri, 13 Dec 1901 20:45:54 GMT to Tue, 19 Jan 2038 03:14:07 GMT. (These are the dates that correspond to the minimum and maximum values for a 32-bit signed integer.) Additionally, not all platforms support negative timestamps, therefore your date range may be limited to no earlier than the Unix epoch. This means that e.g. dates prior to Jan 1, 1970 will not work on Windows, some Linux distributions, and a few other operating systems. PHP 5.1.0 and newer versions overcome this limitation though.

See Also

- strptime()

time

time -- Return current Unix timestamp

Description

int **time** (void)

Returns the current time measured in the number of seconds since the Unix Epoch (January 1 1970 00:00:00 GMT).

Examples

Example #27 - time() example

```
<?php
$nextWeek = time() + (7 * 24 * 60 * 60);
                  // 7 days; 24 hours; 60 mins; 60secs
echo 'Now:       '. date('Y-m-d') ."\n";
echo 'Next Week: '. date('Y-m-d', $nextWeek) ."\n";
// or using strtotime():
echo 'Next Week: '. date('Y-m-d', strtotime('+1 week')) ."\n";
?>
```

The above example will output something similar to:

```
Now:       2005-03-30
Next Week: 2005-04-06
Next Week: 2005-04-06
```

Notes

Tip

Timestamp of the start of the request is available in *$_SERVER['REQUEST_TIME']* since PHP 5.1.

See Also

- date()
- microtime()

timezone_abbreviations_list

timezone_abbreviations_list -- Returns associative array containing dst, offset and the timezone name

Description

array **timezone_abbreviations_list** (void)

array **DateTimeZone::listAbbreviations** (void)

Return Values

Returns array on success or **FALSE** on failure.

Examples

Example #28 - A timezone_abbreviations_list() example

```
<?php
$timezone_abbreviations = DateTimeZone::listAbbreviations();
print_r($timezone_abbreviations["acst"]);
?>
```

The above example will output something similar to:

```
Array
(
   [0] => Array
       (
           [dst] => 1
           [offset] => -14400
           [timezone_id] => America/Porto_Acre
       )

   [1] => Array
       (
           [dst] => 1
           [offset] => -14400
           [timezone_id] => America/Eirunepe
       )

   [2] => Array
       (
           [dst] => 1
           [offset] => -14400
           [timezone_id] => America/Rio_Branco
       )

   [3] => Array
       (
```

```
            [dst] => 1
            [offset] => -14400
            [timezone_id] => Brazil/Acre
        )

)
```

See Also

- timezone_identifiers_list()

timezone_identifiers_list

timezone_identifiers_list -- Returns numerically index array with all timezone identifiers

Description

array **timezone_identifiers_list** (void)

array **DateTimeZone::listIdentifiers** (void)

Return Values

Returns array on success or **FALSE** on failure.

Examples

Example #29 - A timezone_identifiers_list() **example**

```
<?php
$timezone_identifiers = DateTimeZone::listIdentifiers();
for ($i=0; $i < 5; $i++) {
   echo "$timezone_identifiers[$i]\n";
}
?>
```

The above example will output something similar to:

```
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmera
```

See Also

- timezone_abbreviations_list()

timezone_name_from_abbr

timezone_name_from_abbr -- Returns the timezone name from abbrevation

Description

string **timezone_name_from_abbr** (string `$abbr` [, int `$gmtOffset` [, int `$isdst`]])

Parameters

`abbr`
: Time zone abbreviation.

`gmtOffset`
: Offset from GMT in seconds. Defaults to -1 which means that first found time zone corresponding to *`abbr`* is returned. Otherwise exact offset is searched and only if not found then the first time zone with any offset is returned.

`isdst`
: Daylight saving time indicator. If *`abbr`* doesn't exist then the time zone is searched solely by *`offset`* and *`isdst`*.

Return Values

Returns time zone name on success or **FALSE** on failure.

Examples

Example #30 - A timezone_name_from_abbr() example

```
<?php
echo timezone_name_from_abbr("CET") . "\n";
echo timezone_name_from_abbr("", 3600, 0) . "\n";
?>
```

The above example will output something similar to:

```
Europe/Berlin
Europe/Paris
```

See Also

- timezone_abbreviations_list()

timezone_name_get

timezone_name_get -- Returns the name of the timezone

Description

string **timezone_name_get** (DateTimeZone `$object`)

string **DateTimeZone::getName** (void)

Parameters

`object`
DateTimeZone object.

Return Values

Returns time zone name on success or **FALSE** on failure.

timezone_offset_get

timezone_offset_get -- Returns the timezone offset from GMT

Description

int **timezone_offset_get** (DateTimeZone `$object`, DateTime `$datetime`)

int **DateTimeZone::getOffset** (DateTime `$datetime`)

This function returns the offset to GMT for the date/time specified in the *`datetime`* parameter. The GMT offset is calculated with the timezone information contained in the DateTime object being used.

Parameters

`object`
DateTimeZone object.

`datetime`
DateTime that contains the date/time to compute the offset from.

Return Values

Returns time zone offset in seconds on success or **FALSE** on failure.

Examples

Example #31 - timezone_offset_get() examples

```
<?php
// Create two timezone objects, one for Taipei (Taiwan) and one for
// Tokyo (Japan)
$dateTimeZoneTaipei = new DateTimeZone("Asia/Taipei");
$dateTimeZoneJapan = new DateTimeZone("Asia/Tokyo");

// Create two DateTime objects that will contain the same Unix timestamp,
but
// have different timezones attached to them.
$dateTimeTaipei = new DateTime("now", $dateTimeZoneTaipei);
$dateTimeJapan = new DateTime("now", $dateTimeZoneJapan);

// Calculate the GMT offset for the date/time contained in the
$dateTimeTaipei
// object, but using the timezone rules as defined for Tokyo
// ($dateTimeZoneJapan).
$timeOffset = $dateTimeZoneJapan->getOffset($dateTimeTaipei);
```

```
// Should show int(32400) (for dates after Sat Sep 8 01:00:00 1951 JST).
var_dump($timeOffset);
?>
```

timezone_open

timezone_open -- Returns new DateTimeZone object

Description

DateTimeZone **timezone_open** (string `$timezone`)

DateTimeZone **DateTimeZone::__construct** (string `$timezone`)

Parameters

`timezone`
: Time zone identifier as full name (e.g. Europe/Prague) or abbreviation (e.g. CET).

Return Values

Returns DateTimeZone object on success or **FALSE** on failure.

timezone_transitions_get

timezone_transitions_get -- Returns all transitions for the timezone

Description

array **timezone_transitions_get** (DateTimeZone `$object`)

array **DateTimeZone::getTransitions** (void)

Parameters

`object`
DateTimeZone object.

Return Values

Returns numerically indexed array containing associative array with all transitions on success or **FALSE** on failure.

Examples

Example #32 - A timezone_transitions_get() example

```
<?php
$timezone = new DateTimeZone("CET");
print_r(reset($timezone->getTransitions()));
?>
```

The above example will output something similar to:

```
Array
(
   [ts] => -1693706400
   [time] => 1916-04-30T22:00:00+0000
   [offset] => 7200
   [isdst] => 1
   [abbr] => CEST
)
```