**Date Helper**

The Date Helper file contains functions that help you work with dates.

* [Loading this Helper](https://codeigniter.com/user_guide/helpers/date_helper.html#loading-this-helper)
* [Available Functions](https://codeigniter.com/user_guide/helpers/date_helper.html#available-functions)
* [Timezone Reference](https://codeigniter.com/user_guide/helpers/date_helper.html#timezone-reference)

[**Loading this Helper**](https://codeigniter.com/user_guide/helpers/date_helper.html#id1)

This helper is loaded using the following code:

$this**->**load**->**helper('date');

[**Available Functions**](https://codeigniter.com/user_guide/helpers/date_helper.html#id2)

The following functions are available:

**now([*$timezone = NULL*])**

|  |  |
| --- | --- |
| **Parameters:** | * **$timezone** (*string*) – Timezone |
| **Returns:** | UNIX timestamp |
| **Return type:** | int |

Returns the current time as a UNIX timestamp, referenced either to your server’s local time or any PHP supported timezone, based on the “time reference” setting in your config file. If you do not intend to set your master time reference to any other PHP supported timezone (which you’ll typically do if you run a site that lets each user set their own timezone settings) there is no benefit to using this function over PHP’s time() function.

**echo** now('Australia/Victoria');

If a timezone is not provided, it will return time() based on the **time\_reference** setting.

**mdate([*$datestr = ''*[, *$time = ''*]])**

|  |  |
| --- | --- |
| **Parameters:** | * **$datestr** (*string*) – Date string * **$time** (*int*) – UNIX timestamp |
| **Returns:** | MySQL-formatted date |
| **Return type:** | string |

This function is identical to PHP’s [date()](https://php.net/manual/en/function.date.php) function, except that it lets you use MySQL style date codes, where each code letter is preceded with a percent sign, e.g. *%Y %m %d*

The benefit of doing dates this way is that you don’t have to worry about escaping any characters that are not date codes, as you would normally have to do with the date() function.

Example:

$datestring **=** 'Year: %Y Month: %m Day: %d - %h:%i %a';

$time **=** time();

**echo** mdate($datestring, $time);

If a timestamp is not included in the second parameter the current time will be used.

**standard\_date([*$fmt = 'DATE\_RFC822'*[, *$time = NULL*]])**

|  |  |
| --- | --- |
| **Parameters:** | * **$fmt** (*string*) – Date format * **$time** (*int*) – UNIX timestamp |
| **Returns:** | Formatted date or FALSE on invalid format |
| **Return type:** | string |

Lets you generate a date string in one of several standardized formats.

Example:

$format **=** 'DATE\_RFC822';

$time **=** time();

**echo** standard\_date($format, $time);

**Note**

This function is DEPRECATED. Use the native date() combined with [DateTime’s format constants](https://secure.php.net/manual/en/class.datetime.php#datetime.constants.types) instead:

**echo** date(DATE\_RFC822, time());

**Supported formats:**

| **Constant** | **Description** | **Example** |
| --- | --- | --- |
| DATE\_ATOM | Atom | 2005-08-15T16:13:03+0000 |
| DATE\_COOKIE | HTTP Cookies | Sun, 14 Aug 2005 16:13:03 UTC |
| DATE\_ISO8601 | ISO-8601 | 2005-08-14T16:13:03+00:00 |
| DATE\_RFC822 | RFC 822 | Sun, 14 Aug 05 16:13:03 UTC |
| DATE\_RFC850 | RFC 850 | Sunday, 14-Aug-05 16:13:03 UTC |
| DATE\_RFC1036 | RFC 1036 | Sunday, 14-Aug-05 16:13:03 UTC |
| DATE\_RFC1123 | RFC 1123 | Sun, 14 Aug 2005 16:13:03 UTC |
| DATE\_RFC2822 | RFC 2822 | Sun, 14 Aug 2005 16:13:03 +0000 |
| DATE\_RSS | RSS | Sun, 14 Aug 2005 16:13:03 UTC |
| DATE\_W3C | W3C | 2005-08-14T16:13:03+0000 |

**local\_to\_gmt([*$time = ''*])**

|  |  |
| --- | --- |
| **Parameters:** | * **$time** (*int*) – UNIX timestamp |
| **Returns:** | UNIX timestamp |
| **Return type:** | int |

Takes a UNIX timestamp as input and returns it as GMT.

Example:

$gmt **=** local\_to\_gmt(time());

**gmt\_to\_local([*$time = ''*[, *$timezone = 'UTC'*[, *$dst = FALSE*]]])**

|  |  |
| --- | --- |
| **Parameters:** | * **$time** (*int*) – UNIX timestamp * **$timezone** (*string*) – Timezone * **$dst** (*bool*) – Whether DST is active |
| **Returns:** | UNIX timestamp |
| **Return type:** | int |

Takes a UNIX timestamp (referenced to GMT) as input, and converts it to a localized timestamp based on the timezone and Daylight Saving Time submitted.

Example:

$timestamp **=** 1140153693;

$timezone **=** 'UM8';

$daylight\_saving **=** **TRUE**;

**echo** gmt\_to\_local($timestamp, $timezone, $daylight\_saving);

**Note**

For a list of timezones see the reference at the bottom of this page.

**mysql\_to\_unix([*$time = ''*])**

|  |  |
| --- | --- |
| **Parameters:** | * **$time** (*string*) – MySQL timestamp |
| **Returns:** | UNIX timestamp |
| **Return type:** | int |

Takes a MySQL Timestamp as input and returns it as a UNIX timestamp.

Example:

$unix **=** mysql\_to\_unix('20061124092345');

**unix\_to\_human([*$time = ''*[, *$seconds = FALSE*[, *$fmt = 'us'*]]])**

|  |  |
| --- | --- |
| **Parameters:** | * **$time** (*int*) – UNIX timestamp * **$seconds** (*bool*) – Whether to show seconds * **$fmt** (*string*) – format (us or euro) |
| **Returns:** | Formatted date |
| **Return type:** | string |

Takes a UNIX timestamp as input and returns it in a human readable format with this prototype:

YYYY**-**MM**-**DD HH**:**MM**:**SS AM**/**PM

This can be useful if you need to display a date in a form field for submission.

The time can be formatted with or without seconds, and it can be set to European or US format. If only the timestamp is submitted it will return the time without seconds formatted for the U.S.

Examples:

$now **=** time();

**echo** unix\_to\_human($now); *// U.S. time, no seconds*

**echo** unix\_to\_human($now, **TRUE**, 'us'); *// U.S. time with seconds*

**echo** unix\_to\_human($now, **TRUE**, 'eu'); *// Euro time with seconds*

**human\_to\_unix([*$datestr = ''*])**

|  |  |
| --- | --- |
| **Parameters:** | * **$datestr** (*int*) – Date string |
| **Returns:** | UNIX timestamp or FALSE on failure |
| **Return type:** | int |

The opposite of the unix\_to\_time() function. Takes a “human” time as input and returns it as a UNIX timestamp. This is useful if you accept “human” formatted dates submitted via a form. Returns boolean FALSE date string passed to it is not formatted as indicated above.

Example:

$now **=** time();

$human **=** unix\_to\_human($now);

$unix **=** human\_to\_unix($human);

**nice\_date([*$bad\_date = ''*[, *$format = FALSE*]])**

|  |  |
| --- | --- |
| **Parameters:** | * **$bad\_date** (*int*) – The terribly formatted date-like string * **$format** (*string*) – Date format to return (same as PHP’s date() function) |
| **Returns:** | Formatted date |
| **Return type:** | string |

This function can take a number poorly-formed date formats and convert them into something useful. It also accepts well-formed dates.

The function will return a UNIX timestamp by default. You can, optionally, pass a format string (the same type as the PHP date() function accepts) as the second parameter.

Example:

$bad\_date **=** '199605';

*// Should Produce: 1996-05-01*

$better\_date **=** nice\_date($bad\_date, 'Y-m-d');

$bad\_date **=** '9-11-2001';

*// Should Produce: 2001-09-11*

$better\_date **=** nice\_date($bad\_date, 'Y-m-d');

**Note**

This function is DEPRECATED. Use PHP’s native [DateTime class](https://secure.php.net/datetime) instead.

**timespan([*$seconds = 1*[, *$time = ''*[, *$units = ''*]]])**

|  |  |
| --- | --- |
| **Parameters:** | * **$seconds** (*int*) – Number of seconds * **$time** (*string*) – UNIX timestamp * **$units** (*int*) – Number of time units to display |
| **Returns:** | Formatted time difference |
| **Return type:** | string |

Formats a UNIX timestamp so that is appears similar to this:

1 Year, 10 Months, 2 Weeks, 5 Days, 10 Hours, 16 Minutes

The first parameter must contain a UNIX timestamp. The second parameter must contain a timestamp that is greater that the first timestamp. The thirdparameter is optional and limits the number of time units to display.

If the second parameter empty, the current time will be used.

The most common purpose for this function is to show how much time has elapsed from some point in time in the past to now.

Example:

$post\_date **=** '1079621429';

$now **=** time();

$units **=** 2;

**echo** timespan($post\_date, $now, $units);

**Note**

The text generated by this function is found in the following language file: *language/<your\_lang>/date\_lang.php*

**days\_in\_month([*$month = 0*[, *$year = ''*]])**

|  |  |
| --- | --- |
| **Parameters:** | * **$month** (*int*) – a numeric month * **$year** (*int*) – a numeric year |
| **Returns:** | Count of days in the specified month |
| **Return type:** | int |

Returns the number of days in a given month/year. Takes leap years into account.

Example:

**echo** days\_in\_month(06, 2005);

If the second parameter is empty, the current year will be used.

**Note**

This function will alias the native cal\_days\_in\_month(), if it is available.

**date\_range([*$unix\_start = ''*[, *$mixed = ''*[, *$is\_unix = TRUE*[, *$format = 'Y-m-d'*]]]])**

|  |  |
| --- | --- |
| **Parameters:** | * **$unix\_start** (*int*) – UNIX timestamp of the range start date * **$mixed** (*int*) – UNIX timestamp of the range end date or interval in days * **$is\_unix** (*bool*) – set to FALSE if $mixed is not a timestamp * **$format** (*string*) – Output date format, same as in date() |
| **Returns:** | An array of dates |
| **Return type:** | array |

Returns a list of dates within a specified period.

Example:

$range **=** date\_range('2012-01-01', '2012-01-15');

**echo** "First 15 days of 2012:";

**foreach** ($range **as** $date)

{

**echo** $date**.**"\n";

}

**timezones([*$tz = ''*])**

|  |  |
| --- | --- |
| **Parameters:** | * **$tz** (*string*) – A numeric timezone |
| **Returns:** | Hour difference from UTC |
| **Return type:** | int |

Takes a timezone reference (for a list of valid timezones, see the “Timezone Reference” below) and returns the number of hours offset from UTC.

Example:

**echo** timezones('UM5');

This function is useful when used with [timezone\_menu()](https://codeigniter.com/user_guide/helpers/date_helper.html#timezone_menu).

**timezone\_menu([*$default = 'UTC'*[, *$class = ''*[, *$name = 'timezones'*[, *$attributes = ''*]]]])**

|  |  |
| --- | --- |
| **Parameters:** | * **$default** (*string*) – Timezone * **$class** (*string*) – Class name * **$name** (*string*) – Menu name * **$attributes** (*mixed*) – HTML attributes |
| **Returns:** | HTML drop down menu with time zones |
| **Return type:** | string |

Generates a pull-down menu of timezones, like this one:

Principio del formulario



Final del formulario

This menu is useful if you run a membership site in which your users are allowed to set their local timezone value.

The first parameter lets you set the “selected” state of the menu. For example, to set Pacific time as the default you will do this:

**echo** timezone\_menu('UM8');

Please see the timezone reference below to see the values of this menu.

The second parameter lets you set a CSS class name for the menu.

The fourth parameter lets you set one or more attributes on the generated select tag.

**Note**

The text contained in the menu is found in the following language file: *language/<your\_lang>/date\_lang.php*

[**Timezone Reference**](https://codeigniter.com/user_guide/helpers/date_helper.html#id3)

The following table indicates each timezone and its location.

Note some of the location lists have been abridged for clarity and formatting.

| **Time Zone** | **Location** |
| --- | --- |
| UM12 | (UTC - 12:00) Baker/Howland Island |
| UM11 | (UTC - 11:00) Samoa Time Zone, Niue |
| UM10 | (UTC - 10:00) Hawaii-Aleutian Standard Time, Cook Islands |
| UM95 | (UTC - 09:30) Marquesas Islands |
| UM9 | (UTC - 09:00) Alaska Standard Time, Gambier Islands |
| UM8 | (UTC - 08:00) Pacific Standard Time, Clipperton Island |
| UM7 | (UTC - 07:00) Mountain Standard Time |
| UM6 | (UTC - 06:00) Central Standard Time |
| UM5 | (UTC - 05:00) Eastern Standard Time, Western Caribbean |
| UM45 | (UTC - 04:30) Venezuelan Standard Time |
| UM4 | (UTC - 04:00) Atlantic Standard Time, Eastern Caribbean |
| UM35 | (UTC - 03:30) Newfoundland Standard Time |
| UM3 | (UTC - 03:00) Argentina, Brazil, French Guiana, Uruguay |
| UM2 | (UTC - 02:00) South Georgia/South Sandwich Islands |
| UM1 | (UTC -1:00) Azores, Cape Verde Islands |
| UTC | (UTC) Greenwich Mean Time, Western European Time |
| UP1 | (UTC +1:00) Central European Time, West Africa Time |
| UP2 | (UTC +2:00) Central Africa Time, Eastern European Time |
| UP3 | (UTC +3:00) Moscow Time, East Africa Time |
| UP35 | (UTC +3:30) Iran Standard Time |
| UP4 | (UTC +4:00) Azerbaijan Standard Time, Samara Time |
| UP45 | (UTC +4:30) Afghanistan |
| UP5 | (UTC +5:00) Pakistan Standard Time, Yekaterinburg Time |
| UP55 | (UTC +5:30) Indian Standard Time, Sri Lanka Time |
| UP575 | (UTC +5:45) Nepal Time |
| UP6 | (UTC +6:00) Bangladesh Standard Time, Bhutan Time, Omsk Time |
| UP65 | (UTC +6:30) Cocos Islands, Myanmar |
| UP7 | (UTC +7:00) Krasnoyarsk Time, Cambodia, Laos, Thailand, Vietnam |
| UP8 | (UTC +8:00) Australian Western Standard Time, Beijing Time |
| UP875 | (UTC +8:45) Australian Central Western Standard Time |
| UP9 | (UTC +9:00) Japan Standard Time, Korea Standard Time, Yakutsk |
| UP95 | (UTC +9:30) Australian Central Standard Time |
| UP10 | (UTC +10:00) Australian Eastern Standard Time, Vladivostok Time |
| UP105 | (UTC +10:30) Lord Howe Island |
| UP11 | (UTC +11:00) Srednekolymsk Time, Solomon Islands, Vanuatu |
| UP115 | (UTC +11:30) Norfolk Island |
| UP12 | (UTC +12:00) Fiji, Gilbert Islands, Kamchatka, New Zealand |
| UP1275 | (UTC +12:45) Chatham Islands Standard Time |
| UP13 | (UTC +13:00) Phoenix Islands Time, Tonga |
| UP14 | (UTC +14:00) Line Islands |