

# Time = Money

CodeWorks 2009 - New York City, US

Oct 5th, 2009

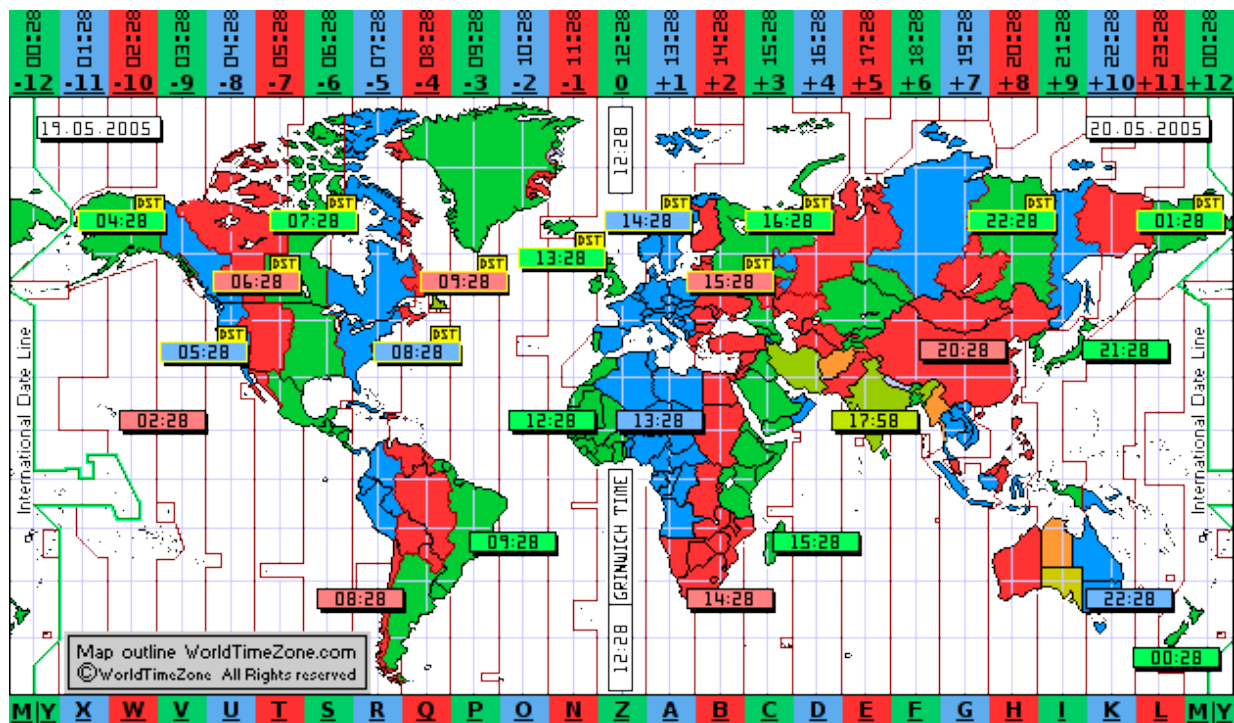
Derick Rethans - dr@ez.no

<http://derickrethans.nl/talks.php> - <http://joind.in/>



- Solar Time, easy to determine, but hard to compare

# Problems Timezones

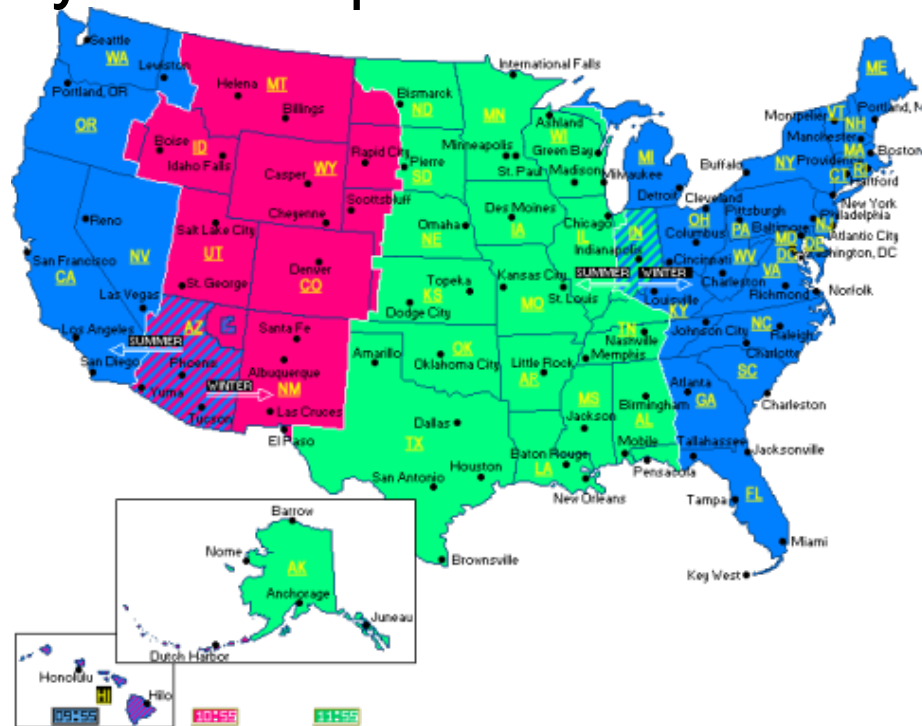


- Most places have whole-hour timezone offsets
- Some places have half-hour or even quarter-hour timezone offsets
- Some places change timezones during the year

# Problems

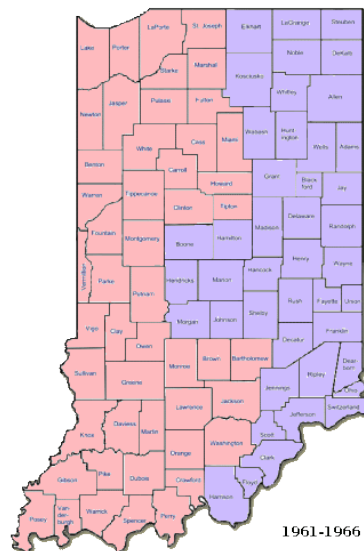
## Daylight Savings Time

- Artificial time offset to save "daylight"
- Not all countries/areas use it
- Switches are not done at the same time for all areas
- There are plenty of exceptions



# Problems

## Daylight Savings Time (Indiana)



(Currently) Three different rules:

- Most counties: Eastern, no DST
- Counties close to Chicago and Evansvile: Central, DST
- Counties close to Cincinnatti and Lousville: Eastern, DST

# Problems

## Other problems with DST and timezones



- Australia has vertical timezones
- Brazil changes change-over dates every year
- Lord Howe Island (Australia) moves only half an hour between DST and non-DST
- Nepal uses an quarter of an hour offsets
- And this continues for a while...

One identifier can mean different zones:

- PST: Pacific Standard Time, Pakistan Standard Time
- EST: Eastern Standard Time (USA), Eastern Standard Time (Australia) and Eastern Brazil Standard Time

One zone can have multiple identifiers

- Central Europe Summer Time: CEST or CETDST

Names can be different on Operating Systems

- Uses Unix timestamp as base unit (seconds since 1970-01-01, 00:00 GMT)
- Only 32 bit integers for timestamps (1902 to 2038)
- Limited to only positive numbers on some Operating Systems (1970 to 2038)
- `strtotime()` is buggy and its implementation is very complex
- No way of dealing correctly with timezones
- Some functions are Operating System dependent
- Avoid functions that deal with timestamps as much as you can!

- "Using a (signed) 64-bit value introduces a new wraparound date in about 290 billion years, on Sunday, December 4, 292,277,026,596. However, this problem is not widely regarded as a pressing issue."
- `strtotime()` has been rewritten
- Nothing is Operating System dependent
- Full support for timezones, DST, date modifications
- New format modifiers: `e` for timezone identifier and `o` for ISO Year
- Advanced date handling functions

### "American" formats:

9/11                                    4:08 pm  
12/22/78                                8:51:00 am

### "Combined" formats:

Sat, 24 Apr 2004 21:48:40 +0200  
2001-11-29T13:20:01.123-05:00

### Descriptive formats:

tomorrow                                last saturday  
four months ago                        +40 days 2 hours

### Textual formats:

December 22. 1978                      22-december-78

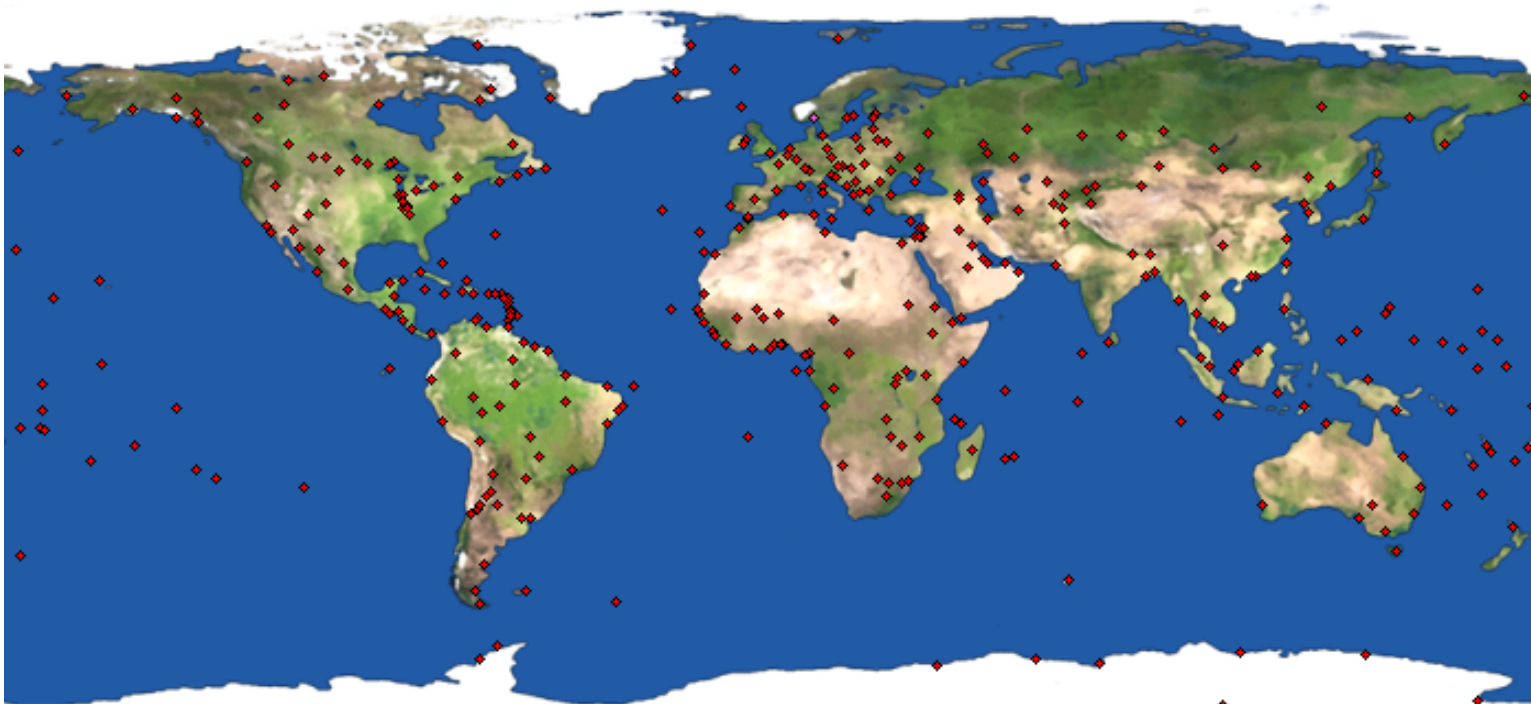
### All ISO 8601 formats:

1978/12/22	1978-12-22	70-4-25
13:03:12.45678	15:57:41.0 pdt	13:03 CEST
13:03:12.45678 +0100	13:03:12.45678 CEST	04:05 -0930
15:57-8	231431 CEST	23:41F

### Database specific formats:

1999-Jan-08  
1999.238

- Bundled timezone database with 560 zones
- Not dependent on timezone abbreviations
- Timezones have the format: Continent/Location or Continent/Location/Sublocation - Like: Europe/Amsterdam, America/Indiana/Knox



- Each zone is identified by the city with the largest population in a specific area.
- Zones are divided into 10 major groups: Africa, America, Antarctica, Artic, Asia, Atlantic, Europe, Indian, Pacific.
- Examples are: America/Toronto, Europe/Berlin, America/New\_York, Europe/Oslo, Asia/Singapore.
- There is also a group "Others" which contains outdated and backwards compatible names, such as Canada/Eastern, Etc/GMT-1 and US/Central. Those should not be used.
- Pick the zone that is closest to your location.
- <http://php.net/timezones>

- An updated database is released about 20 times a year.
- Some of the changes are very sudden.
- PHP releases will therefore often have an outdated version.
- The PECL extension `timezonedb` provides a drop in replacement for the timezone database.
- `pecl install timezonedb`

Parsing strings for date time information with the `strtotime()` function:

```
<?php
    $ts = strtotime("2005-07-11 22:16:50 CEST");
?>
```

With initial timestamp:

```
<?php
    $date = strtotime("2005-07-11 22:16:50 CEST");
    $ts = strtotime("next week", $date);
?>
```

The timestamp's range is still limited to what the CPU supports.

Parsing strings for date time information by instantiating a `DateTime` object:

```
<?php
    $ts = new DateTime("2006-11-07 16:12:15");
?>
```

This function will not return the timestamp as an integer, but instead returns a `DateTime` object which is a wrapper around a 64 bit integer, which you can access (as string) through:

```
<?php
    $ts = new DateTime("2007-09-03 13:44:39");
    echo $ts->format( 'U' ), "\n";
?>
```

## Parsing strings with the date\_parse() function:

```
<pre><?php
$date = "22apr2006 8:36:14.43 #^ Europe/Oslo CEST";
$t = date_parse( $date );

echo $t['year'], '-', $t['month'], '-', $t['day'], " ";
echo $t['hour'], ':', $t['minute'], ':', $t['second'] + $t['fraction'], " ";
echo $t['tz_id'], "\n";

if ( $t['warning_count'] )
    echo "Warnings:\n";
    foreach( $t['warnings'] as $pos => $message )
        echo "- $message @$pos\n";
if ( $t['error_count'] )
    echo "Errors:\n";
    foreach( $t['errors'] as $pos => $message )
        echo "- $message @$pos\n";
```

## Creating strings with the `date_create_format_format()` function:

```
<?php
$date = "06/08/04";
echo DateTime::createFromFormat( '!m/d/y', $date )->format( DateTime::ISO8601 ), "\n";
?>
```

## Parsing strings with the `date_parse_from_format()` function:

```
<?php
$date = "06/08/04";
print_r( date_parse_from_format( '!m*d*y', $date ) );
?>
```

## Formatting using format specifiers:

```
<?php
date_default_timezone_set("Europe/Oslo");
$ts = new DateTime("1979-12-31 09:15");
echo $ts->format("D Y-m-d H:i:s - \I\S\O \W\Y: W/o"), "<br/>\n";
```

All format modifiers as supported by date() are supported too.

## Predefined formats:

```
<?php
date_default_timezone_set("Europe/Oslo");
$ts = new DateTime("December 22nd, 2005 15:41");
echo $ts->format(DATE_ISO8601), "<br/>\n";
echo $ts->format(DateTime::RFC1036), "<br/>\n";
echo $ts->format(DATE_RSS), "<br/>\n";
```

```
<?php
    $date = new DateTime( 'now' );
    echo $date->format(DateTime::ISO8601), "<br/>\n";

    $date->setTime( 15, 0, 7 );
    echo $date->format(DateTime::ISO8601), "<br/>\n";

    $date->setDate( 2008, 07, 22 );
    echo $date->format(DateTime::ISO8601), "<br/>\n";

    echo $date->setIsoDate( 2008, 45, 2 )->format(DateTime::ISO8601), "<br/>\n";

    // new in PHP 5.3:
    $date->setTimestamp( 1200128314 );
    echo $date->format(DateTime::ISO8601), "<br/>\n";
?>
```

## Modifying dates and times:

```
<?php
    date_default_timezone_set("Europe/Oslo");
    $ts = new DateTime("now");
    echo $ts->format(DATE_RFC2822), "<br/>\n";

    $ts->modify("+2 days");
    echo $ts->format(DATE_RFC2822), "<br/>\n";

    $ts->modify("fifth month");
    echo $ts->format(DATE_RFC2822), "<br/>\n";

    $ts->modify("Friday +3 weeks");
    echo $ts->format(DATE_RFC2822), "<br/>\n";

    $ts->modify("next friday");
    echo $ts->format(DATE_RFC2822), "<br/>\n";
?>
```

## Specifying timezone abbreviation while parsing:

```
<?php
    $ts = new DateTime("1978-12-22 09:15 CET");
?>
```

Using timezone abbreviations is deprecated, one should always use either a default timezone, or the full identifier.

## Specifying timezone identifier while parsing:

```
<?php
    $ts = new DateTime("1978-12-22 09:15 Europe/Oslo");
?>
```

## Setting a default timezone:

```
<?php
    date_default_timezone_set("Europe/Oslo");
    $ts = new DateTime("1978-12-22 09:15");
    echo $ts->format("e");
?>
```

## Getting a default timezone:

```
<?php
    $default_identifier = date_default_timezone_get();
    echo $default_identifier;
?>
```

Default timezone is 'guessed' in the following order:

- `date_default_timezone_set()` value
- TZ environment variable
- `php.ini`'s `date.timezone` setting
- System's rendering of timezone abbreviation

### Creating a timezone resource:

```
<?php
    $tz = new DateTimeZone("Asia/Singapore");
?>
```

### Using the timezone when parsing a string with a date representation:

```
<?php
    $tz = new DateTimeZone("Pacific/Honolulu");
    $ts = new DateTime("1978-12-22 09:15", $tz);
?>
```

### A passed timezone object does not override a parsed timezone:

```
<?php
    $tz = new DateTimeZone("Pacific/Honolulu");
    $ts1 = new DateTime("1978-12-22 09:15 CET", $tz);
    $ts2 = new DateTime("1978-12-22 09:15 Europe/Amsterdam", $tz);
    echo $ts2->format( DateTime::RFC2822 );
?>
```

### Getting a timezone's name:

```
<?php
    $tz = new DateTimeZone("Asia/Singapore");
    echo $tz->getName(), ', ';

    $tz = new DateTimeZone("CEST");
    echo $tz->getName();

?>
```

### Getting the current offset to UTC with a timezone for a specific date:

```
<?php
    $tz = new DateTimeZone("Europe/Amsterdam");
    $d = new DateTime("2005-01-22 09:15");
    echo $tz->getOffset($d), ', ';
    $d->modify("+6 months");
    echo $tz->getOffset($d);

?>
```

## Using the timezone when parsing a string with a date representation:

```
<?php
    $tz1 = new DateTimeZone("Pacific/Honolulu");
    $tz2 = new DateTimeZone("Europe/Amsterdam");
    $tz3 = new DateTimeZone("Australia/Melbourne");

    $ts = new DateTime("1978-12-22 09:15", $tz1);
    echo $ts->getTimezone()->getName(), ': ',
        $ts->format (DATE_RFC822), "<br/>";

    $ts->setTimezone ($tz2);
    echo $ts->getTimezone()->getName(), ': ',
        $ts->format (DATE_RFC822), "<br/>";

    $ts->setTimezone ($tz3);
    echo $ts->getTimezone()->getName(), ': ',
        $ts->format (DATE_RFC822);
?>
```

# Timezones Utilities

## Transition Times and Location Information

```
<?php
    $tz = new DateTimeZone("America/Caracas");
    $trs = $tz->getTransitions(strtotime('1960-01-01 UTC'));

    echo "<pre>\n";
    foreach ($trs as $tr) {
        printf("%20s %7d %d %s\n",
            $tr['time'], $tr['offset'], $tr['isdst'], $tr['abbr']);
    }

    $loc = $tz->getLocation();
    echo 'Info: ', join( ' - ', $loc ), "\n";
```

## All supported timezone identifiers:

```
<?php
    $ids = timezone_identifiers_list();
    echo "Number of identifiers: ", count($ids), "<br/>";
    echo implode(", ", array_slice($ids, 0, 5)), '...';
    echo implode(", ", array_slice($ids, -5));
?>
```

### Restrict by continent:

```
<?php
    $sids = timezone_identifiers_list( DateTimeZone::EUROPE );
    echo implode(", ", array_slice($sids, 0, 5)), '...';
?>
```

### Restrict by country:

```
<?php
    $sids = timezone_identifiers_list( DateTimeZone::PER_COUNTRY, 'US' );
    echo implode(", ", array_slice($sids, 0, 10)), '...';
?>
```

# Intervals (PHP 5.3)

## Creating an interval

```
<?php
// full notation
$i = new DateInterval( 'P0003-01-15T06:12:30' );

// abbreviated notation
$i = new DateInterval( 'P3DT6H' );

// from a difference
$d1 = new DateTime();
$d2 = new DateTime( "2009-09-09 09:09 Europe/Amsterdam" );
$i = $d1->diff( $d2 );

// displaying the difference
echo $i->format( '%a days, %h hours, %i minutes' ), "\n";
?>
<?php
// create from string
$i = DateInterval::createFromDateString( "next weekday" );
?>
```

# Intervals (PHP 5.3)

## Applying an interval

```
<pre><?php
$d = new DateTime( 'August 27th, 2009' );

$i = DateInterval::createFromDateString( "next weekday" );
echo $d->add( $i )->format( "l Y-m-d\n" );
echo $d->add( $i )->format( "l Y-m-d\n" );

$i = DateInterval::createFromDateString( "3 months 10 days" );
echo $d->sub( $i )->format( "l Y-m-d\n" );
?>
```

# Periods (PHP 5.3)

## Initializing a period

```
<?php
// start, interval and count
$db = new DateTime( '2008-07-31' );
$di = DateInterval::createFromDateString( 'next weekday' );
$p = new DatePeriod( $db, $di, 3 );

// start, interval and end
$de = new DateTime ( '2008-08-05' );
$p = new DatePeriod( $db, $di, $de );

// ISO 8601 string
$s = "2008-10-04T20:56:18Z/P0000-00-20T03:15:54/2008-12-22T00:00:00Z";
$p = new DatePeriod( $s );
?>
```

# Periods (PHP 5.3)

## Iterating over a period


```
<pre><?php
$db = new DateTime( '2008-12-31' );
$de = new DateTime( '2009-12-31' );
$di = DateInterval::createFromDateString( 'third tuesday of next month' );
$dpp = new DatePeriod( $db, $di, $de, DatePeriod::EXCLUDE_START_DATE );
foreach ( $dpp as $dt )
{
echo $dt->format( "F jS\n" );
}
?>
```

# php|architect's guide to Date/Time handling

## php | architect's Guide to **Date and Time Programming**



Derick Rethans

 nb<sup>™</sup> php | architect  
nanobooks

- This presentation: <http://derickrethans.nl/talks.php>
- World timezones:  
<http://www.worldtimezone.com/index24.php>
- Geonames: <http://www.geonames.org/>
- Questions?: [dr@ez.no](mailto:dr@ez.no)

# User's Timezone with a little javascript...

```
<?php
if (!isset($_GET['tzinfo'])) {
?>
<html>
<script type="text/javascript">
var d = new Date()
var tza = d.toLocaleString().split(" ").slice(-1)
var tzo = d.getTimezoneOffset()
window.location = window.location + '?tzinfo=' + tza + '|' + tzo
</script>
</html>
<?php
} else {
    list( $abbr, $offset ) = explode( '|', $_GET['tzinfo'] );
    echo timezone_name_from_abbr( $abbr, $offset * -60 );
}
?>
```

```
<pre><font size="5"><?php
date_default_timezone_set('America/New_York');
if (isset($_POST['country'])) {
mysql_connect('localhost', 'root', 'weel23'); mysql_select_db('geolocation');
$re = mysql_query( "SELECT * FROM city where country='{$_POST['country']}' AND
normalized_name='{$_POST['city']}'" );
$info = mysql_fetch_assoc( $re );
if ($info) {
    $lat = $info['lat']; $lon = $info['lon'];

    $info = date_sun_info(time(), $lat, $lon);
    echo "Sunrise is at ", date(DateTime::RSS, $info['sunrise']), "\n";
    echo "Sunset is at ", date(DateTime::RSS, $info['sunset']), "\n";
    echo "Day length is ",
    round(($info['sunset'] - $info['sunrise']) / 3600, 1), "hrs";
}
}
?></font></pre>
<form method="post">
Country: <input name="country"/> City: <input name="city"/><br/>
<input type="submit" value="Check"/></form>
```