NAME

Date::Manip::Calc − describes date calculations

SYNOPSIS

Two objects (both of which are either Date::Manip::Date or Date::Manip::Delta objects) may be used to creates a third object based on those two.

```
$delta  = $date->calc($date2 [,$subtract] [,$mode]);

$date2  = $date->calc($delta [,$subtract]);
$date2  = $delta->calc($date1 [,$subtract]);

$delta3 = $delta1->calc($delta2 [,$subtract] [,$no_normalize]);
```

DESCRIPTION

This document describes the different types of calculations that can be done using dates and deltas. Date calculations are much more complicated than they initially appear, so this document is fairly large.

The complication in date calculations is due to the fact that it is impossible to express some parts of a delta as an exact length. Some examples will illustrate this:

As an example, let's take two dates and determine how much time elapsed between them:

```
Nov 3 2016 11:00:00
Dec 5 2016 12:00:00

Elapsed time: 770 hours
```

There are several ways to describe the time that elapsed. The first way is to give the difference exactly. This is the exact delta.

An exact delta is always described in terms of hours, minutes, and seconds.

The problem with this is that we don't think in terms of exact deltas. We think in terms which cannot be expressed exactly.

For example, most people would look at those two dates and think:

```
Perceived: 1 month, 2 days, 1 hour
```

But the two dates:

```
Feb 3 2016 11:00:00
Mar 5 2016 12:00:00

Elapsed time: 745 hours
Perceived: 1 month, 2 days, 1 hour
```

Some fields in a delta do not have an exact length. A year is usually 365 days long, but sometimes it is 366. A month might be 28, 29, 30, or 31 days long.

Perhaps the most unexpected difficulty is that days are not of constant length. Most people would define a day as 24 hours, but when you take daylight saving time into account that definition produces unexpected results. The following calculation illustrates this:

```
Nov 5, 2011 02:30 EDT
+ 24 hour

Result: Nov 6, 2011 01:30 EST
```

This immediately causes most people to redefine a day as the amount of time between the same wall clock time. For example, the amount of time between noon one day and noon the next (regardless of daylight saving time changes).

This definition doesn't work either. For example:

```
Mar 12, 2011 02:30 EST
+ 1 day (same time next day)

Result: Mar 13 02:30 EST
```

But that date does not exist! Neither does:

```
Result: Mar 13 02:30 EDT
```

An alternate calculation could be:

```
Nov 5, 2011 01:30 EDT
+ 1 day (same time next day)

Result: Nov 6, 01:30 EDT
Result: Nov 6, 01:30 EST
```

Both of those results exist. Which result did you mean? The first one is probably correct (since it is 24 hours later), but an hour later, you will have the same clock time again.

So, the same time next day definition doesn't work at all for some dates (during a 'spring forward' type daylight saving time transition) and is ambiguous for others (during a 'fall back' type daylight saving time transition).

Calculations involving exact deltas are unambiguous in all cases.

A second class of delta is called a semi-exact delta, and these add days (and weeks) to the delta, and treats days as a ''same time next day'' at all times except the two cases where the resulting date falls in the period where a daylight saving time transition is occurring. Then it falls back to the 24 hour definition.

A final class of delta is an approximate delta which includes all of the fields (years and months). This allows Date::Manip to handle deltas in a way that is consistent with how most people perceive the elapsed time. It should be noted that there is some uncertainty there as not everyone's definition of how a delta is perceived is the same, but in general, they should be closer to what most people think of.

TYPES OF CALCULATIONS

This document describes the different types of calculations. Calculations involve two types of Date::Manip objects: dates and deltas. These are described in the Date::Manip::Date and Date::Manip::Delta manuals respectively.

Two objects (two dates, two deltas, or one of each) are used. In all cases, if a second object is not passed in, undef is returned.

There are 3 types of calculations:

Date/Date calculations

A calculation involving 2 dates is used to determine the amount of time (the delta) between them.

```
$delta  = $date1->calc($date2 [,$subtract] [,$mode]);
```

Two dates can be worked with and a delta will be produced which is the amount of time between the two dates.

`$date1` and `$date2` are Date::Manip::Date objects with valid dates. The Date::Manip::Delta object returned is the amount of time between them. If `$subtract` is not passed in (or is 0), the delta produced is:

```
DELTA = DATE2 - DATE1
```

If `$subtract` is non-zero, the delta produced is:

```
DELTA = DATE1 - DATE2
```

The `$subtract` argument has special importance when doing approximate calculations, and this is described below.

If either date is invalid, a delta object will be returned which has an error associated with it.

The `$mode` argument describes the type of delta that is produced and is described below in ''MODE''.

Date/Delta calculations

Date/delta calculations can be performed using either a Date::Manip::Date or Date::Manip::Delta object as the primary object:

```
$date2  = $date1->calc($delta [,$subtract]);
$date2  = $delta->calc($date1 [,$subtract]);
```

A date and delta can be combined to yield a date that is the given amount of time before or after it.

`$date1` and `$delta` are Date::Manip::Date and Date::Manip::Delta objects respectively. A new Date::Manip::Date object is produced. If either `$date1` or `$delta` are invalid, the new date object will have an error associated with it.

Both of the calls above perform the same function and produce exactly the same results.

If `$subtract` is not passed in, or is 0, the resulting date is formed as:

```
DATE2 = DATE1 + DELTA
```

If `$subtract` is non-zero, the resulting date is:

```
DATE2 = DATE1 - DELTA
```

The `$subtract` argument has special importance when doing approximate calculations, and this is described below in ''SUBTRACTION''.

Delta/Delta calculations

Delta/delta calculations can be performed to add two amounts of time together, or subtract them.

```
$delta3 = $delta1->calc($delta2 [,$subtract] [,$no_normalize]);
```

If `$subtract` is not passed in, or is 0, the resulting delta formed is:

```
DELTA3 = DELTA1 + DELTA2
```

If `$subtract` is non-zero, then the resulting delta is:

```
DELTA3 = DELTA1 - DELTA2
```

`$delta1` and `$delta2` are valid Date::Manip::Delta objects, and a new Date::Manip::Delta object is produced.

`$no_normalize` can be the string 'nonormalize' or a non-zero value (in which case `$subtract` MUST be entered, even if it is 0).

MODE

Date::Manip calculations can be divided into two different categories: business and non-business; and within those are three sub-categories: exact, semi-exact, and approximate.

Business and non-business calculations

A business calculation is one where the length of the day is determined by the length of the work day, and only business days (i.e. days in which business is conducted) count. Holidays and weekends are omitted (though there is some flexibility in defining what exactly constitutes the work week as described in the Date::Manip::Config manual). This is described in more detail below in ''BUSINESS MODE CONSIDERATIONS''.

A non-business mode calculation is the normal type of calculation where no days are ignored, and all days are full length.

Exact, semi-exact, and approximate calculations

An exact calculation is one in which the delta used (or produced) is an exact delta. An exact delta is described in more detail in the Date::Manip::Delta manual, but the short explanation is that it is a delta

which only involves fields of an exactly known length (hours, minutes, and seconds). Business deltas also include days in the exact part. The value of all other fields in the delta will be zero.

A semi-exact calculation is one in which the deltas used (or produced) is a semi-exact delta. This is also described in the Date::Manip::Delta manual, but the short explanation is that it includes days and weeks (for standard calculations) or weeks (for business calculations) in addition to the exact fields. A semi-exact day is defined as the same clock time on two successive days. So noon to noon is 1 day (even though it may not be exactly 24 hours due to a daylight saving time transition). A week is defined as 7 days. This is described in more detail below.

An approximate calculation is one in which the deltas used (or produced) are approximate, and may include any of the fields.

In date-delta and delta-delta calculations, the mode of the calculation will be determined automatically by the delta. In the case of date-date calculations, the mode is supplied as an argument.

Mode in date-date calculations

When doing a date-date calculation, the following call is used:

```
$delta = $date1->calc($date2 [,$subtract] [,$mode]);
```

`$mode` defaults to ''exact''. The delta produced will be be either a business or non-business delta; exact, semi-exact, or approximate, as specified by `$mode`.

Currently, the possible values that `$mode` can have are:

```
exact    : an exact, non-business calculation
semi     : a semi-exact, non-business calculation
approx   : an approximate, non-business calculation

business : an exact, business calculation
bsemi    : a semi-exact, business calculation
bapprox  : an approximate, business calculation
```

Mode in date-delta calculations

When doing calculations of a date and a delta:

```
$date2 = $date1->calc($delta [,$subtract]);
$date2 = $delta->calc($date1 [,$subtract]);
```

the mode is not passed in. It is determined exclusively by the delta. If `$delta` is a business delta, A business calculation is done. If `$delta` is a non-business delta, a non-business calculation will be done.

The `$delta` will also be classified as exact, semi-exact, or approximate based on which fields are non-zero.

Mode in delta-delta calculations

When doing calculations with two deltas:

```
$delta3 = $delta1->calc($delta2 [,$subtract]);
```

the mode is not passed in. It is determined by the two deltas.

If both deltas are business mode, or both are non-business mode, a new delta will be produced of the same type.

It one of the deltas is a business mode and the other is not, the resulting delta will have an error condition since there is no direct correlation between the two types of deltas. Even though it would be easy to add the two together, it would be impossible to come up with a result that is meaningful.

If both deltas are exact, semi-exact, or approximate, the resulting delta is the same. If one delta is approximate and one is not, then the resulting delta is approximate. It is NOT treated as an error. Likewise, if one is semi-exact and the other exact, a semi-exact delta is produced.

TIMEZONE CONSIDERATIONS

date-date calculations

When doing a business calculation, both dates must be in the same time zone or an error is produced.

For non-business calculations, when calculating the difference between two dates in different time zones, `$date2` will be converted to the same timezone as `$date1` and the returned date will be in this timezone.

date-delta calculations

When adding a delta to a date, the resulting date will be in the same time zone as the original date.

delta-delta calculations

No timezone information applies.

It should also be noted that daylight saving time considerations are currently ignored when doing business calculations. In common usage, daylight saving time changes occurs outside of the business day, so the business day length is constant. As a result, daylight saving time is ignored.

BUSINESS MODE CONSIDERATIONS

In order to correctly do business mode calculations, a config file should exist which contains the section defining holidays (otherwise, weekends will be ignored, but all other days will be counted as business days). This is documented below, and in the Date::Manip::Config section of the documentation. Some config variables (namely WorkWeekBeg, WorkWeekEnd, WorkDayBeg, WorkDayEnd, and WorkDay24Hr) defined the length of the work week and work day.

If the workday is defined as 08:00 to 18:00, a work week consisting of Mon-Sat, and the standard (American) holidays, then from Tuesday at 12:00 to the following Monday at 14:00 is 5 days and 2 hours. If the ''end'' of the day is reached in a calculation, it automatically switches to the next day. So, Tuesday at 12:00 plus 6 hours is Wednesday at 08:00 (provided Wed is not a holiday). Also, a date that is not during a workday automatically becomes the start of the next workday. So, Sunday 12:00 and Monday at 03:00 both automatically becomes Monday at 08:00 (provided Monday is not a holiday).

Note that a business week is treated the same as an exact week (i.e. from Tuesday to Tuesday, regardless of holidays). Because this means that the relationship between days and weeks is NOT unambiguous, when a semi-exact delta is produced from two dates, it will be in terms of d/h/mn/s (i.e. no week field).

Anyone using business mode is going to notice a few quirks about it which should be explained. When I designed business mode, I had in mind what a business which promises 1 business day turnaround really means.

If you do a business calculation (with the workday set to 9:00−17:00), you will get the following:

```
Saturday at noon + 1 business day = Tuesday at 9:00
Saturday at noon − 1 business day = Friday at 9:00
```

What does this mean?

As an example, say I use a business that works 9−5 and they have a drop box so I can drop things off over the weekend and they promise 1 business day turnaround. If I drop something off Friday night, Saturday, or Sunday, it doesn't matter. They're going to get started on it Monday morning. It'll be 1 business day to finish the job, so the earliest I can expect it to be done is around 17:00 Monday or 9:00 Tuesday morning. Unfortunately, there is some ambiguity as to what day 17:00 really falls on, similar to the ambiguity that occurs when you ask what day midnight falls on. Although it's not the only answer, Date::Manip treats midnight as the beginning of a day rather than the end of one. In the same way, 17:00 is equivalent to 9:00 the next day and any time the date calculations encounter 17:00, it automatically switch to 9:00 the next day. Although this introduces some quirks, I think this is justified. I also think that it is the way most people think of it. If I drop something off first thing Monday morning, I would expect to pick it up first thing Tuesday if there is 1 business day turnaround.

Equivalently, if I want a job to be finished on Saturday (despite the fact that I cannot pick it up since the business is closed), I have to drop it off no later than Friday at 9:00. That gives them a full business day to finish it off. Of course, I could just as easily drop it off at 17:00 Thursday, or any time between then and

9:00 Friday. Again, it's a matter of treating 17:00 as ambiguous.

So Saturday + 1 business day = Tuesday at 9:00 (which means anything from Monday 17:00 to Tuesday 9:00), but Monday at 9:01 + 1 business day = Tuesday at 9:01 which is unambiguous.

It should be noted that when adding years, months, and weeks, the business day is ignored. Once they've been added, the resulting date is forced to be a business time (i.e. it moves to the start of the next business day if it wasn't one already) before proceeding with the days, hours, minutes, and seconds part.

EXACT, SEMI-EXACT, AND APPROXIMATE DATE/DELTA CALCULATIONS

This section contains more details about exactly how exact, semi-exact, and approximate calculations are performed for date/delta calculations.

All calculations make use of some exact quantities, including:

```
1 year   = 12 months
1 week   = 7 days
1 hour   = 60 minutes
1 minute = 60 seconds
```

This leaves two relationships which are not exact:

```
1 month  = ? days
1 day    = ? hours
```

For non-business calculations, a day is usually 24 hours long. Due to daylight saving time transitions which might make a day be 23 or 25 hours long (or in some cases, some other length), the relation is not exact. Whenever possible, a day is actually measured as the same time on two days (i.e. Tuesday at noon to Wednesday at noon) even if that period is not precisely 24 hours. For business calculations, a days length is determined by the length of the work day and is known exactly.

Exact calculations involve ONLY quantities of time with a known length, so there is no ambiguity in them.

Approximate and semi-exact calculations involve variable length fields, and so they must be treated specially.

In order to do an approximate or semi-exact calculation, the delta is added to a date in pieces, where the fields in each piece have an exact and known relationship.

For a non-business calculation, a calculation occurs in the following steps:

```
year/month fields added
week/day fields added
hour/minute/second fields added
```

For a business calculation, the steps are:

```
year/month fields added
week field added
day field added
hour/minute/second fields added
```

After each step, a valid date must be present, or it will be adjusted before proceeding to the next step. Note however that for business calculations, the first step must produce a valid date, but not necessarily a business date. The second step will produce a valid business date.

A series of examples will illustrate this.

A date and non-business approximate delta

```
date  = Mar 31 2001 at 12:00:00
delta = 1 year, 1 month, 1 day, 1 hour
```

First, the year/month fields are added without modifying any other field. This would produce:

```
Apr 31, 2002 at 12:00
```

which is not valid. Any time the year/month fields produce a day past the end of the month, the result

is 'truncated' to the last day of the month, so this produces:

```
Apr 30, 2002 at 12:00
```

Next the week/day fields are added producing:

```
May 1, 2002 at 12:00
```

and finally, the exact fields (hour/minute/second) are added to produce:

```
May 1, 2002 at 13:00
```

A simple business calculation

Assuming a normal Monday-Friday work week from 8:00 − 17:00:

```
date  = Wed, Nov 23, 2011 at 12:00
delta = 1 week, 1 day, 1 hour
```

First, the week field is added:

```
Wed, Nov 30, 2011 at 12:00
```

Then the day field is added:

```
Thu, Dec 1, 2011 at 12:00
```

Then the exact fields are added:

```
Thu, Dec 1, 2011 at 13:00
```

A business example where a holiday impacts it

In America, Jul 4 is a holiday, so Mon, Jul 4, 2011 is not a work day.

```
date  = Mon, Jun 27, 2011 at 12:00
delta = 1 week, 1 day, 1 hour
```

First, the week field is added:

```
Mon, Jul 4, 2011 at 12:00
```

Since that is not a work day, it immediately becomes:

```
Tue, Jul 5, 2011 at 8:00
```

Then the day field is added:

```
Wed, Jul 6, 2011 at 8:00
```

and finally the remaining fields:

```
Wed, Jul 6, 2011 at 9:00
```

Calculation where daylight savings time impacts it (fall example)

In the America/New_York timezone (Eastern time), on November 6, 2011, the following time change occurred:

```
2011-11-06 02:00  EDT  => 2011-11-06 01:00  EST
```

Three simple calculations illustrate how this is handled:

```
date  = 2011-11-05 02:30 EDT
delta = 1 day
```

Adding the day produces:

```
2011-11-06 02:30  EDT
```

which is valid, so that is the result.

Similarly:

```
date  = 2011-11-07 02:30 EST
delta = -1 day
```

produces:

```
2011-11-06 02:30 EST
```

which is valid.

Finally:

```
date  = 2011-11-05 02:30 EDT
delta = 2 days
```

produces:

```
2011-11-07 02:30  EST
```

The calculation will preserve the savings time where possible so the resulting day will have the same offset from UTC. If that is not possible, but the resulting day is valid in the other offset, that will be used instead.

Calculation where daylight savings time impacts it (spring example)

In the America/New_York timezone (Eastern time), on March 13, the following time change occurred:

```
2011-03-13 02:00  EST  => 2011-03-13 03:00  EDT
```

In this case, a calculation may produce an invalid date.

```
date  = 2011-03-12 02:30 EST
delta = 1 day
```

produces:

```
2011-03-13 02:30 EST
```

This is not valid. Neither is:

```
2011-03-13 02:30 EDT
```

In this case, the calculation will be redone converting days to 24-hour periods, so the calculation becomes:

```
date  = 2011-03-12 02:30 EST
delta = 24 hours
```

which will produce a valid date:

```
2011-03-13 03:30 EDT
```

EXACT, SEMI-EXACT, AND APPROXIMATE DATE/DATE CALCULATIONS

This section contains more details about exactly how exact, semi-exact, and approximate calculations are performed for date/date calculations.

When calculating the delta between two dates, the delta may take different forms depending on the mode passed in. An exact calculation will produce a delta which included only exact fields. A semi-exact calculation may produce a semi-exact delta, and an approximate calculation may produce an approximate delta. Note that if the two dates are close enough together, an exact delta will be produced (even if the mode is semi-exact or approximate), or it may produce a semi-exact delta in approximate mode.

For example, the two dates ‘‘Mar 12 1995 12:00’’ and ‘‘Apr 13 1995 12:00’’ would have an exact delta of ‘‘744 hours’’, and a semi-exact delta of ‘‘31 days’’. It would have an approximate delta of ‘‘1 month 1 day’’.

Two dates, ‘‘Mar 31 12:00’’ and ‘‘Apr 30 12:00’’ would have deltas ‘‘720 hours’’ (exact), ‘‘30 days’’ (semi-exact) or ‘‘1 month’’ (approximate).

Approximate mode is a more human way of looking at things (you’d say 1 month and 2 days more often

then 33 days), but it is less meaningful in terms of absolute time.

One thing to remember is that an exact delta is exactly the amount of time that has passed, including all effects of daylight saving time. Semi-exact and approximate deltas usually ignore the affects of daylight saving time.

SUBTRACTION

In exact and semi-exact calculations, and in delta-delta calculations, the the `$subtract` argument is easy to understand. When working with an approximate delta however (either when adding an approximate delta to a date, or when taking two dates to get an approximate delta), there is a degree of uncertainty in how the calculation is done, and the `$subtract` argument is used to specify exactly how the approximate delta is to be use. An example illustrates this quite well.

If you take the date Jan 4, 2000 and subtract a delta of ''1 month 1 week'' from it, you end up with Nov 27, 1999 (Jan 4, 2000 minus 1 month is Dec 4, 1999; minus 1 week is Nov 27, 1999). But Nov 27, 1999 plus a delta of ''1 month 1 week'' is Jan 3, 2000 (Nov 27, 1999 plus 1 month is Dec 27, 1999; plus 1 week is Jan 3, 2000).

In other words the approximate delta (but NOT the exact or semi-exact delta) is different depending on whether you move from earlier date to the later date, or vice versa. And depending on what you are calculating, both are useful.

In order to resolve this, the `$subtract` argument can take on the values 0, 1, or 2, and have different meanings.

`$subtract` in approximate date-date calculations

In the call:

```
$delta = $date1->calc($date2,$subtract,"approx");
```

if `$subtract` is 0, the resulting delta can be added to `$date1` to get `$date2`. Obviously `$delta` may still be negative (if `$date2` comes before `$date1`).

If `$subtract` is 1, the resulting delta can be subtracted from `$date1` to get `$date2` (the deltas from these two are identical except for having an opposite sign).

If `$subtract` is 2, the resulting delta can be added to `$date2` to get `$date1`. In other words, the following are identical:

```
$delta = $date1->calc($date2,2,"approx");
$delta = $date2->calc($date1,"approx");
```

`$subtract` in approximate date-delta calculations

In the call:

```
$date2 = $date1->calc($delta,$subtract);
```

If `$subtract` is 0, the resulting date is determined by adding `$delta` to `$date1`.

If `$subtract` is 1, the resulting date is determined by subtracting `$delta` from `$date1`.

If `$subtract` is 2, the resulting date is the date which `$delta` can be added to to get `$date1`.

For business mode calculations, `$date1` will first be adjusted to be a valid work day (if it isn't already), so this may lead to non-intuitive results.

In some cases, it is impossible to do a calculation with `$subtract` = 2. As an example, if the date is ''Dec 31'' and the delta is ''1 month'', there is no date which you can add ''1 month'' to to get ''Dec 31''. When this occurs, the date returned has an error flag.

APPROXIMATE DATE/DATE CALCULATION

There are two different ways to look at the approximate delta between two dates.

In Date::Manip 5.xx, the approximate delta between the two dates:

```
Jan 10 1996 noon
Jan  7 1998 noon
```

was 1:11:4:0:0:0:0 (or 1 year, 11 months, 4 weeks). In calculating this, the first date was adjusted as far as it could go towards the second date without going past it with each unit starting with the years and ending with the seconds.

This gave a strictly positive or negative delta, but it isn't actually how most people would think of the delta.

As of Date::Manip 6.0, the delta is 2:0:0:−3:0:0:0 (or 2 years minus 3 days). Although this leads to mixed-sign deltas, it is actually how more people would think about the delta. It has the additional advantage of being easier to calculate.

For non-business mode calculations, the year/month part of the approximate delta will move a date from the year/month of the first date into the year/month of the second date. The remainder of the delta will adjust the days/hours/minutes/seconds as appropriate.

For approximate business mode calculations, the year, date, and week parts will be done approximately, and the remainder will be done exactly.

KNOWN BUGS

None known.

BUGS AND QUESTIONS

Please refer to the Date::Manip::Problems documentation for information on submitting bug reports or questions to the author.

SEE ALSO

Date::Manip − main module documentation

LICENSE

AUTHOR

Sullivan Beck (sbeck@cpan.org)