NAME

Date::Manip::Changes5to6 − describes differences between 5.xx and 6.00

SYNOPSIS

Date::Manip 6.00 represents a complete rethink and rewrite of Date::Manip. A great deal of effort was made to make sure that 6.00 is almost backwards compatible with 5.xx whenever feasible, but some functionality has changed in backwards incompatible ways. Other parts have been deprecated and will be removed at some point in the future.

This document describes the differences between the 5.xx series and version 6.00. This page primarily describes technical details, most of which do not impact how Date::Manip is used in scripts. If you want to make sure that a script which ran with 5.xx will run with 6.xx, refer to the Date::Manip::Migration5to6 document.

OVERVIEW

The Date::Manip 5.xx series of suffered from several weaknesses. These included:

Poor time zone support

Time zone support in 5.xx was broken. Determining a time zone, and understanding daylight saving time changes was incomplete (at best) and totally inadequate to do true timezone operations.

Parsing too complicated and unstructured

The parsing routines had grown very complicated, and overly permissive over time and were in need of a complete overhaul.

Lacking OO model

Date::Manip 5.xx was written as a functional module, not an OO module, but date handling would lend itself very well to being OO with different classes to handle dates, deltas, and recurrences.

The OO model allows a lot of information to be stored with each date (such as time zone information) which is discarded in the functional interface.

Too monolithic

The entire Date::Manip module was contained in one huge file. Breaking up the module would make it much easier to deal with.

Date::Manip 6.00 is a complete rewrite of Date::Manip to address these and other issues.

The following sections address how Date::Manip 6.00 differs from previous releases, and describes changes that might need to be made to your script in order to upgrade from 5.xx to 6.00.

The most important changes are marked with asterisks.

GENERAL CHANGES

(*) Requires perl 5.10.0

Please see the Date::Manip::Problems document for a discussion of this problem. It's in the KNOWN COMPLAINTS section.

(*) Breaking into smaller modules

Date::Manip module has been broken up from one huge module into a large number of smaller more manageable modules. The main Date::Manip module is still present, and contains all of the functions from Date::Manip 5.xx (except that they now call functions from all the other modules to do the actual work). In general, the Date::Manip module from 6.00 is backwards compatible.

A number of new modules have been created as well. These can be used directly, bypassing the main Date::Manip module. These include the following:

Date::Manip::Base contains many basic date operations which may be used to do simple date manipulation tasks without all the overhead of the full Date::Manip module.

Date::Manip::TZ contains time zone operations.

Handling dates, deltas, and recurrences are now done in Date::Manip::Date, Date::Manip::Delta, and Date::Manip::Recur.

All of these modules are object oriented, and are designed to be used directly, so if you prefer an OO interface over a functional interface, use these modules.

(*) Intermediate data cached

In order to improve the performance of Date::Manip, many intermediate values are cached. This does impact the memory footprint of the module, but it has a huge impact on the performance of the module.

Some types of data depend on the config variables used, and these are cached separately, and this cache is automatically cleared every time a config variable is set. As a result, it is best if you set all config variables at the start, and then leave them alone completely to get optimal use of cached data.

A side effect of all this is that the Memoize module should not be used in conjunction with Date::Manip.

In the version 5.xx documentation, it was mentioned that the Memoize module might be used to improve performance in some cases. This is no longer the case. It should not be used with Date::Manip, even if you use the functional interface instead of the OO interface.

Taint safe

Date::Manip now contains no tainted data, and should run without problems with taint checking on provided you do not set additional methods for determining the system time zone using the curr_zone_methods function.

Ideally, this should never be necessary. If it is necessary, I'd like to hear about it so that I can add whatever standard methods are needed to the built in list.

TIME ZONE SUPPORT

(*) Complete handling of time zones

The biggest problem with Date::Manip 5.xx was it's inability to correctly handle time zones and Daylight Saving Time. That is now fixed. Version 6.00 includes support for every time zone included in the zoneinfo (aka Olson) database which includes the definitions of (hopefully) all of the time zones used in the world.

Individual time zones will no longer be added

Prior to 5.55, time zones were added upon request. Since 6.00 now supports a full set of standard time zones, I will no longer add in individual time zones (Date::Manip::TZ includes functionality for adding them yourself if they are needed). With Date::Manip now having full time zone support, I'm not interested in supporting my own time zone database.

However, I am interested in adding sets of time zones from various ''standards''.

Date::Manip 6.00 includes time zones from the following standards:

```
Olson zoneinfo database
all Microsoft Windows time zones
zones listed in RFC-822
```

If there are additional standards that include additional time zones not included here, please point me to them so they can be added. This could include published lists of time zone names supported on some operating system which have different names than the zoneinfo list.

Nonstandard time zone abbreviations removed

Some of the individual standards that were added in the 5.xx series are not included in any of the standards listed above.

As of 6.00, only time zones from standards will be included in the distribution (others can be added by users using the functions described in Date::Manip::TZ to add aliases for existing time zones).

The following time zones were in Date::Manip 5.xx but not in 6.00.

```
IDLW    −1200    International Date Line West
NT      −1100    Nome
SAT     −0400    Chile
CLDT    −0300    Chile Daylight
AT      −0200    Azores
MEWT    +0100    Middle European Winter
MEZ     +0100    Middle European
FWT     +0100    French Winter
GB      +0100    GMT with daylight saving
SWT     +0100    Swedish Winter
MESZ    +0200    Middle European Summer
FST     +0200    French Summer
METDST  +0200    An alias for MEST used by HP−UX
EETDST  +0300    An alias for eest used by HP−UX
EETEDT  +0300    Eastern Europe, USSR Zone 1
BT      +0300    Baghdad, USSR Zone 2
IT      +0330    Iran
ZP4     +0400    USSR Zone 3
ZP5     +0500    USSR Zone 4
IST     +0530    Indian Standard
ZP6     +0600    USSR Zone 5
AWST    +0800    Australian Western Standard
ROK     +0900    Republic of Korea
AEST    +1000    Australian Eastern Standard
ACDT    +1030    Australian Central Daylight
CADT    +1030    Central Australian Daylight
AEDT    +1100    Australian Eastern Daylight
EADT    +1100    Eastern Australian Daylight
NZT     +1200    New Zealand
IDLE    +1200    International Date Line East
```

A lot of support modules and files

Date::Manip now includes a large number of files and modules that are used to support time zones.

A series of modules are included which are auto-generated from the zoneinfo database. The Date::Manip::Zones, Date::Manip::TZ::*, and Date::Manip::Offset::* modules are all automatically generated and are not intended to be used directly. Instead, the Date::Manip::TZ module is used to access the data stored there.

A separate time zone module (Date::Manip::TZ::*) is included for every single time zone. There is also a module (Date::Manip::Offset::*) for every different offset. All told, there are almost 1000 modules. These are included to make time zone handling more efficient. Rather than calculating everything on the fly, information about each time zone and offset are included here which greatly speeds up the handling of time zones. These modules are only loaded as needed (i.e. only the modules related to the specific time zones you refer to are ever loaded), so there is no performance penalty to having them.

Also included in the distribution are a script (tzdata) and additional module (Date::Manip::TZdata). These are used to automatically generate the time zone modules, and are of no use to anyone other than the maintainer of Date::Manip. They are included solely for the sake of completeness. If someone wanted to fork Date::Manip, all the tools necessary to do so are included in the distribution.

(*) Meaning of `$::TZ` and `$ENV{TZ}`

In Date::Manip 5.x, you could specify what time zone you wanted to work in using either the `$::TZ` or `$ENV{TZ}` variables.

Date::Manip 6.00 makes use of two different time zones: the actual local time zone the computer is running in (and which is used by the system clock), and a time zone that you want to work in.

Typically, these are the same, but they do not have to be.

As of Date::Manip 6.00, the `$::TZ` and `$ENV{TZ}` variables are used only to specify the actual local time zone.

In order to specify an alternate time zone to work in, use the SetDate or ForceDate config variables.

CONFIG FILES AND VARIABLES

(*) Date_Init handling of config variables

The handling of config variables has changed slightly.

Previously, variables passed in to Date_Init overrode values from config files. This has changed slightly. Options to Date_Init are now parsed in the order they are listed, so the following:

```
Date_Init("DateFormat=Other","ConfigFile=DateManip.cnf")
```

would first set the DateFormat variable, and then it would read the config file ‘‘DateManip.cnf’’. If that config file included a DateFormat definition, it would override the one passed in to Date_Init.

The proper way to override config files is to pass the config files in first, followed by any script-specific overrides. In other words:

```
Date_Init("ConfigFile=DateManip.cnf","DateFormat=Other")
```

Date_Init doesn’t return the config variables

In Date::Manip::5.xx, Date_Init could return the list of all config variables. This functionality is no longer supported. Date_Init is used strictly to set config variables.

(*) Config file options

Date::Manip 5.xx had the concept of a global and personal config file. In addition, the personal config file could be looked for in a path of directories. All this was specified using the config variables:

```
GlobalCnf
IgnoreGlobalCnf
PersonalCnf
PersonalCnfPath
PathSep
```

All of these have been removed. Instead, the single config variable:

```
ConfigFile
```

will be used to specify config files (with no distinction between a global and personal config file). Also, no path searching is done. Each must be specified by a complete path. Finally, any number of config files can be used. So the following is valid:

```
Date_Init("ConfigFile=./Manip.cnf","ConfigFile=/tmp/Manip.cnf")
```

Other config variables removed

The following config variables have been removed.

```
TodayIsMidnight  Use DefaultTime instead.

ConvTZ           Use SetDate or ForceDate instead.

Internal         Use Printable instead.

DeltaSigns       Use the Date::Manip::Delta::printf
                 method to print deltas

UpdateCurrTZ     With real time zone handling in
                 place, this is no longer necessary
```

```
IntCharSet      This has been replaced with better support for
                international character sets. The Encoding config
                variable may be used instead.
```

Other config variables deprecated

The following config variables are deprecated and will be removed in some future version:

```
TZ              Use SetDate or ForceDate instead.
```

Holidays

Previously, holidays could be defined as a ‘‘Date + Delta’’ or ‘‘Date − Delta’’ string. These predate recurrences, and introduce some complexity into the handling of holidays. Since recurrences are a much better way to define holidays, the ‘‘Date + Delta’’ and ‘‘Date − Delta’’ strings are no longer supported.

TZ replaced (and enhanced)

The SetDate and ForceDate variables (which include the functionality of the deprecated TZ variable) are much improved as described in the Date::Manip::Config documentation.

Since it is now handles time change correctly (allowing time changes to occur in the alternate time zone), parsed results may be different than in 5.x (but since 5.x didn’t have proper time zone handling, this is a good thing).

DATE PARSING AND OPERATIONS

(*) today, tomorrow, yesterday

The words ‘‘today’’, ‘‘tomorrow’’, and ‘‘yesterday’’ in 5.xx referred to the time now, 24 hours in the future, and 24 hours in the past respectively.

As of 6.00, these are treated strictly as date strings, so they are the current day, the day before, or the day after at the time 00:00:00.

The string ‘‘now’’ still refers to the current date and time.

ISO 8601 formats

A couple of the date formats from Date::Manip 5.xx conflicted with ISO 8601 formats in the spec. These are documented in the Date::Manip::Date documentation.

Dates are now parsed according to the spec (though a couple extensions have been made, which are also documented in the Date::Manip::Date documentation).

There is one change with respect to Date::Manip 5.xx that results from a possible misinterpretation of the standard. In Date::Manip, there is a small amount of ambiguity in how the Www-D date formats are understood.

The date:

```
1996-w02-3
```

might be interpreted in two different ways. It could be interpreted as Wednesday (day 3) of the 2nd week of 1996, or as the 3rd day of the 2nd week of 1996 (which would be Tuesday if the week begins on Sunday). Since the specification only works with weeks which begin on day 1, the two are always equivalent in the specification, and the language of the specification doesn’t clearly indicate one interpretation over the other.

Since Date::Manip supports the concept of weeks starting on days other than day 1 (Monday), the two interpretations are not equivalent.

In Date::Manip 5.xx, the date was interpreted as Wednesday of the 2nd week, but I now believe that the other interpretation (3rd day of the week) is the interpretation intended by the specification. In addition, if this interpretation is used, it is easy to get the other interpretation.

If 1996−w02−3 means the 3rd day of the 2nd week, then to get Wednesday (day 3) of the week, use the following two Date::Manip::Date methods:

```
$err   = $date->parse("1996-w02-1");
$date2 = $date->next(3,1);
```

The first call gets the 1st day of the 2nd week, and the second call gets the next Wednesday.

If 1996-w02-3 is interpreted as Wednesday of the 2nd week, then to get the 3rd day of the week involves significantly more work.

In Date::Manip 6.00, the date will now be parsed as the 3rd day of the 2nd week.

(*) Parsing is now more rigid

The philosophy in Date::Manip 5.xx with respect to parsing dates was ''if there's any conceivable way to find a valid date in the string, do so''. As a result, strings which did not look like they could contain a valid date often would.

This manifested itself it two ways. First, a lot of punctuation was ignored. For example, the string ''01 // 03 -. 75'' was the date 1975-01-03.

Second, a lot of word breaks were optional and it was often acceptable to run strings together. For example, the delta ''in5seconds'' would have worked.

With Date::Manip 6.00, parsing now tries to find a valid date in the string, but uses a more rigidly defined set of allowed formats which should more closely match how the dates would actually be expressed in real life. The punctuation allowed is more rigidly defined, and word breaks are required. So ''01/03/75'' will work, but ''01//03/75'' and ''01/03-75'' won't. Also, ''in5seconds'' will no longer work, though ''in 5 seconds'' will work.

These changes serve to simplify some of the regular expressions used in parsing dates, as well as simplifying the parsing routines. They also help to recognize actually dates as opposed to typos... it was too easy to pass in garbage and get a date out.

Support dropped for a few formats

I've dropped support for a few very uncommon (probably never used) formats. These include (with Jan 3, 2009 as an example):

```
DD/YYmmm      03/09Jan
DD/YYYYmmm    03/2009Jan
mmmYYYY/DD    Jan2009/03
YYYY/DDmmm    2009/03Jan

mmmYYYY       Jan2009
YYYYmmm       2009Jan
```

The last two are no longer supported since they are incomplete.

With the exception of the incomplete forms, these could be added back in with very little effort. If there is ever a request to do so, I probably will.

No longer parses the Apache format

Date::Manip 5.xx supported the format:

```
DD/mmm/YYYY:HH:MN:SS
```

used in the apache logs. Due to the stricter parsing, this format is no longer supported directly. However, the parse_format method may be used to parse the date directly from an apache log line with no need to extract the date string beforehand.

Date_PrevWorkDay behavior

The behavior of Date_PrevWorkDay has changed slightly.

The starting date is checked. If `$timecheck` was non-zero, the check failed if the date was not a business date, or if the time was not during business hours. If `$timecheck` was zero, the check failed if the date was not a business date, but the time was ignored.

In 5.xx, if the check failed, and `$timecheck` was non-zero, day 0 was defined as the start of the next business day, but if `$timecheck` was zero, day 0 was defined as the previous business day at the same time.

In 6.xx, if the check fails, and `$timecheck` is non-zero, the behavior is the same as before. If `$timecheck` is zero, day 0 is defined as the next business day at the same time.

So day 0 is now always the same, where before, day 0 meant two different things depending on whether `$timecheck` was zero or not.

(*) Default time

In Date::Manip 5.xx, the default times for dates was handled in an inconsistent manner. In the Date::Manip::Date documentation, if you parse a date from the ‘‘Common date formats’’ section, in Date::Manip 5.xx, if no time was included, it defaulted to ‘‘00:00:00’’. If you parsed a date from the ‘‘Less common formats’’ section, the default time was the current time.

So running a program on Jun 5, 2009 at noon that parsed the following dates gave the following return values:

```
Jun 12     =>  Jun 12, 2009 at 00:00:00
next week  =>  Jun 12, 2009 at 12:00:00
```

This behavior is changed and now relies on the config variable DefaultTime. If DefaultTime is ‘‘curr’’, the default time for any date which includes no information about the time is the current time. Otherwise, the default time is midnight.

%z format

In Date::Manip 5.xx, the `%z` format would give an offset in the form: −0500. Now it gives it in the form: −05:00:00

DELTAS

Dropped mixed style delta parsing

In Date::Manip 5.xx, a parsed delta could be written in the delta style

```
1:2:3
```

or in a language-specific expanded form:

```
1 hour 2 minutes 3 seconds
```

or in a mixed form:

```
1 hour 2:3
```

The mixed form has been dropped since I doubt that it sees much use in real life, and by dropping the mixed form, the parsing is much simpler.

Approximate date/date calculations

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). As of Date::Manip 6.00, 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 MUCH easier and faster to calculate.

Approximate relationships in deltas

When printing parts of deltas in Date::Manip::5.xx, the approximate relationship of 1 year = 365.25 days was used. This is the correct value for the Julian calendar, but for the Gregorian calendar, a better value is 365.2425, and this is used in version 6.00.

Old style formats

The formats used in the printf command are slightly different than in the old Delta_Format command.

The old formats are described in the Date::Manip::DM5 manual, and the new ones are in the Date::Manip::Delta manual.

The new formats are much more flexible and I encourage you to switch over, however at this point, the old style formats are officially supported for the Delta_Format command.

At some point, the old style formats may be deprecated (and removed at some point beyond that), but for now, they are not.

The old formats are NOT available using the printf method.

RECURRENCES

The day field meaning changed in a few recurrences

The value of the day field can refer to several different things including the day of week number (Monday=1 to Sunday=7), day of month (1−31), day of year (1−366), etc.

In Date::Manip 5.xx, it could also refer to the nth day of the week (i.e. 1 being the 1st day of the week, −1 being the last day of the week). This meaning is no longer used in 6.xx.

For example, the recurrence:

```
1*2:3:4:0:0:0
```

referred to the 3rd occurrence of the 4th day of the week in February.

The meaning has been changed to refer to the 3rd occurrence of day 4 (Thursday) in February. This is a much more useful type of recurrence.

As a result of this change, the related recurrence:

```
1*2:3:−1:0:0:0
```

is invalid. Negative numbers may be used to refer to the nth day of the week, but NOT when referring to the day of week numbers.

Recurrence range now inclusive

Previously, the list of dates implied by the recurrence were on or after the start date, but before the end date.

This has been changed so that the dates may be on or before the end date.

Dropped support for a couple English recurrences

Date::Manip 5.xx claimed support for a recurrence:

```
every 2nd day in June [1997]
```

In actuality, this recurrence is not practical to calculate. It requires a base date which might imply June 1,3,5,... in 1997 but June 2,4,6 in 1998.

In addition, the recurrence does not fit the mold for other recurrences that are an approximate distance apart. This type of recurrence has a number of closely spaced events with 11−month gaps between groups.

I no longer consider this a valid recurrence and support is now dropped for this string.

I also dropped the following for a similar reason:

```
every 6th Tuesday [in 1999]
```

Other minor recurrence changes

Previously, ParseRecur would supply default dates if the start or end were missing. This is no longer done.

DATE::MANIP FUNCTIONS

The Date::Manip module contains the same functions that Date::Manip 5.xx had (though the OO modules do all the work now). In general, the routines behave the same as before with the following exceptions:

Date_ConvTZ

Previously, Date_ConvTZ took 1 to 4 arguments and used the local time zone and the ConvTZ config variable to fill in missing arguments.

Now, the Date_ConvTZ function only supports a 3 argument call:

```
$date = Date_ConvTZ($date,$from,$to);
```

If `$from` is not given, it defaults to the local time zone. If `$to` is not given, it defaults to the local time zone.

The optional 4th argument ($errlevel) is no longer supported. If there is an error, an empty string is returned.

DateCalc

In Date::Manip 5.xx, it was recommended that you pass arguments to ParseDate or ParseDateDelta. This is not recommended with 6.00 since it is much more intelligent about handling the arguments, and you'll just end up parsing the date/delta twice.

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)