

datetime2 v1.5.3: date and time formats

Nicola L. C. Talbot

<http://www.dickimaw-books.com/>

2018-07-20

Abstract

The `datetime2` package replaces the `datetime` package. Languages and regional variations are dealt with by the `datetime2` language modules which are independently maintained and installed. Make sure that when you install `datetime2` you also install the required `datetime2` language modules.

Contents

1	Introduction	5
2	Example Usage	6
3	Displaying the Date and Time	10
4	Storing and Using Dates and Times	17
5	Styles	22
5.1	Predefined Styles	24
5.1.1	Full Styles	24
5.1.2	Time Styles	27
5.1.3	Zone Styles	27
5.2	Defining New Styles	28
6	Multi-Lingual Support	34
7	Standalone Month or Weekday Names	41
8	Package Options	44
9	The datetime2-calc Package	51
10	Migrating from datetime	59
10.1	datetime package options	60
10.2	Time and Date Commands	68
10.3	Saving Dates	80
10.4	Multilingual Support	84
10.5	Predefined Date Formats	89
10.6	Predefined Time Formats	93
10.7	Defining a New Date Format	94
10.8	Defining a New Time Format	102
11	The Code	109
11.1	datetime2.sty code	109
11.1.1	Defaults	115
11.1.2	Styles	124
11.1.3	Saving and Using Dates	149
11.1.4	Language Module Loading	155

11.2 datetime2-calc.sty code	159
11.2.1 Conversions and Calculations	160
11.2.2 Month and Weekday Names	164
Change History	174
Index	175

1 Introduction

I wrote the original `datetime` package back in the 1990s as an alternative to the `ukdate` package, which had dropped out of some of the TeX distributions, so it was designed specifically for UK date formats.¹ However some users found the time formats useful and the ability to save dates for later use, so when `babel` came along I had a number of requests to make `datetime` compatible with `babel` so that the regional date formats were preserved but the other `datetime` functions could be used. Then PDF \TeX came into existence and its `\pdfcreationdate` now provided a way of obtaining the seconds and time zone, which can't be obtained from TeX's `\time` primitive. Over time, the continual updates to the package has started to put a strain on the original naïve \TeX code of my first package, as it's been stretched well past its intended design.

The other problem with `datetime` is that some of the commands aren't expandable but some users want to be able to use them in expandable contexts (such as, for example, PDF bookmarks or writing a date stamp to an external file) or they want to be able to upper case the first letter if the date comes at the start of a sentence. This isn't an issue in English, as the weekday and month names are proper nouns and so automatically start with an upper case letter, regardless of where they appear in a sentence. Users who don't know the history and original purpose of the `datetime` package are puzzled as to why the defaults are all UK English or some styles were hard-coded, and some users are confused as to the ordering of the day, month and year parameters. In addition, some command names were incompatible with other date-related packages, but renaming those commands would break compatibility with older documents.

In order to address all these issues, a replacement package is necessary. Your old documents that use `datetime` should still be able to compile, but for your new documents, you may prefer the improved `datetime2` package instead.

¹Of course, `ukdatetime` would've been a better name, but the 8 dot 3 filename restriction was a concern back then.

2 Example Usage

```
\documentclass{article}
\usepackage{datetime2}
\begin{document}
This PDF was created on \today.
\end{document}
```

In the above example, the date is displayed in the form:

2015-03-01

This is the default style.

```
\documentclass{article}
\usepackage{datetime2}
\begin{document}
This PDF was created on \DTMnow.
\end{document}
```

In the above example, the full date, time and time zone is displayed in the form

2015-03-01 15:35:09Z

or

2015-04-01 08:55:39+01:00

unless you are using Xe_{La}TeX in which case the seconds and time zone are omitted. (Xe_{La}TeX doesn't provide this information, but as from v1.5.2, you can load `texosquery` before `datetime2` and enable the shell escape to obtain this information.) Alternatively you can hide the seconds and zone using the package options `showseconds=false` and `showzone=false`. If you want UTC+0 to be displayed numerically instead of using a `Z` you can use the `showisoZ=false` package option.

```
\documentclass[british]{article}
\usepackage{babel}
\usepackage{datetime2}
\begin{document}
This PDF was created on \DTMnow.
\end{document}
```

This has the same default numerical output as the previous example, but there are now two additional styles available: en-GB and en-GB-numeric. The `\datebritish` command provided by `babel` is redefined to prevent `babel` from overriding your preferred date style. The regional style can be enabled with the `useregional` option.

```
\documentclass[british]{article}
\usepackage{babel}
\usepackage[useregional]{datetime2}
\begin{document}
This PDF was created on \DTMnow.
\end{document}
```

The full date, time and zone are displayed in the form

1st March 2015 3:35pm GMT

or

1st April 2015 8:55am BST

```
\documentclass[british]{article}
\usepackage{babel}
\usepackage[useregional=numeric]{datetime2}
\begin{document}
This PDF was created on \DTMnow.
\end{document}
```

The full date, time and zone are displayed in the form

1/3/2015 15:35:09 GMT

or

1/4/2015 8:55:39 BST

```
\documentclass[english]{article}
\usepackage{babel}
\usepackage[useregional=numeric]{datetime2}
\begin{document}
This PDF was created on \DTMnow.
\end{document}
```

The full date, time and zone are displayed in the form

2015-03-01 15:35:09Z

This is because no *regional dialect* has been specified. The language name `english` is ambiguous, so the default style is used.

```
\documentclass[english]{article}
\usepackage{babel}
\usepackage[userregional]{datetime2}
\begin{document}
This PDF was created on \today.
\end{document}
```

The date is now displayed as

March 1, 2015

since that's \LaTeX 's default behaviour for `\today`. For other languages, you should check the language module documentation to find out what happens when the region can't be determined from the language name.

```
\documentclass[british]{article}
\usepackage{babel}
\usepackage[userregional,showdow]{datetime2}

\DTMLangsetup[en-GB]{abbr}

\begin{document}
This PDF was created on \today.
\end{document}
```

The date is now displayed as

Sun 1st Mar 2015

The options used in `\DTMLangsetup` (such as `abbr`) are provided by the language modules and should be described in the module's documentation. Different languages may have different options so `abbr` may not be available for some of them. The `showdow` (show day of week) option is a package-wide option even though it's a language setting. (This is because the `datetime2-calc` package is also needed if the day of the week should be displayed. As a package option, `showdow` can automatically load the required package.)

You need to check the documentation for the relevant language module or package to find out which styles check the `showdow` setting as not all of them do. For example, the `datetime2-english` module documentation indicates which of the English date styles support this setting. (The `datetime` package had a similar limitation with its `dayofweek` package option.)

If for some reason you can't load `babel` before `datetime2` and can only load it afterwards, then you need to explicitly load the module for each `babel` dialect with:

`\DTMusemodule`

```
\DTMusemodule{<language>}{<module-name>}
```

where *<language>* is the language (or dialect) name used with babel (for example, `british`) and *<module-name>* is the name of the `datetime2` module (for example, `en-GB`).

Example:

```
\documentclass{article}

\usepackage{datetime2}

\usepackage[british,irish]{babel}

\DTMusemodule{british}{en-GB}
\DTMusemodule{irish}{ga-IE}
```

The *<language>* argument should match the `\date<language>` command provided by babel. Similarly for polyglossia:

```
\documentclass{article}

\usepackage{datetime2}

\usepackage{polyglossia}
\setmainlanguage[variant=uk]{english}
\setotherlanguage{irish}

\DTMusemodule{english}{en-GB}
\DTMusemodule{irish}{ga-IE}
```

This ensures that not only is the required `datetime2` regional module loaded but also the date switching mechanism `\date<language>` is modified to prevent babel or polyglossia from interfering with `datetime2`.

Remember that you need to switch on the regional style, if required:

```
\DTMsetregional
```

or through the `useregional` package option.

3 Displaying the Date and Time

A specific date can be displayed using:

`\DTMdisplaydate`

```
\DTMdisplaydate{<year>}{<month>}{<day>}{<dow>}
```

The format used to display the date is governed by the *display style*.

The arguments are all *numerical*: *<year>* is the year, *<month>* is the month number (starting from 1 for January), *<day>* is the day of the month and *<dow>* is the day of the week number starting from 0 for Monday. The day of week number may be -1, which indicates that the style should ignore it. (Some styles always ignore the day of week, regardless of its value.) This command is intended for use in expandable contexts (such as writing the date to another file or using the date in the bookmarks) and is used by `\today`. The date styles must ensure that any fragile content is protected. (This is why *<dow>* isn't an optional argument otherwise the command wouldn't be expandable.)

The *<dow>* argument must always be an integer from -1 to 6. It should not be left blank or set to any other value. In some cases using an incorrect value may not cause a problem, but in other cases it will. So it's best to get into the habit of always setting it correctly.

If you want the *<dow>* value automatically calculated from the date, you can use `\DTMdate` (described below) instead with the `showdow` package option. Note that `\DTMdate` is *robust*. If you require an expandable alternative, the *<dow>* value must be calculated first. The simplest way to do this is to first save the date and then use it (see Section 4).

Examples (with the `showdow` package option set):

- Ignore day of week:

```
\section{\DTMdisplaydate{2016}{2}{10}{-1}}
```

This overrides the `showdow` option in this specific instance.

- Save the date first and then use it:

```
\DTMsavedate{mydate}{2016-02-10}  
\section{\DTMusedate{mydate}}
```

(See Section 4.)

- Another expandable alternative (but less convenient and more prone to error since the date has to be repeated):

```
\DTMcomputedayofweekindex{2016-02-10}{\dowindex}
\section{\DTMdisplaydate{2016}{2}{10}{\dowindex}}
```

(See Section 9.)

- Robust version won't work in PDF bookmarks or case-changing contexts (such as page headers):

```
\section{\DTMdate{2016-02-10}}
```

Note that if there is a table of contents, this will mean that the day of week index has to be calculated *twice*. Once in the table of contents and once in the actual section title. If the section title is also used in the page header, then the day of week index will additionally be calculated on every page that has this section title in the header.

(See the accompanying `datetime2-sample-journal.tex` sample file for more examples of using dates in section titles.)

Some styles may start the date with a word (such as the day of the week name or the month name). In English, proper nouns are capitalised regardless of where they appear in a sentence but some languages use lower case month or day of week names. In this event, if the initial letter needs to be capitalised then you can use:

`\DTMDisplaydate`

```
\DTMDisplaydate{<year>}{<month>}{<day>}{<dow>}
```

which is analogous to `\DTMdisplaydate`. Styles that are unaffected by this issue (for example, numerical or English dates) set `\DTMDisplaydate` to just `\DTMdisplaydate`. As with `\DTMdisplaydate` the style needs to ensure that any fragile content is protected in the event that `\DTMDisplaydate` is used in an expandable context. (Note that for this reason, I don't recommend the use of the commands provided by the `mfirstuc` package as they're not expandable.)

If you want the `<dow>` value automatically calculated from the date, you can use `\DTMDate` (described below) instead with the `showdow` package option.

The current date is displayed using

`\today`

```
\today
```

This uses `\DTMdisplaydate` to format the date so it will match the currently selected date style. There's also a first letter upper case version that uses `\DTMDisplaydate`:

`\Today`

```
\Today
```

Since there are other classes and packages that redefine `\today`, as from version 1.4, the `datetime2` package provides

```
\DTMtoday
```

```
\DTMtoday
```

and

```
\DTMToday
```

```
\DTMToday
```

The package now assigns `\today` and `\Today` to `\DTMtoday` and `\DTMToday`, respectively. If your document loads another package or class that modifies `\today` at the beginning of the document, you can switch it back to `datetime2`'s definition using

```
\let\today\DTMtoday
```

after the start of the document or use the `\AtBeginDocument` hook:

```
\AtBeginDocument{\let\today\DTMtoday}
```

If you use `babel` or `polyglossia` you must make sure you have the relevant `datetime2` language modules installed. (See Section 6.) You also need to make sure that `datetime2` is loaded *after* `babel`/`polyglossia` otherwise `\today` will be redefined so that it no longer uses `\DTMdisplaydate`.

As mentioned above, *some styles* allow the day of the week to be displayed. This requires the `datetime2-calc` package which will automatically be loaded if you set `showdow` in the `datetime2` package option list or if you set `showdow` in `\DTMsetup` *in the preamble*. The package option `calc` will also load `datetime2-calc` or you can load it explicitly using `\usepackage` after `datetime2` has been loaded. (You may use `showdow=true` in the document environment if the `datetime2-calc` package has been loaded in the preamble either explicitly or through the `calc` option.)

When `datetime2-calc` is loaded, it computes the current day of the week (using commands provided by the `pgfcalendar` package) which can then be used by `\today` or `\Today`. If `datetime2-calc` isn't loaded then neither `\today` nor `\Today` will display the day of the week, regardless of the current style.

If you would like a more convenient syntax and don't care about expansion, there is also a robust *non-expandable* command that can be used to display a particular date:

```
\DTMdate
```

```
\DTMdate{<date>}
```

As before there's also a capitalised version:

`\DTMDate`

```
\DTMDate{<date>}
```

In these cases the date should be provided as `<YYYY>-<MM>-<DD>` in the argument `<date>`. For example:

```
\DTMdate{2015-03-23}
```

Note that hyphens must always be used, regardless of the separator options. Take care that the category code of the hyphen hasn't changed when you use this syntax.

The year `<YYYY>` can't be negative¹ in `\DTMdate` or `\DTMDate`. Use `\DTMdisplaydate` or `\DTMDisplaydate` instead.

These commands internally use `\DTMdisplaydate` and `\DTMDisplaydate`, respectively. If the `datetime2-calc` package has been loaded, the day of the week will be computed, otherwise the day of the week will be set to -1. Another benefit of the `datetime2-calc` package is that it allows additional formats permitted by the `pgfcalendar` package:

- `<YYYY>-<MM>-last` (the last day of the given month).
- `<YYYY>-<MM>-<DD>+<n>` (`<n>` days before the given date).
- `<YYYY>-<MM>-<DD>+<n>` (`<n>` days after the given date).
- `<YYYY>-<MM>-last+<n>` (`<n>` days before the last day of the given month).
- `<YYYY>-<MM>-last+<n>` (`<n>` days after the last day of the given month).

See the `pgfcalendar` package for further details.

If you want to be able to use a date in an expandable context that can perform these calculations, consider first saving the date using one of the commands described in Section 4 and then use one of the expandable commands such as `\DTMuse` to display the date.

An error or unexpected results may occur if you try using one of these extended formats without loading the `datetime2-calc` package. An example that only works with `datetime2-calc`:

```
\DTMdate{2015-03-last}
```

An example that works with or without `datetime2-calc`:

```
\DTMdate{2015-03-31}
```

¹Well, actually it can if you put it in braces and don't use `datetime2-calc`.

In this second case, you'll only notice a difference in the output if the style should show the day of the week.

The style of the date is the same as for `\DTMdisplaydate` and `\DTMDisplaydate` (which `\DTMdate` and `\DTMDate` internally use, as mentioned above).

A time can be displayed using

`\DTMdisplaytime`

```
\DTMdisplaytime{<hour>}{<minute>}{<sec>}
```

where the arguments are all numerical (using 24 hours). The *time style* currently in effect determines how the time is formatted. The command is designed to be used in an expandable context so the styles should take care to protect any fragile commands.

Note that this command doesn't display the time zone. To display the time zone, you need to use

`\DTMdisplayzone`

```
\DTMdisplayzone{<TZh>}{<TZm>}
```

where `<TZh>` is the hour offset and `<TZm>` is the minute offset. The display is governed by the *zone style*. Again, the style should protect any fragile commands in case this is used in an expandable context.

The current time (as set at the start of the document build) can be displayed using

`\DTMcurrenttime`

```
\DTMcurrenttime
```

This internally just uses `\DTMdisplaytime` and so is designed for use in an expandable context.

The current zone can be displayed using

`\DTMcurrentzone`

```
\DTMcurrentzone
```

This internally just uses `\DTMdisplayzone` and so is designed for use in an expandable context.

If the PDF_T_EX primitive `\pdfcreationdate` is defined, the current time information is obtained from that, which includes the seconds and time zone. Lua_T_EX also defines this command (now replaced with `\pdffeedback_creationdate`) but X_Y_T_EX doesn't, and in that case the only way to determine the current time is from _T_EX's `\time` primitive which only contains the number of minutes since midnight, which means that the seconds and time zone are unavailable. Therefore if X_Y_T_EX is used, the `showseconds` and `showzone` options are automatically switched off.

New to v1.5.2: if `texosquery` is loaded before `datetime2` and the shell escape is enabled, then if `\pdfcreationdate` and `\pdffeedback` are undefined, the information will be obtained through `\TeXOSQueryNow`. See the `texosquery` documentation for further information about this command.

There is also a non-expandable robust command to display the time:

`\DTMtime`

```
\DTMtime{<tm>}
```

where `<tm>` must be in the 24 hour format `<hh>:<mm>:<ss>` (colon-separated numerical arguments). Take care if you use `babel` with a language setting that makes the colon character active. You will have to switch off the shorthands in order to use this command correctly.

The full date, time and zone (if available) can be displayed using

`\DTMdisplay`

```
\DTMdisplay{<year>}{<month>}{<day>}{<dow>}{<hh>}{<mm>}{<ss>}{<TZh>}{<TZm>}
```

The arguments are all numerical. The way the information is displayed in the document is governed by the *full style* (or *date-time style*). Typically the full style will redefine this command to use `\DTMdisplaydate`, `\DTMdisplaytime` and (optionally) `\DTMdisplayzone`. The `showzone` setting may govern whether or not to display the time zone (although a style may ignore this setting). The separators between the date and time and between the time and zone are governed by the style.

There is also an analogous version if capitalisation is required:

`\DTMDisplay`

```
\DTMDisplay{<year>}{<month>}{<day>}{<dow>}{<hh>}{<mm>}{<ss>}{<TZh>}{<TZm>}
```

Some styles may simply make this equivalent to `\DTMdisplay`. Other styles may use a similar format to `\DTMdisplay` but replace `\DTMdisplaydate` with `\DTMDisplaydate`.

The full current date, time and (optionally) zone can be displayed using:

`\DTMnow`

```
\DTMnow
```

which uses `\DTMdisplay` or

`\DTMNow`

`\DTMNow`

which uses `\DTMDisplay`.

4 Storing and Using Dates and Times

Date, time and zone information can be saved for later use. Note that the information is always saved numerically. The style is only applied when the information is later used. The commands that save the information are robust and not expandable. The commands that use the data are typically expandable although there may be some exceptions. Take care that the colon (:) and hyphen (-) characters haven't had their normal category code changed. (For example, through babel's shortcuts.) In the commands below, the *<name>* (no active characters) is a name that uniquely identifies the information.

Dates are saved using

`\DTMsavedate`

```
\DTMsavedate{<name>}{<date>}
```

where *<date>* is in the same format as for `\DTMdate`. As with `\DTMdate` (and `\DTMDate`) the format can be extended with the `datetime2-calc` package. If you want to access the day of week, you must make sure that `datetime2-calc` has been loaded *before you save the date*. (Remember that the `calc` and `showdoc` package options will automatically load `datetime2-calc`.) If `datetime2-calc` has been loaded, the day of week number will be calculated and saved. Whether or not it is displayed in the document when the date is later used depends on the settings when the date is displayed not when it's saved.

This command will override any previously defined date saved with this *<name>*. If a time or zone hasn't been defined with this *<name>*, the time and zone elements will all be set to 0 otherwise they will remain unchanged.

Note that you can't have a negative year in *<date>*. There's an alternative command you can use instead that doesn't try parsing *<date>*:

`\DTMsavenoparsedate`

```
\DTMsavenoparsedate{<name>}{<YYYY>}{<MM>}{<DD>}{<dow>}
```

The day of week *<dow>* may be -1 if unknown. This command doesn't calculate the day of week, even if `datetime2-calc` has been loaded.

Times are saved using

`\DTMsavetime`

```
\DTMsavetime{<name>}{<time>}
```

where the *<time>* is in the same format as for `\DTMtime`.

This command will override any previously defined time saved with this *<name>*. If a date or zone hasn't been defined with this *<name>*, the date and zone elements will all be set to 0 (or -1 for the day of week) otherwise they will remain unchanged.

Times and zone are saved using

`\DTMsavetimezn`

```
\DTMsavetimezn{<name>}{<time and zone>}
```

where the *<time and zone>* is in the form

<hh>:<mm>:<ss> <TZh>:<TZm>

(Note the space between the seconds and the hour offset.)

This command will override any previously defined time and zone saved with this *<name>*. If a date hasn't been defined with this *<name>*, the year, month and day will be set to zero and the day of the week to -1 otherwise they will remain unchanged.

All date, time and zone information can be saved at the same time using:

`\DTMsavetimestamp`

```
\DTMsavetimestamp{<name>}{<data>}
```

where *<data>* is in the format:

<YYYY>-<MM>-<DD>T<hh>:<mm>:<ss><zone>

The *<zone>* may either be Z or in the form *<TZh>:<TZm>* (for example, -03:00 or -3:0). This will override any date, time or zone data previously saved with this *<name>*.

Alternatively, if the date and time is in PDF form, that is

D: *<YYYY><MM><DD><hh><mm><ss><zone>*

where *<zone>* is either Z or *<TZh>'<TZm>'* then you can use

`\DTMsavefrompdfdata`

```
\DTMsavefrompdfdata{<name>}{<PDF data>}
```

Note that the category code of the initial D is irrelevant (in fact, the initial D is actually ignored) but the other characters should have their usual category code (so take care if, say, babel makes the colon an active character). The *<PDF data>* argument is automatically expanded so may be a control sequence that contains the PDF data. (Not the case with `\DTMsavetimestamp`.)

The current date and time can be saved using:

`\DTMsavenow`

```
\DTMsavenow{<name>}
```

There is also a command that can be used to save the modification date of a file, but it's not available for some T_EX engines:

```
\DTMsavefilemoddate
```

```
\DTMsavefilemoddate{<name>}{<file name>}
```

where *<file name>* is the name of the file (remember to use forward slashes / for the directory divider). If you build your document using PDF_TE_X, this command will use the PDF_TE_X primitive `\pdffilemoddate`. If you use Lua_TE_X this command will attempt to use `os.date` but it uses `%z` for the time zone, which may not work on some operating systems. If you use X_YL_AT_EX this command will generate a warning and will assume a date of 0000-00-00T00:00:00Z unless (from v1.5.2) you have loaded `texosquery` before `datetime2` and you have the shell escape enabled, in which case `\DTMsavefilemoddate` will use `\TeXOSQueryFileDate` to obtain the information. See the `texosquery` manual for further details about this command.

The above commands are all localised to the current scope. If the data is required after the end of the scope, you can make the assignments global using:

```
\DTMmakeglobal
```

```
\DTMmakeglobal{<name>}
```

For example:

```
\DTMsavenow{mydate}\DTMmakeglobal{mydate}
```

A previously saved date can be displayed using the current style with

```
\DTMusedate
```

```
\DTMusedate{<name>}
```

This just uses `\DTMdisplaydate`. An error will occur if *<name>* hasn't been defined. Alternatively for the capitalised version:

```
\DTMUusedate
```

```
\DTMUusedate{<name>}
```

which uses `\DTMDisplaydate` instead.

A previously saved time can be displayed using the current style with

```
\DTMusetime
```

```
\DTMusetime{<name>}
```

This just uses `\DTMdisplaytime`. An error will occur if *<name>* hasn't been defined.

A previously saved zone can be displayed using the current style with

`\DTMusezone`

```
\DTMusezone{<name>}
```

This just uses `\DTMdisplayzone`. An error will occur if `<name>` hasn't been defined.
The entire date, time and zone can be displayed in the current style with

`\DTMuse`

```
\DTMuse{<name>}
```

This uses `\DTMdisplay`. An error will occur if `<name>` hasn't been defined. Alternatively,

`\DTMUse`

```
\DTMUse{<name>}
```

will use `\DTMDisplay` instead.

You can determine if a given `<name>` has been defined using

`\DTMifsaveddate`

```
\DTMifsaveddate{<name>}{<true>}{<false>}
```

The individual numerical elements can be fetched using one of the following commands. These don't check if the given data identified by `<name>` has been defined and will expand to `\relax` if the name isn't recognised.

`\DTMfetchyear`

```
\DTMfetchyear{<name>}
```

This expands to the year.

`\DTMfetchmonth`

```
\DTMfetchmonth{<name>}
```

This expands to the month number.

`\DTMfetchday`

```
\DTMfetchday{<name>}
```

This expands to the day of the month.

`\DTMfetchdow`

```
\DTMfetchdow{<name>}
```

This expands to the day of the week number (-1 if unknown).

`\DTMfetchhour`

```
\DTMfetchhour{<name>}
```

This expands to the hour.

```
\DTMfetchminute
```

```
\DTMfetchminute{<name>}
```

This expands to the minute.

```
\DTMfetchsecond
```

```
\DTMfetchsecond{<name>}
```

This expands to the second.

```
\DTMfetchTZhour
```

```
\DTMfetchTZhour{<name>}
```

This expands to the hour offset.

```
\DTMfetchTZminute
```

```
\DTMfetchTZminute{<name>}
```

This expands to the minute offset.

5 Styles

If you want to just change the date style use:

`\DTMsetdatestyle`

```
\DTMsetdatestyle{<name>}
```

where *<name>* identifies the style. For example:

```
\DTMsetdatestyle{iso}
```

This will just change the date style (`\DTMdisplaydate` and `\DTMdisplaydate`), not the time or zone styles. Note that `\DTMdisplay` typically uses `\DTMdisplaydate` so this will also change the date element of `\DTMdisplay`.

If you want to just change the time style use:

`\DTMsettimestyle`

```
\DTMsettimestyle{<name>}
```

where *<name>* identifies the style. For example:

```
\DTMsettimestyle{iso}
```

This will just change the time style (`\DTMdisplaytime`), not the date or zone styles. Note that `\DTMdisplay` typically uses `\DTMdisplaytime` so this will also change the time element of `\DTMdisplay`.

If you want to just change the zone style use:

`\DTMsetzonestyle`

```
\DTMsetzonestyle{<name>}
```

where *<name>* identifies the style. For example:

```
\DTMsetzonestyle{iso}
```

This will just change the zone style (`\DTMdisplayzone`), not the date or time styles. Note that `\DTMdisplay` typically uses `\DTMdisplayzone` so this will also change the zone element of `\DTMdisplay`.

If you want to change the full style use:

`\DTMsetstyle`

```
\DTMsetstyle{<name>}
```

where *<name>* identifies the style. For example:

```
\DTMsetstyle{iso}
```

Note that in this case this does more than simply

```
\DTMsetdatestyle{iso}\DTMsettimestyle{iso}\DTMsetzonestyle{iso}
```

as it also changes `\DTMdisplay` and `\DTMDisplay`. If the style *<name>* is only a partial style, a warning will be issued for any partial styles that aren't defined for the given name as well as a warning for the undefined full style. An error will occur if there are neither partial nor full styles with the given *<name>*.

The predefined styles listed in Section 5.1.1 are all *full styles*. This means that they change the date, time, zone and full format, so any of them can be used in `\DTMsetdatestyle`, `\DTMsettimestyle`, `\DTMsetzonestyle` or `\DTMsetstyle`. However it's possible for a style to be only a *partial style*, such as those described in Section 5.1.2.

For example, if `foo` is a date style and a time style but isn't a zone style or a full style then you can use

```
\DTMsetdatestyle{foo}
```

and

```
\DTMsettimestyle{foo}
```

but you can't use `\DTMsetzonestyle`. You can use

```
\DTMsetstyle{foo}
```

but this will now only be equivalent to

```
\DTMsetdatestyle{foo}\DTMsettimestyle{foo}
```

and while `\DTMdisplay` and `\DTMDisplay` will typically use these date and time settings, the way that the date, time and zone are arranged will be governed by the full style setting that was already in effect before the date and time style changed.

The style changes are all local and so are affected by the current scope.

In addition to the predefined styles, there are also styles provided by regional modules. These may or may not be set depending on the setting of `useregional`. Those that are provided usually follow the naming scheme *<lang name>* or *<lang code>-<country code>*, where *<lang name>* is the root language name (for example, `english`), *<lang code>* is the ISO language code and *<country code>* is the ISO country code (for example, `en-GB`). If a numeric style is provided, it's usually the name of the text style with `-numeric` appended (for example, `en-GB-numeric`). If you're not sure of the exact naming scheme you can use

```
\DTMtryregional
```

```
\DTMtryregional[<lang name>]{<lang code>}{<country code>}
```

This takes into account the `useregional` option, so will do nothing if `useregional=false`. If the optional argument is omitted, the root language name will be determined from the supplied ISO language code if it's recognised.

If there's no match, no change will be made. There will be no warnings or error messages.

For example:

```
\DTMsetup{useregional=numeric}  
\DTMtryregional{en}{GB}
```

This will set the style `en-GB-numeric` (assuming the `en-GB` module has been loaded).

In this example:

```
\DTMsetup{useregional}  
\DTMtryregional{nl}{BE}
```

the style is set to `dutch` (assuming the `dutch` module has been loaded), since there's no style called `nl-BE`.

If the ISO codes are stored in control sequences which may or may not be defined, you can use the starred version which expects commands for the last two arguments:

```
\DTMtryregional *
```

```
\DTMtryregional*[\langle lang name \rangle]{\langle lang code cs \rangle}{\langle country code cs \rangle}
```

5.1 Predefined Styles

The base `datetime2` package provides a number of predefined numerical styles. Section 5.1.1 lists the full styles, which can be used with `\DTMsetstyle`, `\DTMsetdatestyle`, `\DTMsettimestyle` and `\DTMsetzonestyle`. Section 5.1.2 lists the predefined (partial) times styles, which can be used with `\DTMsettimestyle` and `\DTMsetstyle`.

5.1.1 Full Styles

The following are predefined full styles that are provided by the base `datetime2` package. Additional styles are available through the language modules (see Section 6).

`default` The `default` style displays the date in the form

$$\langle YYYY \rangle \langle YMsep \rangle \langle MM \rangle \langle MDsep \rangle \langle DD \rangle$$

where the month $\langle MM \rangle$ and day of the month $\langle DD \rangle$ numbers are formatted as two digits. The separators $\langle YMsep \rangle$ and $\langle MDsep \rangle$ default to a hyphen but can be changed

using the options `yearmonthsep`, `monthdaysep` or `datesep` (either through the package options or using `\DTMsetup`).

The time is displayed in the form:

$$\langle hh \rangle \langle HMsep \rangle \langle mm \rangle \langle MSsep \rangle \langle ss \rangle$$

where the hour, month and seconds are formatted as two digits. The final $\langle MSsep \rangle \langle ss \rangle$ is omitted if the option `showseconds` has been set to `false`. The separators $\langle HMsep \rangle$ and $\langle MSsep \rangle$ default to a colon (`:`) but these may be changed using the options `hourminsep`, `minsecsep` or `datetimesep`.

The zone is displayed in form

$$\langle TZh \rangle \langle HMsep \rangle \langle TZm \rangle$$

or just `Z` if the option `showisoZ` is set to `true` and both $\langle TZh \rangle$ and $\langle TZm \rangle$ are zero. The separator $\langle HMsep \rangle$ is the same as used for the time format. The final $\langle HMsep \rangle \langle TZm \rangle$ is omitted if the option `showzoneminutes` is set to `false`. The hour offset $\langle TZh \rangle$ is formatted as two digits preceded by either `+` or `-` and the minute offset is formatted as two digits. Note that since one of the main purposes of this package is to provide expandable date commands that can be used to write information to external files, no attempt is made to convert the hyphen `-` (for negative offsets) into a minus sign. If you want it rendered correctly in your document, consider placing the time zone command in math mode and adjust the separators as necessary.

The full style is in the form

$$\langle date \rangle \langle DTsep \rangle \langle time \rangle \langle TZsep \rangle \langle zone \rangle$$

The $\langle date \rangle \langle DTsep \rangle$ part is omitted if the option `showdate` is set to `false`, and the $\langle TZsep \rangle \langle zone \rangle$ part is omitted if the option `showzone` is set to `false`. The separator between the date and time $\langle DTsep \rangle$ defaults to `\space` but may be changed using the `datetimesep` option. The separator between the time and zone $\langle TZsep \rangle$ defaults to nothing but may be changed using the `timezonesep` option.

`iso` The `iso` style is like the default style but the separators can't be changed. The separators used in the date format are fixed as hyphens and the separators used in the time and zone formats are fixed as colons. In the full format, the separator between the date and time is fixed as `T` and there's no separator between the time and zone. The only options that can change the `iso` style are `showseconds`, `showdate`, `showzone`, `showzoneminutes` and `showisoZ`.

yyyymd This is like the default style except that the month and date aren't forced into a two-digit format.

ddmmyyy This is like the default style except that the date is formatted in the reverse order

$\langle DD \rangle \langle MDsep \rangle \langle MM \rangle \langle YMsep \rangle \langle YYYY \rangle$

The day and month are displayed as two-digits and the separators are as for the default style. The options that modify the default style similarly modify this style.

dmyyy This is like the ddmmyyy style except that it doesn't force the day and month into a two-digit format. The options that modify the default style similarly modify this style.

dmyy This is like the dmyy style except that it only displays the final two digits of the year. The options that modify the default style similarly modify this style.

ddmmyy This is like the default style except that the date is formatted in the reverse order

$\langle DD \rangle \langle MDsep \rangle \langle MM \rangle \langle YMsep \rangle \langle YY \rangle$

The day, month and year are displayed as two-digits and the separators are as for the default style. The options that modify the default style similarly modify this style.

mmddyy This is like the ddmmyyy style except the day and month numbers are reversed. The separator between the month and day is still given by the monthdaysep or datesep options. The separator between the day and year is given by the dayyearsep or datesep options.

mmddy This is like the ddmmyy style except the day and month numbers are reversed. The separator between the month and day is still given by the monthdaysep or datesep options. The separator between the day and year is given by the dayyearsep or datesep options.

mdyy This is like the mmddyy style except that it doesn't force the day and month into a two-digit format.

mdyy This is like the mdyy style except that the year only has the final two digits displayed.

pdf This formats the date, time and zone so that the full style is in the form required by the date settings in \pdfinfo. The date format is

$D:\langle YYYY \rangle \langle MM \rangle \langle DD \rangle$

where the month and day numbers are displayed as two digits.

The time format is

`<hh><mm><ss>`

where the numbers are displayed as two digits.

The zone format is

`<hh>'<mm>'`

or Z for zero time offset if the option `showisoZ` is used. (The `showisoZ` option is the only option that modifies the `pdf` style.) The hour and minutes are displayed as two digits where the hour has the sign present (either + or -).

The full style is a concatenation of the date, time and zone.

`D:<YYYY><MM><DD><hh><mm><ss><hh>'<mm>'`

5.1.2 Time Styles

There's only one predefined time (partial) style provided by the base `datetime2` package. This style can be used to override the time format part of full styles. For example, to use the default full style with the `hmmss` time style:

```
\DTMsetstyle{default}\DTMsettimestyle{hmmss}
```

`hmmss` The `hmmss` style is like the time style provided by the full default style except that the hour isn't forced into two digits.

5.1.3 Zone Styles

The following are predefined zone (partial) styles that are provided by the base `datetime2` package. These styles can be used to override the zone format part of full styles. For example, to use the default full style with the `map` zone style:

```
\DTMsetstyle{default}\DTMsetzonestyle{map}
```

`map` The `map` style uses `\DTMusezonemapordefault` to display the mapping, if one exists, or use the default style, if a mapping doesn't exist. For example:

```
\DTMNatoZoneMaps  
\DTMsetzonestyle{map}
```

This first defines the NATO mappings and then switches to the map style.

hhmm The hhmm style displays the time zone in the form

$\langle TZh \rangle \langle HMsep \rangle \langle TZm \rangle$

where $\langle HMsep \rangle$ is given by the hourminsep option. This style honours the showzone-minutes option but ignores the showisoZ option. The hour is always prefixed by the sign.

5.2 Defining New Styles

A new date style can be defined using:

`\DTMnewdatestyle`

```
\DTMnewdatestyle{\langle name \rangle}{\langle definition \rangle}
```

This defines a partial style that should only modify `\DTMdisplaydate` and `\DTMDisplaydate`. The redefinition of these commands should be placed in $\langle definition \rangle$.

A new time style can be defined using:

`\DTMnewtimestyle`

```
\DTMnewtimestyle{\langle name \rangle}{\langle definition \rangle}
```

This defines a partial style that should only modify `\DTMdisplaytime`. The redefinition should be placed in $\langle definition \rangle$.

A new zone style can be defined using:

`\DTMnewzonestyle`

```
\DTMnewzonestyle{\langle name \rangle}{\langle definition \rangle}
```

This defines a partial style that should only modify `\DTMdisplayzone`. The redefinition should be placed in $\langle definition \rangle$.

A new full style can be defined using:

`\DTMnewstyle`

```
\DTMnewstyle{\langle name \rangle}{\langle date style definition \rangle}{\langle time style definition \rangle}
{\langle zone style definition \rangle}{\langle full style definition \rangle}
```

This does

```
\DTMnewdatestyle{\langle name \rangle}{\langle date style definition \rangle}
```

```
\DTMnewtimestyle{<name>}{<time style definition>}
```

```
\DTMnewzonestyle{<name>}{<zone style definition>}
```

and finally *<full style definition>* should redefine `\DTMdisplay` and `\DTMDisplay`.

Remember to use a double-hash to reference the parameters (`##1`, `##2` etc) within *<definition>* in all the above. In each case *<name>* is the label identifying the style and shouldn't contain active characters.

As from version 1.2, you can redefine existing styles with the following commands.

```
\DTMrenewdatestyle
```

```
\DTMrenewdatestyle{<name>}{<definition>}
```

This redefines the named date style. The original may be either a partial or a full style.

```
\DTMrenewtimestyle
```

```
\DTMrenewtimestyle{<name>}{<definition>}
```

This redefines the named time style. The original may be either a partial or a full style.

```
\DTMrenewzonestyle
```

```
\DTMrenewzonestyle{<name>}{<definition>}
```

This redefines the named time zone style. The original may be either a partial or a full style.

```
\DTMrenewstyle
```

```
\DTMrenewstyle{<name>}{<definition>}
```

This redefines the named full style. The original style must also be a full style.

There are also commands analogous to `\providecommand` that will define styles that don't already exist.

```
\DTMprovidedatestyle
```

```
\DTMprovidedatestyle{<name>}{<definition>}
```

This defines the named date style. This won't do anything if either a partial date style or a full style with the given name already exists.

```
\DTMprovidetimestyle
```

```
\DTMprovidetimestyle{<name>}{<definition>}
```

This defines the named time style. This won't do anything if either a partial time style or a full style with the given name already exists.

`\DTMprovidezonestyle`

```
\DTMprovidezonestyle{<name>}{<definition>}
```

This defines the named zone style. This won't do anything if either a partial zone style or a full style with the given name already exists.

`\DTMprovidestyle`

```
\DTMprovidestyle{<name>}{<definition>}
```

This defines the named full style if the named full style doesn't already exist. This internally uses the previous three commands for the partial elements of the full style, so if a partial style with this name already exists, it won't be changed.

As from v1.5.2, you can determine if a style exists using:

`\DTMifhasstyle`

```
\DTMifhasstyle{<name>}{<true>}{<false>}
```

If the (full) style given by `<name>` exists, this does `<true>` otherwise it does `<false>`.

Similarly for a partial date style

`\DTMifhasdatestyle`

```
\DTMifhasdatestyle{<name>}{<true>}{<false>}
```

partial time style

`\DTMifhastimestyle`

```
\DTMifhastimestyle{<name>}{<true>}{<false>}
```

and partial time zone style

`\DTMifhaszonestyle`

```
\DTMifhaszonestyle{<name>}{<true>}{<false>}
```

There are some helper commands provided that you might want to use in the style definitions.

`\DTMtwodigits`

```
\DTMtwodigits{<number>}
```

This displays `<number>` so that it has *exactly* two digits. Unlike `\two@digits` this will check for a negative number and will trim a number whose absolute value is greater than or equal to 100. This command is expandable.

`\DTMcentury`

```
\DTMcentury{<year>}
```

This converts `<year>` to the century. If `<year>` is negative it does:

```
-\DTMcentury{-<year>}
```

Example:

```
\DTMcentury{1945}
```

expands to 20 (not 19). Note that

```
\DTMcentury{1900}
```

expands to 19.

`\DTMdivhundred`

```
\DTMdivhundred{<number>}
```

This expands to $\lfloor \langle number \rangle / 100 \rfloor$ (integer division by 100 rounded down). For example:

```
\DTMdivhundred{1945}
```

expands to 19 and

```
\DTMdivhundred{1900}
```

expands to 19.

`\DTMtexorpdfstring`

```
\DTMtexorpdfstring{<TeX>}{<PDF>}
```

If `hyperref` is loaded, this is equivalent to `\texorpdfstring` otherwise it just does the first argument and ignores the second. (The check for `hyperref` is deferred until the start of the document environment, so it doesn't matter if `hyperref` is loaded after `datetime2`.) This command may be used to provide alternative text to use if the date/time/zone is displayed in the PDF bookmarks.

`\DTMsep`

```
\DTMsep{<tag>}
```

This accesses the value of the `<tag>sep` base package option. (Not the language module options.) For example

```
\DTMsep{yearmonth}
```

expands to the value supplied by the yearmonthsep package option.

`\DTMusezonemap`

```
\DTMusezonemap{<TZh>}{<TZm>}
```

This expands to the time zone abbreviation or `\relax` if no mapping has been set for the given time zone.

You can define a time zone mapping using

`\DTMdefzonemap`

```
\DTMdefzonemap{<TZh>}{<TZm>}{<abbr>}
```

For example

```
\DTMdefzonemap{00}{00}{GMT}  
\DTMdefzonemap{01}{00}{BST}
```

Note that `datetime2` doesn't know anything about daylight saving, so this is only really designed for dates and times in a specific location. This overwrites any previous mapping for this time zone.

The base `datetime2` package provides

`\DTMNatoZoneMaps`

```
\DTMNatoZoneMaps
```

This defines the military/NATO mappings from A (Alpha time) to Z (Zulu time). You can use this command if you want these time zones (but remember to set an appropriate time zone style that uses the zone mappings).

The language modules may provide mappings that are enabled when you switch to that style. For example, the `en-GB` language module provides the `mapzone` option which, if set to `true`, will map `+00:00` to GMT and `+01:00` to BST. See the documentation for the language module for further details.

`\DTMclearmap`

```
\DTMclearmap{<TZh>}{<TZm>}
```

Clears the time zone mapping. The regional time zone styles should use

`\DTMresetzones`

```
\DTMresetzones
```

before applying any regional mappings. This defaults to nothing which means that any mappings previously defined by other styles won't be cleared. You can redefine this command if you want to clear any mappings that aren't relevant for other regions.

You can test if a mapping is defined using

`\DTMhaszonemap`

```
\DTMhaszonemap{<TZh>}{<TZm>}{<true>}{<false>}
```

This will do *<true>* if there is a mapping defined for that time zone or *<false>* otherwise.

`\DTMusezonemapordefault`

```
\DTMusezonemapordefault{<TZh>}{<TZm>}
```

This will use the mapping if its defined otherwise it will expand to the format *<TZh><HMsep><TZm>* where *<HMsep><TZm>* is omitted if the option `showzoneminutes` is set to `false`. The separator *<HMsep>* is as given by the `hourminsep` option. (The `showisoZ` option isn't used here so UTC+00:00 will be displayed as +00:00 or +00 if there's no mapping.)

Here's an example of a simple date style that just displays the year and month as two digits but uses the `yearmonthsep` option:

```
\newdatestyle
{mmyy}% label
{% definitions
  \renewcommand*{\DTMdisplaydate}[4]{%
    \DTMtwodigits{##2}\DTMsep{yearmonth}\DTMtwodigits{##1}}%
  \renewcommand*{\DTMdisplaydate}{\DTMdisplaydate}%
}
```

If you want to distribute your new styles, just put the definitions in a package and upload it to CTAN. For example (replace `mystylename` with something more appropriate, and also change the date in the `\ProvidesPackage` line):

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{mystylename}[2014/03/24 v1.0]
\RequirePackage{datetime2}

% style definitions here

\endinput
```

Save the file as `mystylename.sty`, add some documentation about the style (or styles) provided and read the instructions at <http://www.ctan.org/upload> and <http://www.ctan.org/file/help/ctan/CTAN-upload-addendum>. The upload location for additions to the `datetime2` package (either for packages defining new styles or for language modules) should be `/macros/latex/contrib/datetime2-contrib/mystylename` (remember to replace `mystylename` as appropriate).

6 Multi-Lingual Support

If you want to use `datetime2` with `babel` or `polyglossia`, make sure you load `babel/polyglossia` *before* you load `datetime2` otherwise their `\date<language>` will overwrite `\datetime2`'s definition of `\today`. *Additionally* you need to make sure you install the relevant `datetime2` language modules. These modules are automatically loaded, if required, by `datetime2` but only if they are already installed. Remember that if you use \LaTeX you won't have the seconds or time zone available for the current date and time (unless you first load `texosquery` and have the shell escape enabled).

If the required language modules aren't installed or `datetime2` is loaded before `babel/polyglossia` then `datetime2`'s definition of `\today` will be overridden and may no longer match the currently selected date style.

Each language module defines a textual style (where the month is displayed as a word) for that language or region which can be used in the argument of `\DTMsetstyle`, `\DTMsetdatestyle`, `\DTMsettimestyle` or `\DTMsetzonestyle`. The language module may also define a numeric style. In the ambiguous cases where the language name alone doesn't indicate the region (for example, `english` instead of `UKenglish` or `USenglish`) the module should use the default numeric style (see Section 5.1.1).

The textual style provided by the module will automatically be set using `\DTMsetstyle` if the *useregional* option is set to *text*. By default `useregional` is `false`, unless the language/region is passed via the `datetime2` package option list. (The `useregional` option is unaffected if the setting is passed through the document class option list.) The numeric style provided by the module will automatically be set if the `useregional` option is set to `numeric`. See the descriptions for the `useregional` and `style` options in Section 8.

Be careful not to mix the language/region options between the document class option list and the `babel/polyglossia` interface. For example, don't do:

```
\documentclass[en-GB]{article}
\usepackage[canadien,british]{babel}
```

The above example will prevent the `tracklang` package from picking up the `babel` setting and it will only detect the `en-GB` option. Use only the document class options or only the `babel` package option list or duplicate *all* the `babel` package options with analogous `tracklang` options in the document class. For example

```
\documentclass[canadien,british]{article}
\usepackage{babel}
```

or

```
\documentclass{article}
\usepackage[canadien,british]{babel}
```

or

```
\documentclass[fr-CA,en-GB]{article}
\usepackage[canadien,british]{babel}
```

Language modules may be used without babel or polyglossia. For example:

```
\documentclass{article}
\usepackage[en-GB]{datetime2}
\begin{document}
\today
\end{document}
```

If you have more than one language or region you will need to switch styles using `\DTMsetstyle` etc if you aren't using babel or polyglossia:

```
\documentclass{article}
\usepackage[en-GB,en-CA]{datetime2}
\begin{document}
\DTMsetstyle{en-GB}\today.
\DTMsetstyle{en-CA}\today
\end{document}
```

If you want to change the number separators for the *regional* numeric styles, you need to use `\DTMlangsetup`. If you want to change the number separators for the base `datetime2` predefined numeric styles (see Section 5.1) then you need to use `\DTMsetup` or the package options. You therefore need to use `\DTMsetup` for the ambiguous regionless language numeric settings since they just use the default style. Check the module documentation to find out if the default style is used.

Examples of use:

1. Language option specified through the document class and picked up by `tracklang` (which is loaded by `datetime2`). This setting is also picked up by `babel` which is loaded before `datetime2`.

```
\documentclass[british]{article}

\usepackage{babel}
\usepackage{datetime2}

\begin{document}
\today
\end{document}
```

The date is displayed in the default format 2015-03-01.

In this case, the en-GB language module is loaded which defines the text style en-GB and the numeric style en-GB-numeric. Since useregional hasn't been set, \today uses datetime2's default numerical format. If babel was loaded after datetime2, the babel's hook management system would overwrite datetime2's definition of \today so that it no longer used \DTMdisplaydate. A similar result is obtained if in the above example babel is replaced with polyglossia (where the language is set in the document class option).

You can change the useregional setting either through datetime2's package options or using \DTMsetup however it will only have an effect during the module loading (when the value is changed via the package option) and when \date<language> is used. An alternative is to use \DTMsetregional which will also do \date<language> if it is available (where <language> is the current value of \language) otherwise it will iterate through the list of known dialects and try to set each one in turn.

This means that using \DTMsetregional may cause the style to change to a different region if you have multiple regions defined.

For example, in the document below, the date is displayed using the default numeric format because useregional has been changed *after* babel uses \datebritish to set the language at the start of the document.

```
\documentclass[british]{article}

\usepackage{babel}
\usepackage{datetime2}

\begin{document}
\DTMsetup{useregional}
\today
\end{document}
```

So here \today again displays the date in the form 2015-03-01.

If the setting is moved to the preamble:

```
\documentclass[british]{article}

\usepackage{babel}
\usepackage{datetime2}

\DTMsetup{useregional}
\begin{document}
\today
\end{document}
```

then the `useregional` setting is checked at the beginning of the document when `babel` uses `\datebritish`. So in this case `\today` will display the date in the form 1st March 2015.

2. Language setting specified through `babel`'s package option list:

```
\documentclass{article}

\usepackage[british]{babel}
\usepackage{datetime2}

\begin{document}
\today
\end{document}
```

This has the same result as placing `british` in the document class option list, so the date is again displayed in the default format 2015-03-01 but, as in the previous example, the `en-GB` and `en-GB-numeric` styles are both defined if required.

However a problem occurs if `babel` is replaced by `polyglossia`:

```
\documentclass{article}

\usepackage{fontspec}
\usepackage{polyglossia}
\setdefaultlanguage[variant=uk]{english}

\usepackage{datetime2}

\begin{document}
\today
\end{document}
```

In this case `tracklang` is unable to pick up the variant and can only detect the root language, so it will load the generic `english` module instead of the `en-GB` module. This means that the `en-GB` and `en-GB-numeric` styles are no longer available. However, since `useregional` is `false` the date is still displayed using the default numeric style in the form 2015-03-01.

3. As mentioned above neither `babel` nor `polyglossia` are required in order to use the `datetime2` language modules. You can simply supply the language setting in the package option list:

```
\documentclass{article}

\usepackage[british]{datetime2}

\begin{document}
\today
\end{document}
```

This additionally sets `useregional=true` (since the language is in the package option list not the document class option list) so the date produced by `\today` now uses the `en-GB` date style in the form 1st March 2015.

4. The regional numeric format can be used instead if `useregional` is set to `numeric`:

```
\documentclass{article}

\usepackage[british,useregional=numeric]{datetime2}

\begin{document}
\today
\end{document}
```

This now displays the date in the form 1/3/2015.

Many of the language options have synonyms. In addition to the `babel` synonyms (such as `british` or `UKenglish`) the `tracklang` package provides options in ISO form, such as `en-GB`. Note that the style name provided by each language module is independent of the package option used to select that style. So regardless of whether you use `british`, `UKenglish` or `en-GB`, the text style name is `en-GB` and the numeric style name is `en-GB-numeric`. If just `english` is used, the text style name is `english` but the numeric style is `default`.

Languages where the region is automatically implied, such as `scottish`, provide a text style with the root language name (`scottish` in this instance) and a numeric style in the form `<language>-numeric` (such as `scottish-numeric`). Note that the `irish` module has regionless styles `irish` and `irish-numeric` but also has regional styles `ga-IE` and `ga-IE-numeric` (for the Republic of Ireland) and `ga-GB` and `ga-GB-numeric` (for Northern Ireland). In this case the regionless style has a numeric style instead of using the `default` style since both `ga-IE-numeric` and `ga-GB-numeric` are the same so there's no ambiguity. The only difference in the three modules `datetime2-irish`, `datetime2-ga-IE` and `datetime2-ga-GB` is the time zone mappings.

The language or regional modules may provide additional settings that can be applied using

`\DTMLangsetup`

```
\DTMLangsetup[<module-name list>]{<options>}
```

where `<module-name list>` is a comma-separated list of modules that have previously been loaded (such as `en-GB`, `en-US`) and `<options>` is a `<key>=<value>` list of options.

Note that the names in the `<module-name list>` are the identifying names of the module (such as `en-GB` or `english`) which aren't necessarily the same as the language name supplied to whatever language package you are using (such as `babel` or `polyglossia`). The code for each module should be in the file `datetime2-<module-name>.ldf`, which should be in TeX's path.

If *<module-name list>* is omitted, then the list of all loaded modules is assumed. There is also a starred version of this command (as from v1.3) which suppresses the warning if the given *<options>* aren't available for any of the modules named in *<module-name list>*. You may prefer to use the starred version if you omit *<module-name list>* to skip the warnings from the base modules that don't support the given options.

The modules may also provide user commands to further customise the style. These settings should all be described in the module's documentation, which should be accessible via `texdoc datetime2-<language>` where *<language>* is the root language name in lower case (such as `english`).

Note that although I maintain the `datetime2` English language module, I don't maintain the other modules. If you have an issue with one of the other modules, please contact the module maintainer. If there is no maintainer, feel free to volunteer to take over the maintenance ([send me a message](#)). If there's no module for your language you can create your own module and upload it to CTAN in the `/macros/latex/contrib/datetime2-contrib/datetime2-<language>` directory.

You can use the English or Irish modules as a template for a language with multiple regions. Just download the English source files `datetime2-english.dtx` and `datetime2-english.ins` or the Irish source files `datetime2-irish.dtx` and `datetime2-irish.ins` from CTAN and make the appropriate modifications. Alternatively you can use the Scottish module as a template for a single-region language. Just download the Scottish source files `datetime2-scottish.dtx` and `datetime2-scottish.ins` from CTAN and make the appropriate modifications. (Don't forget to provide a README file.)

Each language module should be in a file named `datetime2-<lang>.ldf` where *<lang>* is either the language name or in the form *<language ISO code>-<country ISO code>*. (See the `tracklang` documentation for further details of the naming scheme.)

A regional module may load a base module for the same language using

`\RequireDateTimeModule`

```
\RequireDateTimeModule{<name>}
```

This will input the file `datetime2-<name>.ldf`. This command should not be used outside the `datetime2` language module files. If you are creating a package that explicitly needs to load one of these files, then you can use:

`\DTMusemodule`

```
\DTMusemodule{<language>}{<name>}
```

where *<language>* is the `babel` or `polyglossia` language or dialect name that identifies the relevant `\date<language>` macro (for example, `english`) and *<name>* is the same as above (for example, `en-GB`).

Note that `\RequireDateTimeModule` (which is also internally used by `\DTMusemodule`) stores a mapping from the language name and the module name. You can determine what module was loaded for a given dialect name using

`\DTMdialecttomodulemap`

```
\DTMdialecttomodulemap{<dialect>}
```

This expands to the required module name or `\relax` if the given dialect name wasn't used to load a module. For example:

```
\documentclass{article}
\usepackage[british]{datetime2}

\begin{document}

british map: \DTMdialecttomodulemap[british].
english map: \DTMdialecttomodulemap[english].

\end{document}
```

This produces:

british map: en-GB. english map: .

In the above, the second instance expands to `\relax`.

If you want to provide a language module don't assume all users want to use the same input encoding or babel shorthands as you. Use \LaTeX commands for non-ASCII characters (and remember to use `\protect` where necessary).

As an addendum to the above warning, Lua \TeX and Xe \TeX support UTF-8 characters without the need to make them active, so I recommend you provide two files: one with the \LaTeX commands, such as `\c`, for (PDF) \LaTeX users, and one with UTF-8 characters for Lua \LaTeX and Xe \LaTeX users. For example, the `fr-FR` module could start with:

```
\ProvidesDateTimeModule{fr-FR}

\RequirePackage{ifxetex,ifluatex}

\ifxetex
  \RequireDateTimeModule{french-utf8}
\else
  \ifluatex
    \RequireDateTimeModule{french-utf8}
  \else
    \RequireDateTimeModule{french-ascii}
  \fi
\fi
```

This helps provide fully expandable dates for Lua \LaTeX and Xe \LaTeX users. (See the Scottish or Irish modules for examples.)

7 Standalone Month or Weekday Names

If you want the month name or weekday name to appear in a section or chapter heading, it's best to use the expandable commands provided by the language modules rather than the robust commands provided by `datetime2-calc`. Remember that you can't use robust commands in PDF bookmarks and such commands may prevent case-changing in headers for page styles that use `\MakeUppercase`.

The language or regional modules described in Section 6 typically provide an expandable command

```
\DTM<root-language>monthname{<n>}
```

which takes a numerical argument that indicates the month number. This command is used in the date style which ensures that even if the document language has switched but not the date style, then the month name will be in the correct language for that style. (Otherwise you could end up with a mix of style from one dialect using names from another language, which was one of the problems with the original `datetime` package.)

For example, if the `english` module is loaded (which is automatically loaded by the English dialect modules, such as `en-GB`) then the command `\DTMenglishmonthname` is defined. So if you're writing in English and you want to display just the month name, then you can do:

```
\DTMenglishmonthname{1}
```

Some language modules, where month names aren't automatically capitalised, may additionally define a version that has the first letter in upper case. For example, the `french` module defines `\DTMfrenchMonthname` in addition to `\DTMfrenchmonthname`.

Some of the modules may have other alternatives. For example, the `serbian` module provides Cyrillic (`\DTMserbiancyrmonthname`) and Latin (`\DTMserbianlatinmonthname`) month names. It also provides `\DTMserbianmonthname`, which defaults to `\DTMserbiancyrmonthname` but can be redefined using `\DTMLangsetup`.

To find out the available commands for the module you are using, see that module's documentation.

If you are writing a document that uses multiple languages and you simply want to display the month name in the currently selected language, then you can use the robust command

```
\DTMmonthname{<n>}
```

provided by the `datetime2-calc` package, described in Section 9. Remember that the `datetime2-calc` package also loads the `pgfcalendar` package, which provides the expandable command `\pgfcalendarmonthname`. (The `pgfcalendar` package provides multilingual support via the translator package.)

Some of the language modules additionally provide a command that displays the first letter in upper case. This isn't provided for languages where the month name is always displayed with a capital first letter, such as in English. For example, the `serbian` module also defines `\DTMserbiancyrMonthname` and `\DTMserbianlatinMonthname`, with `\DTMserbianMonthname` initially defined to use the Cyrillic version.

So, if you specifically want to display the Serbian Cyrillic month name with the first letter in upper case, you need to make sure the `serbian` module is loaded and then use the provided `\DTMserbiancyrMonthname` command.

If you want the month name to vary according to the current language setting in the document, you can use the robust command

```
\DTMMonthname{<n>}
```

provided by `datetime2-calc`, which will first attempt `\DTM<language>Monthname` and then `\DTM<language>monthname` before falling back on the `\pgfcalendarmonthname`, see Section 9 for further details.

Some, but not all, language modules provide a command (or commands) for month name abbreviations. It's up to the maintainer of the module to add these if they currently aren't provided. You will need to check the module documentation to find out if abbreviations are supported. If they are supported, they should be in the form

```
\DTM<root-language>shortmonthname{<n>}
```

Again there may be variations, such as new and old styles or alternative alphabets. For example, the `english` module provides `\DTMenglishshortmonthname`.

As with the full form, if you want to display the abbreviation in a specific language (or variation), then use the command provided by the relevant language module. If you want the abbreviation to pick up the current language, then you can use the robust command

```
\DTMshortmonthname{<n>}
```

provided by the `datetime2-calc` package. This will fall back on `\pgfcalendarmonthshortname` if `\DTM<language>shortmonthname` isn't defined. See Section 9 for further details.

Modules may additionally provide a version of the abbreviated form that starts with a capital letter. This should typically be in the form

```
\DTM<root-language>shortMonthname{<n>}
```

Check the module documentation to see if this is provided. Again, the `datetime2-calc` package provides a robust command

```
\DTMshortMonthname{<n>}
```

that attempts to determine the relevant module command from the language name.

Language modules may or may not provide a command that displays the weekday name. As with the month name, there may or may not be an abbreviated version or capital first letter version or variations such as new/old styles. Check the module documentation for further details. If the module doesn't provide a weekday name macro, then the provided styles won't support the `showdow` option.

If the module provides weekday name support, then the name will typically be provided by a macro in the form

```
\DTM<root-language>weekdayname{<n>}
```

where `<n>` is an integer from 0 (Monday) to 6 (Sunday). For example, the `english` module provides `\DTMenglishweekdayname`.

Again, the `datetime2-calc` package provides *robust* commands that attempt to find the relevant module-provided command based on the current language. If not found, the fallback commands are those provided by the `pgfcalendar` package. See Section 9 for further details.

8 Package Options

The following package options are provided. Most of these are *<key>=<value>* options, unless stated otherwise.

Settings that govern the predefined numerical styles (not including the fixed styles `iso` and `pdf`):

yearmonthsep This sets the separator between the year and month for the big-endian and little-endian styles. Default: - (hyphen). Note that if you want a space as a separator you need to use `\space`. If you simply use a space character (for example, `yearmonthsep={ }`) then the separator will be discarded. The same applies for the other separators described below.

monthdaysep This sets the separator between the month and day. Default: - (hyphen).

dayyearsep This sets the separator between the day and year for the middle-endian styles. Default: - (hyphen).

datesep This sets the separators between the day and month, the month and year, and the day and year. Example:

```
\usepackage[datesep=/{datetime2}
```

This is equivalent to:

```
\usepackage[yearmonthsep=/,monthdaysep=/,dayyearsep=/{datetime2}
```

hourminsep This sets the separator between the hour and minute. (Both for the time and for the zone.) Default: : (colon).

minsecsep This sets the separator between the minute and seconds. Default: : (colon).

timesep This sets the separators between the hour and minute and between the minute and seconds. Example:

```
\usepackage[timesep=:]{datetime2}
```

This is equivalent to:

```
\usepackage[hourminsep=:,minsecsep=:]{datetime2}
```

The following settings are used by the predefined numerical styles when displaying the full date, time and zone (excluding the fixed styles `iso` and `pdf`) with commands that use `\DTMdisplay` or `\DTMDisplay`.

datetimesep Sets the separator between the date and time. Default: `\space`.

timezonesep Sets the separator between the time and zone. Default: empty.

The following settings are used by the predefined styles and may also be used by the language modules.

showseconds Boolean key to determine whether or not to show the seconds when the time is displayed. The `iso` style honours this setting but the `pdf` style ignores it. Default: `true` unless \XeTeX is used (and `texosquery` hasn't first been loaded with shell escape enabled).

showdate Boolean key to determine whether or not to show the date with commands that use `\DTMdisplay` or `\DTMDisplay`. (Some styles may ignore this.) The `iso` style honours this setting but the `pdf` style ignores it. Default: `true`.

showzone Boolean key to determine whether or not to show the time zone with commands that use `\DTMdisplay` or `\DTMDisplay`. (Some styles may ignore this.) The `iso` style honours this setting but the `pdf` style ignores it. Default: `true` unless \XeTeX is used (and `texosquery` hasn't first been loaded with shell escape enabled).

showzoneminutes Boolean key to determine whether or not to show the zone offset minutes. The `iso` style honours this setting but the `pdf` style ignores it. This setting is ignored if `showzone` is `false`. Default: `true`.

showisoZ Boolean key to determine whether or not to show UTC+00:00 as Z instead of numerically. This option may be ignored by zone styles that use the zone mappings. If you want all the time zones in military form, you can use `\DTMNaToZoneMaps` to set up the time zone abbreviations and then use a zone style that uses the mappings. Default: `true`.

General settings:

useregional Allowed values: `false`, `text` or `numeric`. You may also use `num` as an abbreviation for `numeric`. If no value is supplied `text` is assumed.

If you haven't loaded `babel` or `polyglossia`, this key only has an effect when used as a package option. Consider instead using `\DTMsetregional`.

This key determines whether or not to *use* the loaded regional settings and, if the regional setting should be used, it determines whether the text style (months as words) or numeric style should be used. If you have loaded one of those packages, the change

comes into effect at module load time and whenever `\date<language>` is used (which includes at the beginning of the document environment). If you want to switch the style at any other time, you need to use `\DTMsetstyle` but unless `useregional=false` the next instance of `\date<language>` will change the style.

Note that setting this option to `false` doesn't prevent the modules from being loaded. It just prevents them from automatically setting the style and prevents `\date<language>` from changing the style if you are using `babel` or `polyglossia`.

The default value is `false` unless the language or region is passed to the `datetime2` package option list. However, using `style` will set `useregional` to `false`.

If you want to change this value after the language modules have been loaded, instead of using `\DTMsetup`, you can use

`\DTMsetregional`

```
\DTMsetregional[<value>]
```

This will do `\DTMsetup{userregional=<value>}`. Then if `<value>` is `false`, it will set the default style otherwise it will use `\date<lang>` if it's defined (where `<lang>` is given by `\language`). If `\date<lang>` isn't defined, it will iterate over the list of dialects associated with the document and use `\DTMtryregional` for each dialect. If `<value>` is omitted, `text` is assumed.

Examples:

```
\documentclass[british]{article}
\usepackage{babel}
\usepackage{datetime2}
```

In the above `useregional` is `false`.

```
\documentclass{article}
\usepackage[british]{datetime2}
```

In the above `useregional` is `text`.

```
\documentclass{article}
\usepackage[british,style=iso]{datetime2}
```

In the above `useregional` is `false`. (The `british` option implements `usnumerical=text` but the `style` option then implements `usnumerical=false`.)

```
\documentclass{article}
\usepackage[style=iso,british]{datetime2}
```

In the above `useregional` is `text`. (The `style` option implements `usnumerical=false` but the `british` option then implements `usnumerical=text`.)

style Sets the current style using `\DTMsetstyle` when the `datetime2` package has finished loading. This also sets `useregional=false` but that setting can be overridden later in the option list.

Default value: empty (use the default style or the regional style, according to the value of `useregional`).

This key isn't available in `\DTMsetup`. Use `\DTMsetstyle` instead.

calc Load the `datetime2-calc` package. This will allow the day of week to be computed and allow you to use the `pgfcalendar` offset style date formats in commands like `\DTMdate` as well as defining the commands described in Section 9. This option doesn't take a value. It can't be switched off. This option can't be used in `\DTMsetkeys`. The default is to not load `datetime2-calc`.

showdow This is a boolean key that determines whether or not to show the day of week in styles that support this. Note that `showdow=true` will automatically load `datetime2-calc` so

```
\usepackage[showdow]{datetime2}
```

is equivalent to

```
\usepackage[showdow,calc]{datetime2}
```

This option may be used in `\DTMsetup`, but if you attempt to switch it on in the document environment you'll get an error if the `datetime2-calc` package hasn't been loaded.

Not all styles support this setting. Default: `false`.

This option is actually a language-dependent option and isn't used by the base package, but it's implemented as a package option as the `datetime2-calc` package is also needed if the day of the week should be displayed. As a package option, `showdow` can automatically load the required package.

You need to check the documentation to find out which styles check the `showdow` setting as not all of them do.

warn This is a boolean key. If `true` (default) `datetime2` warnings will be displayed. If `false`, the warnings will be suppressed. Default: `true`.

Any additional option passed to the `datetime2` package (not through `\DTMsetup`) will be considered a `tracklang` option and will be passed to `\TrackPredefinedDialect`. (See the `tracklang` documentation for further details of that command.)

Apart from calc, style and the regional options, all the above options can also be set using:

`\DTMsetup`

```
\DTMsetup{<option list>}
```

The language modules may additionally provide options which can be set using:

`\DTMlangsetup`

```
\DTMlangsetup[<module-name list>]{<option list>}
```

This will set the *<option list>* for each module listed in *<module-name list>*. Unknown options will generate a warning rather than an error message. The default value of *<module-name list>* is the list of all loaded modules.

Example:

```
\documentclass{article}
\usepackage[british]{datetime2}
\DTMlangsetup{mapzone}
```

The module list here is `english-base,en-GB` and since the `english-base` doesn't have a `mapzone` option, this will result in a warning:

```
Package datetime2 Warning: Region `english-base' has ignored
(datetime2)                the following settings:
(datetime2)                mapzone
```

You can either ignore the warning or use the optional argument to exclude the `english-base` module:

```
\documentclass{article}
\usepackage[british]{datetime2}
\DTMlangsetup[en-GB]{mapzone}
```

Alternatively you can use the starred version:

```
\documentclass{article}
\usepackage[british]{datetime2}
\DTMlangsetup*{mapzone}
```

Note that some modules may have options with the same name as the above listed package options, but the keys are defined in different families (see `xkeyval` documentation) so you need to take care to use `\DTMsetup` for package-wide settings and `\DTMlangsetup` for the module-specific settings.

For example, the `datesep` package option described above is used by the predefined numerical styles but regional modules that provide their own numerical styles may use a different date separator that matches their region so they may also provide a `datesep` option independent of the base `datesep` option.

Examples:


```

\documentclass [british] {article}
\usepackage [datesep=.] {datetime2}
\begin{document}
\today
\end{document}

```

The above displays the date in the form 2018.07.20 since the default style is in use and datesep is used as a package option.

```

\documentclass [british] {article}
\usepackage {datetime2}
\DTMsetup {datesep=.}
\begin{document}
\today
\end{document}

```

The above displays the date in the form 2018.07.20 since the default style is in use and datesep is used in \DTMsetup.

```

\documentclass [british] {article}
\usepackage {datetime2}
\DTMlangsetup {datesep=.}
\begin{document}
\today
\end{document}

```

The above displays the date in the form 2018-07-20 since the default style is in use but datesep is used in \DTMlangsetup, which only influences the en-GB-numeric style, which isn't the current style.

```

\documentclass [british] {article}
\usepackage [userregional=numeric] {datetime2}
\DTMlangsetup {datesep=.}
\begin{document}
\today
\end{document}

```

The above displays the date in the form 20.7.2018 since the en-GB-numeric style is in use and datesep is used in \DTMlangsetup.

```

\documentclass [british] {article}
\usepackage [userregional] {datetime2}
\DTMlangsetup {datesep=.}
\begin{document}
\today
\end{document}

```

The above displays the date in the form 20th July 2018 since the en-GB style is in use and datesep is used in \DTMlangsetup, which only influences the en-GB-numeric style.

```
\documentclass[british]{article}
\usepackage[userregional=numeric]{datetime2}
\DTMsetup{datesep=.}
\begin{document}
\today
\end{document}
```

The above displays the date in the form 20/7/2018 since the en-GB-numeric style is in use but datesep is used in \DTMsetup which influences the base predefined numeric styles not the regional styles.

9 The datetime2-calc Package

The datetime2-calc package can be loaded after datetime2 in the usual way:

```
\usepackage{datetime2}
\usepackage{datetime2-calc}
```

or using the calc package option to datetime2:

```
\usepackage[calc]{datetime2}
```

or by using showdow=true:

```
\usepackage[showdow]{datetime2}
```

This package loads the pgfcalendar package which provides a way of computing the day of week from a given date. Once datetime2-calc has been loaded, you can enable or disable the weekday in dates where the style supports this, but note that *not all styles support this*, even if the datetime2-calc package has been loaded.

As with the commands in Section 4, the commands described below that save date/time information will *overwrite* any previously defined date/time data with the same identifying *<name>*. However, they may only overwrite specific elements of the data (for example, just the year, month, day and day of week elements) and leave the other elements unchanged. Where the remaining elements are undefined they'll be set to zero, except for the day of week element, which will be set to -1.

In addition to enabling the weekday calculations, the datetime2-calc package also provides the following commands:

`\DTMsavejulianday`

```
\DTMsavejulianday{<name>}{<number>}
```

This uses `\pgfcalendarjuliantodate` to obtain the year, month and day from the given Julian day number and uses `\pgfcalendarjuliantoweekday` to obtain the day of week and then saves it. The date can later be used with commands such as `\DTMuse{<name>}` described in Section 4. Example:

```
\DTMsavejulianday{mydate}{2457023}
```

`\DTMsaveddatetojuliandate`

```
\DTMsaveddatetojulianday{<name>}{<register>}
```

This uses `\pgfcalendardatetojulian` to convert a previously saved date (identified by $\langle name \rangle$) to a Julian day. The result is stored in $\langle register \rangle$ which should be a count register (not a \LaTeX counter name). Example:

```
\newcount\myct
\DTMsaveddatetojulianday{mydate}{\myct}
```

`\DTMsaveddateoffsettojulianday`

```
\DTMsaveddateoffsettojulianday{<name>}{<offset>}{<register>}
```

This is like the previous command but converts the date obtained by incrementing the saved date with $\langle offset \rangle$. The result is stored in $\langle register \rangle$. This is equivalent to

```
\pgfcalendardatetojulian{<y>-<m>-<d>+<offset>}{<register>}
```

where $\langle y \rangle$, $\langle m \rangle$ and $\langle d \rangle$ are the year, month and day fetched from the saved date. A negative $\langle offset \rangle$ indicates an earlier date. Example:

```
\DTMsaveddateoffsettojulianday{mydate}{2}{\myct}
```

or

```
\DTMsaveddateoffsettojulianday{mydate}{-7}{\myct}
```

`\DTMifdate`

```
\DTMifdate{<name>}{<test>}{<true>}{<false>}
```

This is just a convenient interface to `\pgfcalendarifdate` for a saved date (identified by $\langle name \rangle$). The remaining arguments are the same as the final three arguments of `\pgfcalendarifdate`. Note that the equals, at least, at most and between keywords available in $\langle test \rangle$ need to be in the format specified by the pgf manual, but remember that you can use commands like `\DTMfetchyear`. Example:

```
Is \texttt{mydate2} (\DTMusedate{mydate2}) before
\texttt{mydate} (\DTMusedate{mydate})?
\DTMifdate
{mydate2}
{at most=
\DTMfetchyear{mydate}-\DTMfetchmonth{mydate}-\DTMfetchday{mydate}}
{yes}{no}.
```

`\DTMsaveddatediff`

```
\DTMsaveddatediff{<name1>}{<name2>}{<register>}
```

Computes the difference (in days) between two saved dates and stores the result in the given count register. The first date is identified by $\langle name1 \rangle$ and the second date is identified by

<name2>. The dates are converted to their respective Julian day numbers *<J1>* and *<J2>* and the result is given by *<J1>-<J2>*.

Note that the time and zone are not taken into account, even if they were provided when the dates were stored.

Example:

```
\DTMsaveddatediff{mydate}{mydate2}{\myct}

\DTMusedate{mydate} is
\ifnum\myct=0
the same day as
\else
\ifnum\myct<0
\number-\myct\space day\ifnum\myct<-1s\fi\space before
\else
\number\myct\space day\ifnum\myct>1s\fi\space after
\fi
\fi
\DTMusedate{mydate2}.
```

The `datetime2-calc` package also provides commands that convert a datetime instance into Zulu¹ time (UTC+00:00).

`\DTMsaveaszulutime`

```
\DTMsaveaszulutime{<name>}{<YYYY>}{<MM>}{<DD>}{<hh>}{<mm>}
{<ss>}{<TZh>}{<TZm>}
```

This converts the given datetime instance into UTC+00:00 and saves the result. You can then use the date with commands like `\DTMused` described in Section 4. The *<name>* argument is the label identifying the saved data. The other arguments are all numbers. Example:

```
\DTMsaveaszulutime{mydate}{2014}{6}{3}{20}{45}{0}{6}{0}
```

`\DTMtozulu`

```
\DTMtozulu{<name1>}{<name2>}
```

Uses `\DTMsaveaszulutime` to convert the datetime stored in *<name1>* and saves it to *<name2>*. Example:

```
\DTMsavetimestamp{mydate}{2014-05-01T03:55:00 -06:00}
Original date: \DTMused{mydate}.
```

```
\DTMtozulu{mydate}{mydate2}
UTC+00:00: \DTMused{mydate2}.
```

¹That's Zulu as in the NATO alphabet representation of the letter Z.

The above produces (using the default format):

Original date: 2014-05-01 03:55:00-06:00.

UTC+00:00: 2014-05-01 09:55:00Z.

The pgfcalendar package also provides a variety of useful date-related commands. See the documentation (part of the pgf manual) for further details. Note that the language modules don't use pgfcalendar month and weekday names as the pgfcalendar package isn't loaded by default and the styles need to match the language with the syntax. However, since the datetime2-calc package automatically loads the pgfcalendar package, as from v1.3 the datetime2-calc provides robust month name and weekday name commands that may be used outside of date styles, which fallback on the commands provided by pgfcalendar.

The following commands, which are all robust, should not be used in date styles, since each language style must use the name macro for that specific language to match the style. Make sure you have the relevant language module installed and loaded to allow these commands to work correctly. See also Section 7. Remember that instead of these robust commands, you can simply just use the commands provided by the pgfcalendar package. (See the pgf manual for further details.)

`\DTMmonthname`

`\DTMmonthname{<n>}`

This checks if `\DTM<lang>monthname` exists where `<lang>` is given by `\language`. If so, that macro is used. For example:

```
\documentclass{article}
\usepackage[english]{babel}
\usepackage[calc]{datetime2}
\begin{document}
\DTMmonthname{12}
\end{document}
```

Here `\language` is `english`, so this uses `\DTMenglishmonthname` which is defined by the `english` module.

If the test with `\language` didn't work, `\DTMmonthname` then tries with `<lang>` set to

```
\TrackedLanguageFromDialect{\language}
```

(`\TrackedLanguageFromDialect` is provided by the `tracklang` package.) If `\DTM<lang>monthname` exists, then this command is used. For example:

```
\documentclass{article}
\usepackage[british]{babel}
\usepackage[calc]{datetime2}
\begin{document}
\DTMmonthname{12}
\end{document}
```

Here `\language` is `british`, so this again uses `\DTMenglishmonthname` as the root language is obtained from the dialect to language mapping.

Note that this won't work if you confuse `tracklang` by using an alternative dialect name in the class option or by directly loading `tracklang` with different dialect labels.

In the event that neither of those commands exist, `\DTMmonthname` will fallback on `\pgfcalendarmonthname` (provided by the `pgfcalendar` package). This will also issue a warning. For example:

```
\documentclass{article}
\usepackage[calc]{datetime2}
\begin{document}
\DTMmonthname{12}
\end{document}
```

This produces the warning message:

```
Can't find underlying language macro for
\DTMmonthname (language: english);
using pgfcalendar macro instead
```

and uses `\pgfcalendarmonthname` instead of `\DTMenglishmonthname` (which hasn't been defined because the `english` module *hasn't been loaded*).

You can switch off the warning by setting the `warn` option to `false` or by redefining `\dtmnamewarning` to ignore its argument.

`\DTMMonthname`

```
\DTMMonthname{<n>}
```

This checks if `\DTM<lang>Monthname` exists. First where `<lang>` is given by `\language` and then where `<lang>` is given by

```
\TrackedLanguageFromDialect{\language}
```

If neither of those values of `<lang>` match a defined command, `\DTMMonthname` then tests for non-case-changing versions `\DTM<lang>monthname`. This is because not all language modules provide a macro for use at the start of a sentence since some languages always start month names with a capital letter. For example, the `english` module provides `\DTMenglishmonthname` but doesn't provide an upper case alternative, since English month names always start with a capital. Therefore:

```
\documentclass{article}
\usepackage[english]{babel}
\usepackage[calc]{datetime2}
\begin{document}
\DTMMonthname{12}
\end{document}
```

just uses `\DTMenglishmonthname`.

As before, if the relevant command can't be detected for any case of $\langle lang \rangle$ for either `\DTM $\langle lang \rangle$ Monthname` or `\DTM $\langle lang \rangle$ monthname` (for example, the required language module hasn't been loaded) then `\DTMMonthname` will use `\pgfcalendarmonthname` and attempt to convert the first letter to upper case.

Since some *but not all* language modules also provide month name abbreviations, the `datetime2-calc` package also provides:

`\DTMshortmonthname`

```
\DTMshortmonthname{ $\langle n \rangle$ }
```

This behaves in a similar way to `\DTMmonthname` but tries to determine if `\DTM $\langle lang \rangle$ shortmonthname` exists (where $\langle lang \rangle$ is either `\language` or obtained from `\language` using `tracklang`'s dialect to language mapping). If no command can be found, the fallback uses `\pgfcalendarmonthshortname` provided by the `pgfcalendar` package. For example, if the language module hasn't been loaded or if the language module doesn't provide an abbreviated version.

Similarly there is a version for the start of a sentence for languages that normally use lower case month names:

`\DTMshortMonthname`

```
\DTMshortMonthname{ $\langle n \rangle$ }
```

There are also analogous commands for the weekday names, where $\langle n \rangle$ is an integer from 0 (Monday) to 6 (Sunday). This index can be computed using:

`\DTMcomputedayofweekindex`

```
\DTMcomputedayofweekindex{ $\langle date \rangle$ }{ $\langle cs \rangle$ }
```

where $\langle date \rangle$ is in the form $\langle YYYY \rangle$ - $\langle MM \rangle$ - $\langle DD \rangle$ and $\langle cs \rangle$ is a control sequence in which to store the result. Remember that date styles automatically access the day of week index from the fourth argument of `\DTMdisplaydate`, so this command shouldn't be used within a date style.

`\DTMweekdayname`

```
\DTMweekdayname{ $\langle n \rangle$ }
```

This checks if `\DTM $\langle lang \rangle$ weekdayname` exists where $\langle lang \rangle$ is given by `\language`. If it does, that macro is used. For example:

```
\documentclass{article}
\usepackage[english]{babel}
\usepackage[calc]{datetime2}
\begin{document}
\DTMweekdayname{6}
\end{document}
```


This uses `\DTMenglishweekdayname`, which is provided by the `english` module.

If `\DTM⟨lang⟩weekdayname` doesn't exist with `⟨lang⟩` set to `\languagename`, `\DTMweekdayname` then tests with `⟨lang⟩` set to:

```
\TrackedLanguageFromDialect{\languagename}
```

If the command `\DTM⟨lang⟩weekdayname` exists in this case, that command is used. For example:

```
\documentclass{article}
\usepackage[british]{babel}
\usepackage[calc]{datetime2}
\begin{document}
\DTMweekdayname{6}
\end{document}
```

This uses `\DTMenglishweekdayname` as it can determine (through `tracklang`) that `british` has been defined as a dialect of `english`.

If this second test fails, then `\DTMweekdayname` will issue a warning and fallback on `\pgfcalendarweekdayname`, provided by the `pgfcalendar` package. For example:

```
\documentclass{article}
\usepackage[calc]{datetime2}
\begin{document}
\DTMweekdayname{6}
\end{document}
```

This produces the warning message:

```
Can't find underlying language macro for
\DTMweekdayname (language: english);
using pgfcalendar macro instead
```

As before, this warning is produced with `\dtmnamewarning`.

The first letter upper case version is:

`\DTMWeekdayname`

```
\DTMWeekdayname{⟨n⟩}
```

As with `\DTMMonthname`, if no upper case version can be found in the relevant language module this will use the non-case-changing version. The fallback `pgfcalendar` macro is again `\pgfcalendarweekdayname` with an attempt to convert the first letter to upper case.

Again, abbreviations may or may not be supported by language modules. If they're not supported, the fallback is `\pgfcalendarweekdayshortname`.

`\DTMshortweekdayname`

```
\DTMshortweekdayname{⟨n⟩}
```

which will attempt to use `\DTM⟨lang⟩shortweekdayname` and

`\DTMshortWeekdayname`

```
\DTMshortWeekdayname{<n>}
```

which will attempt to use `\DTM<lang>shortWeekdayname` or `\DTM<lang>shortweekdayname`.
For completeness, there's also a language-sensitive date ordinal command:

`\DTMordinal`

```
\DTMordinal{<n>}
```

where `<n>` is a number from 1 to 31. Again, this shouldn't be used in date styles, but only if a standalone date ordinal is required. For most languages, this only has a suffix for the first day of the month (that is where `<n>` is 1) or the suffix may simply be a full stop (period). It should not be confused with `fntcount's \ordinalnum` command, which is for general ordinals rather than date-specific ordinals.

10 Migrating from datetime

This section is for users who want to switch over from the old datetime package.

Note that datetime2 is modularised for improved efficiency both in terms of package overheads and spreading the maintenance load. This means that you only need to install datetime2 if you only want the base numeric styles, but if you want multilingual or regional support, you need to additionally load the required language module or modules.

For example, consider the following document that uses datetime:

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[french]{babel}
\usepackage{datetime}
\begin{document}
\today.
\end{document}
```

This just needs to have the datetime package installed, which includes the necessary file `dt-french.def`.

This example can be adjusted for datetime2:

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[french]{babel}
\usepackage[userregional]{datetime2}
\begin{document}
\today.
\end{document}
```

This requires both datetime2 and the `french` module, so you need to ensure that both have been installed.

The above example also draws attention to another change from datetime and that concerns the default package behaviour. The datetime package defaults to a British date style, unless babel has been loaded first, whereas the datetime2 package defaults to an ISO numeric style, unless language or regional settings are provided in the class option. As illustrated in the above, you need the `userregional` option if you want `\datefrench` (or equivalent) to switch the date style.

10.1 datetime package options

The datetime package provides the following options, which can be emulated with datetime2 or through one of its dependent modules or packages:

long This option was designed for full British dates, and is the default if babel isn't loaded.

Example:

```
\documentclass{article}
\usepackage[long]{datetime}
\begin{document}
\today
\end{document}
```

This produces the date in the form: Wednesday 20th January, 2016.

This can be achieved with datetime2 and the `english` module as follows:

```
\documentclass{article}
\usepackage[en-GB,showdow]{datetime2}
\DTMLangsetup[en-GB]{ord=raise,monthyearsep={, \space}}
\begin{document}
\today
\end{document}
```

short This option was designed for abbreviated British dates. For example:

```
\documentclass{article}
\usepackage[short]{datetime}
\begin{document}
\today
\end{document}
```

This produces the date in the form: Wed 20th Jan, 2016

This can be achieved with datetime2 and the `english` module as follows:

```
\documentclass{article}
\usepackage[en-GB,showdow]{datetime2}
\DTMLangsetup[en-GB]{abbr,ord=raise,monthyearsep={, \space}}
\begin{document}
\today
\end{document}
```

iso This option was designed for $\langle YYYY \rangle$ - $\langle MM \rangle$ - $\langle DD \rangle$ dates. For example:

```
\documentclass{article}
\usepackage[iso]{datetime}
\begin{document}
\today
\end{document}
```

This produces the date in the form: 2016-01-20. This is default for `datetime2`:

```
\documentclass{article}
\usepackage{datetime2}
\begin{document}
\today
\end{document}
```

If you don't want the style to depend on the separator settings, you can use the `iso` style:

```
\documentclass{article}
\usepackage[style=iso]{datetime2}
\begin{document}
\today
\end{document}
```

yyyymmdd This option was designed for $\langle YYYY \rangle / \langle MM \rangle / \langle DD \rangle$ dates. For example:

```
\documentclass{article}
\usepackage[yyyymmdd]{datetime2}
\begin{document}
\today
\end{document}
```

This produces the date in the form: 2016/01/20. This is very nearly in the same form as the default `datetime2` style. All that needs changing are the separators between the year, month and day:

```
\documentclass{article}
\usepackage[datesep=/]{datetime2}
\begin{document}
\today
\end{document}
```

ddmmyyyy This option was designed for $\langle DD \rangle / \langle MM \rangle / \langle YYYY \rangle$ dates. For example:

```
\documentclass{article}
\usepackage[ddmmyyyy]{datetime2}
\begin{document}
\today
\end{document}
```

This produces the date in the form: 20/01/2016. This can be changed to `datetime2` with a couple of package options:

```
\documentclass{article}
\usepackage[datesep=/,style=ddmmyyyy]{datetime2}
\begin{document}
\today
\end{document}
```

dmyyyy This option was designed for $\langle D \rangle / \langle M \rangle / \langle YYYY \rangle$ dates. For example:

```
\documentclass{article}
\usepackage[ddmmyyyy]{datetime}
\begin{document}
\today
\end{document}
```

This produces the date in the form: 20/1/2016. This can be changed to `datetime2` with a couple of package options:

```
\documentclass{article}
\usepackage[datesep=/,style=dmyyyy]{datetime}
\begin{document}
\today
\end{document}
```

ddmmyy This option was designed for $\langle DD \rangle / \langle MM \rangle / \langle YY \rangle$ dates. For example:

```
\documentclass{article}
\usepackage[ddmmyy]{datetime}
\begin{document}
\today
\end{document}
```

This produces the date in the form: 20/01/16. This can be changed to `datetime2` with a couple of package options:

```
\documentclass{article}
\usepackage[datesep=/,style=ddmmyy]{datetime}
\begin{document}
\today
\end{document}
```

dmyy This option was designed for $\langle D \rangle / \langle M \rangle / \langle YY \rangle$ dates. For example:

```
\documentclass{article}
\usepackage[dmyy]{datetime}
\begin{document}
\today
\end{document}
```

This produces the date in the form: 20/1/16. This can be changed to `datetime2` with a couple of package options:

```
\documentclass{article}
\usepackage[datesep=/,style=dmyy]{datetime}
\begin{document}
\today
\end{document}
```

text This option was designed for a full UK textual date. For example:

```
\documentclass{article}
\usepackage[text]{datetime}
\begin{document}
\today
\end{document}
```

This produces the date in the form: Wednesday the Twentieth of January, Two Thousand and Sixteen

This document can be changed to datetime2 through the datetime2-en-fulltext package, which needs to be installed separately:

```
\documentclass{article}
\usepackage{datetime2-en-fulltext}
\DTMsetdatestyle{en-FullText}
\begin{document}
\today
\end{document}
```

us This option was designed for the standard US date. For example:

```
\documentclass{article}
\usepackage[us]{datetime}
\begin{document}
\today
\end{document}
```

This produces the date in the form: January 20, 2016

This can be achieved with datetime2 and the **english** module as follows:

```
\documentclass{article}
\usepackage[en-US]{datetime2}
\begin{document}
\today
\end{document}
```

mmddyyy This option was designed for $\langle MM \rangle / \langle DD \rangle / \langle YYYY \rangle$ dates. For example:

```
\documentclass{article}
\usepackage[mmddyyy]{datetime}
\begin{document}
\today
\end{document}
```

This produces the date in the form: 01/20/2016. This can be changed to `datetime2` with a couple of package options:

```
\documentclass{article}
\usepackage[datesep=/,style=mmddyyyy]{datetime}
\begin{document}
\today
\end{document}
```

mdyyyy This option was designed for $\langle M \rangle / \langle D \rangle / \langle YYYY \rangle$ dates. For example:

```
\documentclass{article}
\usepackage[mmddyyyy]{datetime}
\begin{document}
\today
\end{document}
```

This produces the date in the form: 1/20/2016. This can be changed to `datetime2` with a couple of package options:

```
\documentclass{article}
\usepackage[datesep=/,style=mdyyyy]{datetime}
\begin{document}
\today
\end{document}
```

mmddyy This option was designed for $\langle MM \rangle / \langle DD \rangle / \langle YY \rangle$ dates. For example:

```
\documentclass{article}
\usepackage[mmddyy]{datetime}
\begin{document}
\today
\end{document}
```

This produces the date in the form: 01/20/16. This can be changed to `datetime2` with a couple of package options:

```
\documentclass{article}
\usepackage[datesep=/,style=mmddyy]{datetime}
\begin{document}
\today
\end{document}
```

mdyy This option was designed for $\langle M \rangle / \langle D \rangle / \langle YY \rangle$ dates. For example:

```
\documentclass{article}
\usepackage[mdyy]{datetime}
\begin{document}
\today
\end{document}
```


This produces the date in the form: 1/20/16. This can be changed to `datetime2` with a couple of package options:

```
\documentclass{article}
\usepackage[datesep=/,style=mdyy]{datetime}
\begin{document}
\today
\end{document}
```

raise This option was designed to make the ordinal `st,nd,rd,th` appear as a subscript. It was originally just intended for British dates. This is one of the default settings for `datetime`. For example:

```
\documentclass{article}
\usepackage[raise]{datetime}

\begin{document}
\today
\end{document}
```

This produces the date in the form: Wednesday 20th January, 2016

With `datetime2`, this setting *may* be provided by a language module, where appropriate.

For example:

```
\documentclass{article}
\usepackage[en-GB,showdown]{datetime2}
\DTMLangsetup[en-GB]{ord=raise}

\begin{document}
\today
\end{document}
```

level This option was designed to make the ordinal `st,nd,rd,th` appear level with the rest of the text (to counteract the previous option). For example:

```
\documentclass{article}
\usepackage[level]{datetime}

\begin{document}
\today
\end{document}
```

This produces the date in the form: Wednesday 20th January, 2016

With `datetime2` this setting may be provided by a language module, where appropriate.

This is, in fact, the default setting for the `en-GB` style:

```
\documentclass{article}
```

```
\usepackage[en-GB,showdow]{datetime2}
```

```
\begin{document}  
\today  
\end{document}
```

but can be explicitly set using:

```
\DTMlangsetup[en-GB]{ord=level}
```

dayofweek This option was designed to show the weekday name for those styles that supported it. This is one of the default datetime settings. For example:

```
\documentclass{article}  
\usepackage[dayofweek]{datetime}  
  
\begin{document}  
\today  
\end{document}
```

This produces the date in the form: Wednesday 20th January, 2016

With datetime2 this setting *may* be provided by a language module, where supported. For example:

```
\documentclass{article}  
\usepackage[en-GB,showdow]{datetime2}  
  
\begin{document}  
\today  
\end{document}
```

Some language modules don't support this option (just as some language settings with datetime also don't support the dayofweek option.)

nodayofweek This option was designed to hide the weekday name for those styles that supported it (to counteract the previous option). For example:

```
\documentclass{article}  
\usepackage[nodayofweek]{datetime}  
  
\begin{document}  
\today  
\end{document}
```

This produces the date in the form: 20th January, 2016

This setting is the default for datetime2. However, if it showdow has been switched on, it can later be switched off using

```
\DTMsetup{showdow=false}
```

hhmmss This option was designed for time formats in the style $\langle HH \rangle : \langle MM \rangle : \langle SS \rangle$. For example:

```
\documentclass{article}
\usepackage[hhmmss]{datetime}

\begin{document}
\currenttime
\end{document}
```

This produces the time in the form: 17:28:52.

This is the default for `datetime2`:

```
\documentclass{article}
\usepackage{datetime2}

\begin{document}
\DTMcurrenttime
\end{document}
```

24hr This option was designed for 24 hour time formats in the style $\langle HH \rangle : \langle MM \rangle$. For example:

```
\documentclass{article}
\usepackage[24hr]{datetime}

\begin{document}
\currenttime
\end{document}
```

This produces the time in the form: 17:28.

This can be achieved using the default `datetime2` time style with the seconds suppressed:

```
\documentclass{article}
\usepackage[showseconds=false]{datetime2}

\begin{document}
\DTMcurrenttime
\end{document}
```

12hr This option was designed for 12 hour time formats with “am” or “pm” suffixes. For example:

```
\documentclass{article}
\usepackage[12hr]{datetime}

\begin{document}
\currenttime
\end{document}
```

This produces the time in the form: 5:28pm.

This can be achieved through a `datetime2` language module that supports this format. For example, the `english` module:

```
\documentclass{article}
\usepackage[en-GB]{datetime2}

\begin{document}
\DTMcurrenttime
\end{document}
```

oclock This option was designed for a UK-style full text time. For example:

```
\documentclass{article}
\usepackage[oclock]{datetime2}

\begin{document}
\currenttime
\end{document}
```

This produces the time in the form: Twenty minutes past Six in the afternoon

This can be changed to `datetime2` through the `datetime2-en-fulltext` package, which needs to be installed separately:

```
\documentclass{article}
\usepackage{datetime2-en-fulltext}
\DTMsettimestyle{en-FullText}
\begin{document}
\DTMcurrenttime
\end{document}
```

The `datetime` package option `nodate` was provided before multilingual support was added to allow `babel` users to use the time commands without causing a conflict with the date. Once support for the `babel` package was added, this option became superfluous.

10.2 Time and Date Commands

```
\today
```

This is a robust command in later versions of `datetime`. In earlier versions it was fragile and had to be protected when used in moving arguments. With `datetime2`, this command is designed to be expandable and so therefore is not robust and shouldn't need protecting. (The date styles should take care of any fragile commands, such as `\textsuperscript`, where necessary.)

The current time is displayed with `datetime` using

`datetime`

```
\currenttime
```

The `datetime2` equivalent is:

`datetime2`

```
\DTMcurrenttime
```

Again, the command provided by `datetime` is robust and the equivalent command provided by `datetime2` is designed to be expandable.

A specific date is display by `datetime`'s

`datetime`

```
\formatdate{<DD>}{<MM>}{<YYYY>}
```

command. The arguments are in little-endian (UK) order with the day, month and year. With `datetime2` you can use either:

`datetime2`

```
\DTMdisplay{<YYYY>}{<MM>}{<DD>}{<dow>}
```

(expandable) or

`datetime2`

```
\DTMdate{<YYYY>-<MM>-<DD>}
```

(robust) where `<YYYY>` is the year, `<MM>` is the month number, `<DD>` is the day of month number and `<dow>` is the day of week number (starting from 0 for Monday) or -1 to disregard it.

A specific time is displayed by `datetime`'s

`datetime`

```
\formattime{<hh>}{<mm>}{<ss>}
```

command, which has three arguments: the hour (24) the minutes past the hour and the seconds past the minute. With `datetime2` you can use either:

`datetime2`

```
\DTMdisplaytime{<hh>}{<mm>}{<ss>}
```

(expandable) or

datetime2

```
\DTMtime{<hh>:<mm>:<ss>}
```

(robust) where $\langle hh \rangle$ is the hour, $\langle mm \rangle$ is the minutes and $\langle ss \rangle$ is the seconds.
The date separator used by the predefined datetime styles is given by

datetime

```
\dateseparator
```

which needs to be redefined if required. With datetime2, the date separator for the base numeric styles (except the fixed iso style) can be changed through the datesep package option. For example:

```
\usepackage[datesep={.}]{datetime2}
```

or

```
\DTMsetup{datesep={.}}
```

Some of the language modules may also provide a similar option. For example:

```
\usepackage[en-GB]{datetime2}  
\DTMLangsetup[en-GB]{datesep={.}}
```

The time separator for datetime is given by

datetime

```
\timeseparator
```

With datetime2, the time separator for the basic numeric styles (not including the fixed iso style) can be changed through the timesep package option. For example:

```
\usepackage[timesep={.}]{datetime2}
```

or

```
\usepackage{datetime2}  
\DTMsetup{timesep={.}}
```

Some of the language modules may also provide a similar option. For example:

```
\usepackage[en-GB]{datetime2}  
\DTMLangsetup[en-GB]{timesep={.}}
```

datetime

```
\pdfdate
```

The `\pdfdate` command provided by datetime for use within `\pdfinfo` became redundant with the introduction of `\pdfcreationdate` to PDF_T_EX version 1.30.0.

Old style (using datetime):

```
\pdfinfo{
  /Author (Me)
  /Title (A Sample Document)
  /CreationDate (D:20040501215500)
  /ModDate (D:\pdfdate)
}
```

New style (simply using PDF_T_EX):

```
\pdfinfo{
  /Author (Me)
  /Title (A Sample Document)
  /CreationDate (D:20040501215500)
  /ModDate (\pdfcreationdate)
}
```

Alternatively you can use the pdf style:

```
\DTMsetstyle{pdf}
\pdfinfo{
  /Author (Me)
  /Title (A Sample Document)
  /CreationDate (\DTMdisplay{2004}{05}{01}{-1}{21}{55}{00}{00}{00})
  /ModDate (\DTMnow)
}
```

The datetime command to display the month name is

datetime

```
\monthname[⟨n⟩]
```

which can't be expanded. With datetime2, the month name for a specific language can be obtained from a command provided by the relevant module. For example, the `english` module provides

datetime2-english

```
\DTMenglishmonthname{⟨n⟩}
```

which is expandable. These types of commands are designed for use within language-dependent date styles. This ensures that the name matches the style. (One of the failings of datetime was that the original date styles provided when the package was originally only intended for British dates produced weird hybrid styles when multilingual support was later added and styles such as long were used with another language.) These types of commands provided by the datetime2 language modules are analogous to the `\monthname⟨language⟩` commands provided by datetime's supporting language files (for example,

datetime-defaults

```
\monthnameenglish[⟨n⟩]
```

in datetime-defaults or

dt-french.def

```
\monthnamefrench[⟨n⟩]
```

defined in dt-french.def) except that the datetime commands have an optional argument which means they're not expandable.

If you load the datetime2-calc package, either explicitly or through the calc or showdoc datetime2 package options, then pgfcalendar will also be loaded. In which case you can use

pgfcalendar

```
\pgfcalendarmonthname{⟨n⟩}
```

even if none of the datetime2 language modules have been loaded. This command requires the translator package to provide multilingual support. See the pgf manual for further details.

Another possibility if you want the month name alone using the current language is to use the robust command

datetime2-calc

```
\DTMmonthname{⟨n⟩}
```

defined by datetime2-calc. This is the closest match to datetime's `\monthname` command but note that the argument isn't optional. Remember that you can use `\month` for the current month number, which is the value of `⟨n⟩` when omitted in `\monthname`.

Old style (datetime):

```
\documentclass{article}

\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[french,english]{babel}

\usepackage{datetime}

\begin{document}
\selectlanguage{french}
\monthname

\selectlanguage{english}
\monthname
\end{document}
```

New style (datetime2 with the `french` and `english` modules):


```

\documentclass{article}

\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[french,english]{babel}

\usepackage[calc]{datetime2}

\begin{document}
\selectlanguage{french}
\DTMmonthname{\month}

\selectlanguage{english}
\DTMmonthname{\month}
\end{document}

```

The abbreviated month name is given by

datetime

```
\shortmonthname[⟨n⟩]
```

in datetime. Similar to above, language modules may provide an expandable command to produce the abbreviated name in that specific language. For example, the `english` module provides

datetime2-english

```
\DTMenglishshortmonthname{⟨n⟩}
```

As with `\DTMenglishmonthname`, this is designed for use in English date styles.

Again, if `datetime2-calc` is loaded, the `pgfcalendar` command is also available:

pgfcalendar

```
\pgfcalendarmonthshortname{⟨n⟩}
```

Alternative you can use the robust command defined by `datetime2-calc`:

datetime2-calc

```
\DTMshortmonthname{⟨n⟩}
```

Old style (datetime):

```

\documentclass{article}

\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[french,english]{babel}

```

```

\usepackage{datetime}

\begin{document}
\selectlanguage{french}
\shortmonthname

\selectlanguage{english}
\shortmonthname
\end{document}

```

New style (datetime2 with the `french` and `english` modules):

```

\documentclass{article}

\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[french,english]{babel}

\usepackage[calc]{datetime2}

\begin{document}
\selectlanguage{french}
\DTMshortmonthname{\month}

\selectlanguage{english}
\DTMshortmonthname{\month}
\end{document}

```

See Section 9 and Section 7 for further details.

The weekday names in `datetime` are more complicated and not all the `dt-⟨lang⟩.def` files provide translations. The ones that do support the weekday provide

`dt-⟨lang⟩.def`

```
\dayofweeknameid⟨lang⟩{⟨n⟩}
```

which takes a single argument that's an integer from 1 (Sunday) to 7 (Saturday). For example,

`datetime-defaults`

```
\dayofweeknameidenglish{⟨n⟩}
```

or

`dt-french.def`

```
\dayofweeknameidfrench{⟨n⟩}
```

The `datetime2` language modules that support the weekday name provide

`datetime2-⟨lang⟩`

```
\DTM⟨lang⟩weekdayname{⟨dow⟩}
```

which takes a single argument that's an integer from 0 (Monday) to 6 (Sunday). As with `date-time`, not all of the `datetime2` language modules support the weekday.

Old style (`datetime`):

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[french,english]{babel}
\usepackage{datetime}

\begin{document}
\dayofweeknameidfrench{1}

\dayofweeknameidenglish{1}
\end{document}
```

New style (`datetime2` with the `english` and `french` modules):

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[french,english]{babel}
\usepackage{datetime2}

\begin{document}
\DTMfrenchweekdayname{6}

\DTMenglishweekdayname{6}
\end{document}
```

As with the month name, date styles should explicitly use the weekday macro (if provided) for the specific language to display the weekday name rather than using a macro that varies according to the current language.

The `datetime` package provides

`datetime`

```
\dayofweeknameid{⟨n⟩}
```

to provide the weekday name for the current language. If the current language doesn't provide a translation for the weekday names, then the English names are used. This command basically attempts to use `\dayofweeknameid⟨lang⟩` (where `⟨lang⟩` is given by `\languagename`) if it exists otherwise it uses `\dayofweeknameidenglish`. This language-sensitive macro is a fragile command that requires protection in moving arguments.

As before, if the `datetime2-calc` package is loaded, the `pgfcalendar` package's commands are also available including

`pgfcalendar`

```
\pgfcalendarweekdayname{⟨dow⟩}
```

where the argument $\langle dow \rangle$ is an integer from 0 (Monday) to 6 (Sunday). Multilingual support is provided through the translator package.

The `datetime2-calc` package also provides a robust language-sensitive command:

`datetime2-calc`

```
\DTMweekdayname{ $\langle dow \rangle$ }
```

This attempts to use `\DTM $\langle lang \rangle$ weekdayname` if it exists, where $\langle lang \rangle$ is either `\language` or obtained from the dialect-to-language mapping provided by `tracklang`. If both attempts fail, `\DTMweekdayname` will fallback on `\pgfcalendarweekdayname` (with a warning through `\dtmnamewarning`).

Old style (`datetime`):

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[french,english]{babel}

\usepackage{datetime}

\begin{document}

\selectlanguage{french}
\dayofweeknameid{1}

\selectlanguage{english}
\dayofweeknameid{1}
\end{document}
```

New style (`datetime2` with the `english` and `french` modules):

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[french,english]{babel}

\usepackage[calc]{datetime2}

\begin{document}

\selectlanguage{french}
\DTMweekdayname{6}

\selectlanguage{english}
\DTMweekdayname{6}
\end{document}
```

The `datetime` package also provides the command

datetime

```
\dayofweekname{<DD>}{<MM>}{<YYYY>}
```

that has three arguments: the day of the month, the month number and the year. This calculates the day of week index $\langle n \rangle$, an integer from 1 (Sunday) to 7 (Saturday), and then uses the result in the argument of `\dayofweeknameid{<n>}`. The `\dayofweekname` command was provided by `datetime` as a convenient shortcut for use in date styles. (Similarly for `\shortdayofweekname`.) With the `datetime2` date styles, the day-of-week index is automatically available through the fourth argument of `\DTMdisplaydate` (and `\DTMDisplaydate`), so there is little use for an equivalent command. Additionally, if a date has already been saved with `datetime2`, the weekday can be extracted from the saved data through `\DTMfetchdow`, which can be used in the argument of commands like `\DTMweekdayname` or `\DTMenglishweekdayname`.

However, if a calculation is required for some reason, it can be obtained using the `pgfcalendar` commands `\pgfcalendardatetojulian` and `\pgfcalendarjuliantoweekday`, which are described in the `pgf` manual.

The `datetime` package provides the conditional

datetime

```
\ifshowdow
```

which can be used to determine whether or not styles should display the weekday name (if supported). The `datetime2` package has the analogous conditional

datetime2

```
\ifDTMshowdow
```

The `datetime` package provides the command

datetime

```
\ordinaldate{<n>}
```

as a date-type ordinal where the argument should be an integer from 1 to 31. For English, this just uses `fmtcount`'s non-expandable `\ordinalnum` command. For Breton, Welsh and French a suffix is only added when the argument is 1. For all other languages, this command just displays the number.

With `datetime2`, the language modules may or may not provide a command to display the ordinal but most of them do. For example, the `english` module provides

datetime2-english

```
\DTMenglishordinal{<n>}
```

This displays the suffix using

datetime2-english

```
\DTMenglishfmtordsuffix{<suffix>}
```

The definition of this command is changed by the styles provided by the English regional modules. For example, the en-US style redefines `\DTMenglishfmtordsuffix` to ignore its argument. See the documentation for the `english` module for further details.

The `french` module provides:

datetime2-french

```
\DTMfrenchordinal{<n>}
```

This displays `<n>` and if `<n>` is 1, a suffix is appended. See the `french` module documentation for further details.

The `breton` module provides:

datetime2-breton

```
\DTMbretonordinal{<n>}
```

which similarly appends a suffix if `<n>` is 1 but not for other values. In this case, the suffix for the first day is formatted using

datetime2-breton

```
\DTMbretonfmtordinal{<suffix>}
```

which is redefined by the `ord` option. See the `breton` module documentation for further details.

The `welsh` module similarly provides:

datetime2-welsh

```
\DTMwelshordinal{<n>}
```

and

datetime2-welsh

```
\DTMwelshfmtordinal{<suffix>}
```

See the `welsh` module documentation for further details.

Other modules may simply define `\DTM<lang>ordinal` to just display its argument. For example, the `german` module provides

datetime2-german

```
\DTMgermanordinal{<n>}
```

which just displays `<n>` (the day of month number).

Alternatively, modules may define `\DTM⟨lang⟩ordinal` to display its argument followed by a full stop (period). For example, the `norsk` module provides

`datetime2-norsk`

```
\DTMnorskordinal{⟨n⟩}
```

which displays $\langle n \rangle$ followed by a full stop.

As before, the date styles should explicitly use the ordinal macro that matches the style. However, if you have some need to display the day of month independent of any of the styles, you can use

`datetime2-calc`

```
\DTMordinal{⟨n⟩}
```

which is provided by `datetime2-calc`. This attempts to use `\DTM⟨lang⟩ordinal` if it exists, otherwise it just displays $\langle n \rangle$.

Old style (`datetime`):

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[french,english]{babel}

\usepackage{datetime}

\begin{document}

\selectlanguage{french}
\ordinaldate{1}

\selectlanguage{english}
\ordinaldate{1}
\end{document}
```

This displays 1^{er} in the first case (through `\ordinaldatefrench`) and 1st in the second case (through `fmtcount`'s `\ordinalnum`).

New style (`datetime2` with the `english` and `french` modules):

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[french,english]{babel}

\usepackage[calc]{datetime2}

\begin{document}

\selectlanguage{french}
```

```

\DTMordinal{1}

\selectlanguage{english}
\DTMordinal{1}
\end{document}

```

This displays 1^{er} in the first case and just 1 in the second case, because the regionless `english` module doesn't use a suffix.

To achieve the same result as the `datetime` example, a few modifications are needed:

```

\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[french,british]{babel}

\usepackage[calc,useregional]{datetime2}
\DTMlangsetup[en-GB]{ord=raise}

\begin{document}

\selectlanguage{french}
\DTMordinal{1}

\selectlanguage{british}
\DTMordinal{1}
\end{document}

```

The `datetime` package provides the command

`datetime`

```
\twodigit{<n>}
```

for use in date styles that require two-digit numbers. The `datetime2` package provides

`datetime2`

```
\DTMtwodigits{<n>}
```

10.3 Saving Dates

The `datetime` package provides some commands for saving a date for later use. (There are no equivalent commands for saving a time in `datetime`.) With `datetime`, a date is saved using

`datetime`

```
\newdate{<name>}{<DD>}{<MM>}{<YYYY>}
```

With `datetime2`, a date can be saved using

datetime2

```
\DTMsavedate{<name>}{<date>}
```

where *<date>* is in the form *<YYYY>-<MM>-<DD>*.

A previously saved date can be displayed with datetime using:

datetime

```
\displaydate{<name>}
```

With datetime2, the saved date can be displayed using:

datetime2

```
\DTMusedate{<name>}
```

Old style (datetime):

```
\documentclass{article}
\usepackage{datetime}

\begin{document}

\newdate{mydate}{20}{1}{2016}
\displaydate{mydate}

\end{document}
```

New style (datetime2):

```
\documentclass{article}
\usepackage[en-GB,showdow]{datetime2}

\begin{document}

\DTMsavedate{mydate}{2016-01-20}
\DTMusedate{mydate}

\end{document}
```

Individual elements of the date can be extracted with the datetime commands:

datetime

```
\getdateday{<name>}
```

for the day of the month,

datetime

```
\getdatemonth{<name>}
```

for the month number, and

datetime

```
\getdateyear{<name>}
```

for the year.

These elements can be fetched in datetime2 using:

datetime2

```
\DTMfetchday{<name>}
```

for the day of the month,

datetime2

```
\DTMfetchmonth{<name>}
```

for the month number, and

datetime2

```
\DTMfetchyear{<name>}
```

for the year. Additionally you can fetch the day of week index if it has been computed:

datetime2

```
\DTMfetchdow{<name>}
```

Old style (datetime):

```
\documentclass{article}
```

```
\usepackage{datetime}
```

```
\begin{document}
```

```
\newdate{mydate}{20}{1}{2016}
```

```
Year: \getdateyear{mydate}.
```

```
Month: \getdatemonth{mydate}.
```

```
Day: \getdateday{mydate}.
```

```
\end{document}
```

New style (datetime2):

```
\documentclass{article}
```

```
\usepackage{datetime2}
```

```

\begin{document}

\DTMsavedate{mydate}{2016-01-20}

Year: \DTMfetchyear{mydate}.
Month: \DTMfetchmonth{mydate}.
Day: \DTMfetchday{mydate}.

\end{document}

```

With `datetime2`, you can also save the current time (as in the time of the document build) with

`datetime2`

```
\DTMsavenow{<name>}
```

and then access each field of the date, time and zone.

Old style (`datetime`):

```

\documentclass{article}

\usepackage{datetime}

\begin{document}

Year: \number\year.
Month: \number\month.
Day: \number\day.
DOW: \computedayofweek{\day}{\month}{\year}\number\dayofweek.

Hour: \number\currenthour.
Minute: \number\currentminute.
Second: \number\currentsecond.

\end{document}

```

New style (`datetime2`):

```

\documentclass{article}

\usepackage[calc]{datetime2}

\begin{document}
\DTMsavenow{now}

Year: \DTMfetchyear{now}.
Month: \DTMfetchmonth{now}.
Day: \DTMfetchday{now}.
DOW: \DTMfetchdow{now}.

```

```

Hour: \DTMfetchhour{now}.
Minute: \DTMfetchminute{now}.
Second: \DTMfetchsecond{now}.

Time Zone Hour: \DTMfetchTZhour{now}.
Time Zone Minute: \DTMfetchTZminute{now}.

\end{document}

```

(Note that the day of week index is different as `datetime2` uses the same indexing system as `pgfcalendar`.)

10.4 Multilingual Support

The `datetime` package comes with the following files (in addition to `datetime.sty` and `datetime-defaults.sty`):

<code>dt-american.def</code>	<code>dt-dutch.def</code>	<code>dt-lsorbian.def</code>	<code>dt-slovak.def</code>
<code>dt-australian.def</code>	<code>dt-esperanto.def</code>	<code>dt-magyar.def</code>	<code>dt-slovene.def</code>
<code>dt-austrian.def</code>	<code>dt-estonian.def</code>	<code>dt-naustrian.def</code>	<code>dt-spanish.def</code>
<code>dt-bahasa.def</code>	<code>dt-finnish.def</code>	<code>dt-newzealand.def</code>	<code>dt-swedish.def</code>
<code>dt-basque.def</code>	<code>dt-french.def</code>	<code>dt-ngerman.def</code>	<code>dt-turkish.def</code>
<code>dt-breton.def</code>	<code>dt-galician.def</code>	<code>dt-norsk.def</code>	<code>dt-UKenglish.def</code>
<code>dt-british.def</code>	<code>dt-german.def</code>	<code>dt-polish.def</code>	<code>dt-ukraineb.def</code>
<code>dt-bulgarian.def</code>	<code>dt-greek.def</code>	<code>dt-portuges.def</code>	<code>dt-USenglish.def</code>
<code>dt-canadian.def</code>	<code>dt-hebrew.def</code>	<code>dt-romanian.def</code>	<code>dt-usorbian.def</code>
<code>dt-catalan.def</code>	<code>dt-icelandic.def</code>	<code>dt-russian.def</code>	<code>dt-welsh.def</code>
<code>dt-croatian.def</code>	<code>dt-irish.def</code>	<code>dt-samin.def</code>	
<code>dt-czech.def</code>	<code>dt-italian.def</code>	<code>dt-scottish.def</code>	
<code>dt-danish.def</code>	<code>dt-latin.def</code>	<code>dt-serbian.def</code>	

These `dt-⟨lang⟩.def` files provide the code to integrate `datetime` with `babel` for each language given by `⟨lang⟩`. This means that if you have `datetime` installed and there's a `.def` file that matches the language you are using with `babel`, then all you need to do is load `babel` before `datetime`.

With `datetime2`, language support is provided in separate independently-maintained modules. The actual `datetime2` package itself just comes with two files: `datetime2.sty` and `datetime2-calc.sty`. This means that if you only want to use the basic numeric styles and aren't using `babel` or `polyglossia`, then that's all you need. (Although you'll obviously need to install dependent packages, such as `pgf` which provides the `pgfcalendar` package used by `datetime2-calc`. However `fntcount`, which is required by `datetime`, isn't required by `datetime2`.)

If you want to use `datetime2` with language support, then you only need to install the modules for the required language. For example, if you only use English, you can just install the `english` module and if you only use French, you can just install the `french` module.

At the time of writing, the following modules are available on CTAN:

<code>datetime2-bahasai</code>	<code>datetime2-galician</code>	<code>datetime2-russian</code>
<code>datetime2-basque</code>	<code>datetime2-german</code>	<code>datetime2-samin</code>
<code>datetime2-breton</code>	<code>datetime2-greek</code>	<code>datetime2-scottish</code>
<code>datetime2-bulgarian</code>	<code>datetime2-hebrew</code>	<code>datetime2-serbian</code>
<code>datetime2-catalan</code>	<code>datetime2-icelandic</code>	<code>datetime2-slovak</code>
<code>datetime2-croatian</code>	<code>datetime2-irish</code>	<code>datetime2-slovene</code>
<code>datetime2-czech</code>	<code>datetime2-italian</code>	<code>datetime2-spanish</code>
<code>datetime2-danish</code>	<code>datetime2-latin</code>	<code>datetime2-swedish</code>
<code>datetime2-dutch</code>	<code>datetime2-lsorbian</code>	<code>datetime2-turkish</code>
<code>datetime2-english</code>	<code>datetime2-magyar</code>	<code>datetime2-ukrainian</code>
<code>datetime2-esperanto</code>	<code>datetime2-norsk</code>	<code>datetime2-usorbian</code>
<code>datetime2-estonian</code>	<code>datetime2-polish</code>	<code>datetime2-welsh</code>
<code>datetime2-finnish</code>	<code>datetime2-portuges</code>	
<code>datetime2-french</code>	<code>datetime2-romanian</code>	

Some of these only support the root language but some, such as `english`, provide support for different regions. There is also a supplementary package `datetime2-en-fulltext` that replicates `datetime`'s text and `oclock` styles (and requires `fmtcount`).

Old style (`datetime`):

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[french,british]{babel}

\usepackage{datetime}

\begin{document}
\selectlanguage{french}
\today

\selectlanguage{british}
\today

\end{document}
```

This produces:

21 janvier 2016

Thursday 21st January, 2016

New style (`datetime2`, `datetime2-english` and `datetime2-french`):

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
```

```

\usepackage[french,british]{babel}

\usepackage[showdoc,useregional]{datetime2}

\begin{document}
\selectlanguage{french}
\today

\selectlanguage{british}
\today

\end{document}

```

This produces:

```

21 janvier 2016
Thursday 21st January 2016

```

There's a slight difference in the appearance of the British date. An exact reproduction of the datetime format can be achieved by modifying the en-GB options:

```

\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[french,british]{babel}

\usepackage[showdoc,useregional]{datetime2}
\DTMLangsetup[en-GB]{ord=raise,monthyearsep={,\space}}

\begin{document}
\selectlanguage{french}
\today

\selectlanguage{british}
\today

\end{document}

```

Note that neither dt-french.def from datetime nor datetime2-french support the show day of week option.

The datetime package provides:

datetime

```
\setdefaultdate{<date declaration>}
```

to always use the date style given by *<date declaration>* instead of letting babel switch the date format every time the language changes.

In datetime2, the default is the reverse: the style won't change when the language changes unless the languages (or regions) have been listed in the datetime2 package options. If the

regional styles have been enabled, allowing babel to change the date style whenever the language changes, then you can switch this behaviour off by setting the `useregional` option to `false`.

Old style (datetime):

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[french,british]{babel}

\usepackage{datetime}

\yyymmdddate

\begin{document}
\selectlanguage{french}
\today

\selectlanguage{british}
\today

\end{document}
```

This produces:

21 janvier 2016

Thursday 21st January, 2016

(The numeric date style has been overridden.)

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[french,british]{babel}

\usepackage{datetime2}

\DTMsetdatestyle{iso}

\begin{document}
\selectlanguage{french}
\today

\selectlanguage{british}
\today

\end{document}
```

This produces:

2016-01-21

2016-01-21

(The ISO date style overrides the language setting.)

Compare this to:

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[french,british]{babel}

\usepackage[en-GB,fr-FR]{datetime2}

\DTMsetdatestyle{iso}

\begin{document}
\selectlanguage{french}
\today

\selectlanguage{british}
\today

\end{document}
```

This produces:

21 janvier 2016

21st January 2016

Examples that explicitly suppress the language-sensitive dates follow. First with datetime:

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[french,british]{babel}

\usepackage{datetime}

\setdefaultdate{\yyyymmdddate}

\begin{document}
\selectlanguage{french}
\today

\selectlanguage{british}
\today

\end{document}
```

This produces:

2016/01/21

2016/01/21

Now with `datetime2`:

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[french,british]{babel}

\usepackage[en-GB,fr-FR,useregional=false]{datetime2}

\DTMsetdatestyle{iso}

\begin{document}
\selectlanguage{french}
\today

\selectlanguage{british}
\today

\end{document}
```

This produces:

2016-01-21

2016-01-21

10.5 Predefined Date Formats

The `datetime` date styles were set with declarations such as `\longdate` (or package options that used the associated declaration, such as `long`). With `datetime2`, date styles are set using

`datetime2`

```
\DTMsetdatestyle{<style name>}
```

(which just changes the date style without changing the time style) or

`datetime2`

```
\DTMsetstyle{<style name>}
```

which sets the full date-time style.

This section lists the `datetime` declarations and how the same style can be set through `datetime2`.

`datetime`

```
\yyyymmdddate
```

By default this style produces a date in the form 2016/01/20. (The separator is governed by `\dateseparator`.) This is the default style for `datetime2` with the exception of the separator, which defaults to a hyphen. To reproduce the `datetime` format, you can set the date style to `default` (if it has been previously changed from the default) and change the separator with the `datesep` option:

```
\DTMsetdatestyle{default}  
\DTMsetup{datesep={/}}
```

`datetime`

```
\longdate
```

This is the default date format for `datetime` and this style produces the date in the form: Wednesday 8th March, 2000. This is actually a regional style for some of the English dialects, so with `datetime2` this additionally needs the `english` module installed. To exactly replicate this date format, including the day of week name, the superscript ordinal suffix and the comma after the month name, you need the `showdown` and `en-GB` package options and the `ord` and `monthyearsep` options for the `en-GB` style. For example:

```
\usepackage[showdown,en-GB]{datetime2}  
\DTMlangsetup[en-GB]{ord=raise,monthyearsep={,\space}}
```

Another possibility is:

```
\usepackage[british]{babel}  
\usepackage[showdown]{datetime2}  
\DTMlangsetup[en-GB]{ord=raise,monthyearsep={,\space}}  
\DTMsetdatestyle{en-GB}
```

There are other regional styles in the `english` module that produce the same format, such as `en-GG`.

`datetime`

```
\shortdate
```

This is similar to `\longdate` but uses abbreviated names to produce a date in the form: Wed 8th Mar, 2000. With `datetime2`, this is like the above but additionally needs the `abbr` option for the `en-GB` style:

```
\usepackage[showdown,en-GB]{datetime2}  
\DTMlangsetup[en-GB]{ord=raise,monthyearsep={,\space},abbr}
```

or

```
\usepackage[british]{babel}  
\usepackage[showdown]{datetime2}  
\DTMlangsetup[en-GB]{ord=raise,monthyearsep={,\space},abbr}  
\DTMsetdatestyle{en-GB}
```

datetime

```
\ddmmyyyydate
```

This produces a date in the form 08/03/2000. This can be reproduced with just datetime2 using the ddmmyyyy style with the date separator set to a slash:

```
\DTMsetdatestyle{ddmmyyyy}  
\DTMsetup{datesep=/}
```

datetime

```
\dmyyyydate
```

This produces a date in the form 8/3/2000. This can be reproduced with just datetime2 using the dmyyyy style with the date separator set to a slash:

```
\DTMsetdatestyle{dmyyyy}  
\DTMsetup{datesep=/}
```

This format is also the style of some of the regional numeric date styles. For example, with the [english](#) module:

```
\usepackage[british]{babel}  
\usepackage{datetime2}  
\DTMsetdatestyle{en-GB-numeric}
```

datetime

```
\ddmmydate
```

This produces a date in the form 08/03/00. This can be reproduced with just datetime2 using the ddmyyy style with the date separator set to a slash:

```
\DTMsetdatestyle{ddmmyy}  
\DTMsetup{datesep=/}
```

datetime

```
\dmyydate
```

This produces a date in the form 8/3/00. This can be reproduced with just datetime2 using the dmyy style with the date separator set to a slash:

```
\DTMsetdatestyle{dmyy}  
\DTMsetup{datesep=/}
```

datetime

```
\textdate
```

This style is designed to produce a full text British date in the form: Wednesday the Eighth of March, Two Thousand. The `datetime2-en-fulltext` package is required to reproduce this style:

```
\usepackage[showdow]{datetime2-en-fulltext}
\DTMsetdatestyle{en-FullText}
```

Note that with both `datetime` and `datetime2-en-fulltext` this style should not be used if the current language isn't English.

`datetime`

```
\usdate
```

This style is designed to produce TeX's default US date format in the form March 8, 2000. This style can be reproduced with `datetime2` and the `english` module:

```
\usepackage[en-US]{datetime2}
```

or

```
\usepackage[USenglish]{babel}
\usepackage[userregional]{datetime2}
\DTMsetdatestyle{en-US}
```

`datetime`

```
\mddyyydate
```

This style produces a date in a middle-endian format in the form: 03/08/2000. This style can be reproduced with `datetime2` using the `mddyyy` style with the date separator set to a slash:

```
\DTMsetup{datesep=/}
\DTMsetdatestyle{mddyyy}
```

`datetime`

```
\mdyyydate
```

This style produces a date in a middle-endian format in the form: 3/8/2000. This style can be reproduced with `datetime2` using the `mdyyy` style with the date separator set to a slash:

```
\DTMsetup{datesep=/}
\DTMsetdatestyle{mdyyy}
```

This format is also the style of some of the regional numeric date styles. For example, with the `english` module:

```
\usepackage[USenglish]{babel}
\usepackage{datetime2}
\DTMsetdatestyle{en-US-numeric}
```

datetime

```
\mddydate
```

This style produces a date in a middle-endian format in the form: 03/08/00. This style can be reproduced with datetime2 using the mddy style with the date separator set to a slash:

```
\DTMsetup{datesep=/}  
\DTMsetdatestyle{mddy}
```

datetime

```
\mdyydate
```

This style produces a date in a middle-endian format in the form: 3/8/00. This style can be reproduced with datetime2 using the mdyy style with the date separator set to a slash:

```
\DTMsetup{datesep=/}  
\DTMsetdatestyle{mdyy}
```

10.6 Predefined Time Formats

The time formats are set in datetime using

datetime

```
\settimeformat{<style-name>}
```

With datetime2, the time styles are set using

datetime2

```
\DTMsettimestyle{<style-name>}
```

(which just changes the time style without changing the date style) or

datetime2

```
\DTMsetstyle{<style name>}
```

which sets the full date-time style.

The datetime package provides the following time styles:

xxivtime This style produces the time in twenty-four hour format in the form 09:28. (The default time style for datetime.) Note that this style has no seconds. The format can be reproduced in datetime2 with the default style and the showseconds option set to false:

```
\DTMsettimestyle{default}  
\DTMsetup{showseconds=false}
```

hhmmsstime This is like the previous style but includes the seconds. This is the default time style for `datetime2`:

```
\DTMsettimestyle{default}
```

ampmtime This style produces the time in twelve hour format in the form 9:28am or 7:54pm. This style is available in some of the regional modules for `datetime2`. For example, using the `en-GB` style in the `english` module:

```
\usepackage[en-GB]{datetime2}
```

or

```
\usepackage[british]{babel}  
\usepackage{datetime2}  
\DTMsettimestyle{en-GB}
```

oclock This style is designed for a full text English time format in the form: Twenty-Eight minutes past Ten in the afternoon. This style can be reproduced with the `en-FullText` style provided by the `datetime2-en-fulltext` package:

```
\usepackage{datetime2-en-fulltext}  
\DTMsettimestyle{en-FullText}
```

10.7 Defining a New Date Format

The `datetime` package provides:

`datetime`

```
\newdateformat{<name>}{<format>}
```

to define a new date style. Within *format*, the placeholder commands `\THEDAY`, `\THEMONTH` and `\THEYEAR` are used to represent the relevant day, month and year values. There are also counter placeholders `DAY`, `MONTH` and `YEAR`, which may be used instead.

A necessary consequence of allowing placeholder commands in *format* means that these commands must be set as appropriate before the date can be formatted. This means that the formatted date can't be expanded and the date commands must be made robust to protect them in moving arguments. This is an inherent problem with the `datetime` package that can't be fixed without breaking backwards compatibility and is one of the main reasons for introducing the replacement `datetime2` package.

The `datetime2` package provides a better way of providing date styles, which are defined using:

`datetime2`

```
\DTMnewdatestyle{<name>}{<definition>}
```

Instead of using placeholder commands, the date styles simply redefine the date formatting macros within *<definition>*. There are two principle date formatting commands that the date style must define although styles may redefine additional helper commands if necessary.

The two main date formatting commands are:

datetime2

```
\DTMdisplaydate{<YYYY>}{<MM>}{<DD>}{<dow>}
```

for use where no case-changing is required and

datetime2

```
\DTMDisplaydate{<YYYY>}{<MM>}{<DD>}{<dow>}
```

for use where the date must begin with an upper case letter, for example if the date occurs at the start of a sentence. For styles where the case-change is irrelevant (for example, numeric styles or styles that always start with an upper case letter), the `\DTMDisplaydate` command may simply be set to `\DTMdisplaydate`.

The datetime manual provides some examples of new date styles. The first is simply a numeric little-endian style with a hyphen separating each number:

```
\newdateformat{mydate}{\THEDAY-\THEMONTH-\THEYEAR}
```

To convert this into a datetime2 format, the new style simply needs to redefine `\DTMdisplaydate` so that it has the following definition:

```
\renewcommand{\DTMdisplaydate}[4]{#3-#2-#1}
```

Note that this command must always have four arguments, even if one or more of them are ignored. So here `\THEYEAR` is just the first argument #1, `\THEMONTH` is the second argument #2 and `\THEDAY` is the third argument #3. One other thing to note is that the arguments may be supplied with a leading zero. If you want to trim this off, you can use TeX's `\number` primitive. For example:

```
\renewcommand{\DTMdisplaydate}[4]{\number#3-\number#2-\number#1 }
```

This is the better method to allow for, say, registers used in any of the arguments.

Therefore this new date style can be defined for datetime2 as follows:

```
\DTMnewdatestyle{mydate}{%
  \renewcommand{\DTMdisplaydate}[4]{\number##3-\number##2-\number##1 }%
  \renewcommand{\DTMdisplaydate}{\DTMdisplaydate}%
}
```

(Note the need to double the # in the parameters, as is usual when redefining a command within another command in this manner.)

For example, using datetime:

```
\documentclass{article}
\usepackage{datetime}
```

```
\newdateformat{mydate}{\THEDAY-\THEMONTH-\THEYEAR}
\mydate
```

```
\begin{document}
\today.
\end{document}
```

Now using datetime2:

```
\documentclass{article}
\usepackage{datetime2}

\DTMnewdatestyle{mydate}{%
  \renewcommand{\DTMdisplaydate}[4]{\number##3-\number##2-\number##1 }%
  \renewcommand{\DTMdisplaydate}{\DTMdisplaydate}%
}
\DTMsetdatestyle{mydate}

\begin{document}
\today.
\end{document}
```

Another example provided in the datetime manual ensures two digits for the month and day of month:

```
\newdateformat{dashdate}{\twodigit{\THEDAY}-\twodigit{\THEMONTH}-\THEYEAR}
```

This is a minor modification to the previous example. The two digit number format can be obtained through datetime2's `\DTMtwodigits` command:

```
\DTMnewdatestyle{dashdate}{%
  \renewcommand{\DTMdisplaydate}[4]{%
    \DTMtwodigits{##3}-\DTMtwodigits{##2}-\number##1 }%
  \renewcommand{\DTMdisplaydate}{\DTMdisplaydate}%
}
```

A complete example follows for datetime:

```
\documentclass{article}
\usepackage{datetime}

\newdateformat{dashdate}{\twodigit{\THEDAY}-\twodigit{\THEMONTH}-\THEYEAR}
\dashdate

\begin{document}
\today.
\end{document}
```

And for datetime2:

```
\documentclass{article}
```



```

\usepackage{datetime2}

\DTMnewdatestyle{dashdate}{%
  \renewcommand{\DTMdisplaydate}[4]{%
    \DTMtwodigits{##3}-\DTMtwodigits{##2}-\number##1 }%
  \renewcommand{\DTMDisplaydate}{\DTMdisplaydate}%
}
\DTMsetdatestyle{dashdate}

\begin{document}
\today.
\end{document}

```

You may have noticed that there's no equivalent of `datetime`'s counter placeholders, but there's no real need for them and any attempt at implementing them would return to the original problem of preventing an expandable date format. \TeX counters all internally use \TeX count registers and counter formatting commands have to access those registers to determine the counter value.

To illustrate this, the `datetime` manual provides the example:

```

\newdateformat{usvardate}{\monthname[\THEMONTH] \ordinal{DAY}, \THEYEAR}

```

This uses the `DAY` counter placeholder since `\ordinal` (provided by the `fmtcount` package) requires a counter name as the argument. However, `\ordinal` internally uses `\ordinalnum` with the internal count register as the argument, so the format could just as easily be defined as:

```

\newdateformat{usvardate}{\monthname[\THEMONTH] \ordinalnum{\THEDAY}, \THEYEAR}

```

So how can this style be reproduced with `datetime2`? A naïve approach is:

```

\DTMnewdatestyle{usvardate}{%
  \renewcommand{\DTMdisplaydate}[4]{%
    \DTMmonthname{##2} \ordinalnum{##2}, \number##1 }%
  \renewcommand{\DTMDisplaydate}{\DTMdisplaydate}%
}

```

This requires both `datetime2-calc` (for `\DTMmonthname`) and `fmtcount` (for `\ordinalnum`).

A complete example that uses it:

```

\documentclass{article}

\usepackage{fmtcount}
\usepackage[calc]{datetime2}

\DTMnewdatestyle{usvardate}{%
  \renewcommand{\DTMdisplaydate}[4]{%
    \DTMmonthname{##2} \ordinalnum{##2}, \number##1 }%
  \renewcommand{\DTMDisplaydate}{\DTMdisplaydate}%
}

```

```
\DTMsetdatestyle{usvardate}
```

```
\begin{document}  
\today.  
\end{document}
```

This produces a warning from `\DTMmonthname` because `\DTMenglishmonthname` hasn't been defined, so it uses `\pgfcalendarmonthname` instead. The document displays the text:

January 1st, 2016.

This seems to produce the correct output, but let's see what happens if we make a minor modification to the example:

```
\documentclass{article}  
  
\usepackage[english,french]{babel}  
\usepackage{fmtcount}  
\usepackage[calc]{datetime2}  
  
\DTMnewdatestyle{usvardate}{%  
  \renewcommand{\DTMdisplaydate}[4]{%  
    \DTMmonthname{##2} \ordinalnum{##2}, \number##1 }%  
  \renewcommand{\DTMdisplaydate}{\DTMdisplaydate}%  
}  
\DTMsetdatestyle{usvardate}  
  
\begin{document}  
\today.  
\end{document}
```

This now produces:

janvier 1^{er}, 2016.

which doesn't match the rationale of the style being a variation of the standard US date style nor does it match the little-endian syntax of French dates.

Now let's make another minor change to the example:

```
\documentclass{article}  
  
\usepackage{fmtcount}  
\usepackage[calc]{datetime2}  
  
\DTMnewdatestyle{usvardate}{%  
  \renewcommand{\DTMdisplaydate}[4]{%  
    \DTMmonthname{##2} \ordinalnum{##2}, \number##1 }%  
  \renewcommand{\DTMdisplaydate}{\DTMdisplaydate}%  
}  
\DTMsetdatestyle{usvardate}
```

```

\begin{document}
\section{\today: an example}

\today.
\end{document}

```

This document can't compile properly and causes the error:

```

! Argument of \@sect has an extra }.
<inserted text>
      \par

```

This is because the style definition has made `\today` fragile because it uses an unprotected fragile command. This can be fixed by protecting `\ordinalnum` in the style definition.

Let's make another modification:

```

\documentclass{article}

\usepackage{fmtcount}
\usepackage[calc]{datetime2}

\DTMnewdatestyle{usvardate}{%
  \renewcommand{\DTMdisplaydate}[4]{%
    \DTMmonthname{##2} \protect\ordinalnum{##2}, \number##1 }%
  \renewcommand{\DTMdisplaydate}{\DTMdisplaydate}%
}
\DTMsetdatestyle{usvardate}

\pagestyle{headings}

\begin{document}
\section{\today: an example}

\today.
\end{document}

```

This compiles without error now that `\ordinalnum` has been protected, but the page header appears as:

1 January 1st, 2016: AN EXAMPLE 1

The date hasn't been rendered in upper case so the header doesn't look right.

If `hyperref` is added, another problem becomes evident:

```

\documentclass{article}

\usepackage{fmtcount}
\usepackage[calc]{datetime2}
\usepackage[colorlinks]{hyperref}

```

```

\DTMnewdatestyle{usvardate}{%
  \renewcommand{\DTMdisplaydate}[4]{%
    \DTMmonthname{##2} \protect\ordinalnum{##2}, \number##1 }%
  \renewcommand{\DTMdisplaydate}{\DTMdisplaydate}%
}
\DTMsetdatestyle{usvardate}

\pagestyle{headings}

\begin{document}
\section{\today: an example}

\today.
\end{document}

```

The PDF bookmarks for this document show the section title as:

01 , 2016: an example

Both the header and the bookmark problem are caused by non-expandable elements of the date style. *The same problems occur with datetime*, and is one of the main reasons for developing a replacement package since these problems can't be fixed by datetime. In fact, the datetime version of this example looks even worse in the bookmarks:

```

\documentclass{article}

\usepackage{datetime}
\usepackage[colorlinks]{hyperref}

\newdateformat{usvardate}{\monthname[\THEMONTH] \ordinalnum{\THEDAY}, \THEYEAR}
\usvardate

\pagestyle{headings}

\begin{document}
\section{\today: an example}

\today.
\end{document}

```

The bookmark now looks like:

===[0], 0: an example

(The page header is the same as for the datetime2 example above, with the date not matching the rest of the heading case.)

A more appropriate way of defining this style with datetime2 package is to use the *expandable* commands provided by the [english](#) module:

```

\documentclass{article}

```

```

\usepackage[calc,en-GB]{datetime2}
\usepackage[colorlinks]{hyperref}

\DTMnewdatestyle{usvardate}{%
  \renewcommand*{\DTMenglishfmtordsuffix}{\DTMenGBfmtordsuffix}%
  \renewcommand{\DTMdisplaydate}[4]{%
    \DTMenglishmonthname{##2} \DTMenglishordinal{##2}, \number##1 }%
  \renewcommand{\DTMdisplaydate}{\DTMdisplaydate}%
}
\DTMsetdatestyle{usvardate}

\pagestyle{headings}

\begin{document}
\section{\today: an example}

\today.
\end{document}

```

This fixes both the header and the bookmarks.

If you don't want to rely on remembering to use the en-GB option to load the en-GB style (which defines `\DTMenGBfmtordsuffix`) you can use this alternative:

```

\documentclass{article}

\usepackage[calc]{datetime2}
\usepackage[colorlinks]{hyperref}

\DTMusemodule{english}{en-GB}

\DTMnewdatestyle{usvardate}{%
  \renewcommand*{\DTMenglishfmtordsuffix}{\DTMenGBfmtordsuffix}%
  \renewcommand{\DTMdisplaydate}[4]{%
    \DTMenglishmonthname{##2} \DTMenglishordinal{##2}, \number##1 }%
  \renewcommand{\DTMdisplaydate}{\DTMdisplaydate}%
}
\DTMsetdatestyle{usvardate}

\pagestyle{headings}

\begin{document}
\section{\today: an example}

\today.
\end{document}

```

This is a useful method to employ if you want to define the new style in a package without forcing the package options used to load `datetime2`.

Note that this new style definition is unaffected by language changes, so the other problem with the `datetime` version of this style is also eliminated.

10.8 Defining a New Time Format

The `datetime` package provides:

`datetime`

```
\newtimeformat{<name>}{<format>}
```

to define a new time style. Within `<format>`, the placeholder commands `\THEHOUR`, `\THEMINUTE`, `\THESECOND`, `\THEHOURXII`, `\THETOHOUR`, `\THETOMINUTE` may be used. There are also corresponding placeholder counters: `HOUR`, `MINUTE`, `SECOND`, `HOURXII`, `TOHOUR` and `TOMINUTE`.

This placeholder style of format for the time has the same problems as that for the date styles described in the previous section.

Time styles are defined in `datetime2` using:

`datetime2`

```
\DTMnewtimestyle{<name>}{<definition>}
```

As with the date styles, the `datetime2` package provides styles that redefine a formatting macro. In this case, the time (without the zone) is simply formatted with

`datetime2`

```
\DTMdisplaytime{<hh>}{<mm>}{<ss>}
```

so the `<definition>` part of `\DTMnewtimestyle` needs to redefine this command.

The placeholder commands `\THEHOUR`, `\THEMINUTE` and `\THESECOND` from `datetime` can now be represented by the first, second and third arguments of `\DTMdisplaytime`. The other placeholders are more complicated as they need to be calculated which may prevent the style from being an expandable format.

The `datetime` manual provides a simple example of defining a new time style:

```
\newtimeformat{dottime}{\twodigit{\THEHOUR}.\twodigit{\THEMINUTE}}
```

This style is then set using:

```
\settimeformat{dottime}
```

An equivalent time style can be defined with `datetime2`:

```
\DTMnewtimestyle{dottime}{%  
\renewcommand*{\DTMdisplaytime[3]{\DTMtwodigits{##1}.\DTMtwodigits{##2}}}
```

A complete example using `datetime`:

```
\documentclass{article}
```

```

\usepackage{datetime}

\newtimeformat{dottime}{\twodigit{\THEHOUR}.\twodigit{\THEMINUTE}}
\settimeformat{dottime}

\begin{document}
\currenttime.
\end{document}

```

A complete example using datetime2:

```

\documentclass{article}

\usepackage{datetime2}

\DTMnewtimestyle{dottime}{%
  \renewcommand*{\DTMdisplaytime[3]{\DTMtwodigits{##1}.\DTMtwodigits{##2}}}
\DTMsettimestyle{dottime}

\begin{document}
\DTMcurrenttime.
\end{document}

```

Note that the datetime example has a problem with PDF bookmarks if the time is used in a sectioning command, for the same reasons as those discussed above in the previous section. This can be seen with a slight modification to the example:

```

\documentclass{article}

\usepackage{datetime}
\usepackage[colorlinks]{hyperref}

\newtimeformat{dottime}{\twodigit{\THEHOUR}.\twodigit{\THEMINUTE}}
\settimeformat{dottime}

\begin{document}
\section{\currenttime: an example}

\currenttime.
\end{document}

```

The bookmark appears as:

```

====by 1=by 12by 12 by -=by -60by -1=0.00: an example

```

The datetime2 example works fine with a similar modification:

```

\documentclass{article}

\usepackage{datetime2}

```

```

\usepackage[colorlinks]{hyperref}

\DTMnewtimestyle{dottime}{%
  \renewcommand*\DTMdisplaytime[3]{\DTMtwodigits{##1}.\DTMtwodigits{##2}}
\DTMsettimestyle{dottime}

\begin{document}
\section{\DTMcurrenttime: an example}

\DTMcurrenttime.
\end{document}

```

There is one slight drawback with this example that's irrelevant to the choice of package. The bookmark (and table of contents, if present) will always show the time from the previous \LaTeX run. This is only a problem when using the current time, especially if seconds are required or the document build takes longer than a minute. If the example has a specific time instead of one that changes for every \LaTeX run, then this issue is eliminated.

If you want a time style that requires the other placeholder commands provided by `datetime`, then it's more complicated to convert to `datetime2` as the values that `datetime` conveniently computes and stores in the placeholder commands `\THEHOURXII`, `\THETO HOUR` and `\THETOMINUTE` now need to be calculated, preferably in an expandable way.

An expandable twelve hour time format can be illustrated with the `englishampm` style provided by the `english` module:¹

```

\DTMnewtimestyle
{englishampm}% label
{%
  \renewcommand*\DTMdisplaytime[3]{%
    \ifnum##2=0
      \ifnum##1=12
        \DTMtexorpdfstring
          {\DTMenglishampmfmt{\DTMenglishnoon}}%
          {\DTMenglishnoon}%
      \else
        \ifnum##1=0
          \DTMtexorpdfstring
            {\DTMenglishampmfmt{\DTMenglishmidnight}}%
            {\DTMenglishmidnight}%
        \else
          \ifnum##1=24
            \DTMtexorpdfstring
              {\DTMenglishampmfmt{\DTMenglishmidnight}}%
              {\DTMenglishmidnight}%
          \else
            \ifnum##1<12
              \number##1
              \DTMtexorpdfstring

```

¹Bug fix from `datetime2-english v1.03` included


```

        {\DTMenglishampfmt{\DTMenglisham}}%
        {\DTMenglisham}%
    \else
        \number\numexpr##1-12\relax
        \DTMtexorpdfstring
        {\DTMenglishampfmt{\DTMenglishpm}}%
        {\DTMenglishpm}%
    \fi
    \fi
    \fi
    \fi
\else
\ifnum##1<13
    \ifnum##1=0
        12%
    \else
        \number##1
    \fi
    \DTMenglishtimesep\DTMtwodigits{##2}%
\ifnum##1=12
    \DTMtexorpdfstring
    {\DTMenglishampfmt{\DTMenglishpm}}%
    {\DTMenglishpm}%
\else
    \DTMtexorpdfstring
    {\DTMenglishampfmt{\DTMenglisham}}%
    {\DTMenglisham}%
\fi
\else
    \number\numexpr##1-12\relax
    \DTMenglishtimesep\DTMtwodigits{##2}%
\ifnum##1=24
    \DTMtexorpdfstring
    {\DTMenglishampfmt{\DTMenglisham}}%
    {\DTMenglisham}%
\else
    \DTMtexorpdfstring
    {\DTMenglishampfmt{\DTMenglishpm}}%
    {\DTMenglishpm}%
\fi
\fi
\fi
}%
}%

```

This is certainly more complicated than the `ampmtime` format provided by `datetime`, which is defined as:

```
\newtimeformat{ampmtime}%
```

```

{%
  \ifthenelse{\value{HOUR}=0}{12}{\THEHOURXII}%
  \timeseparator
  \twodigit{\THEMINUTE}%
  \ifthenelse{\value{HOUR}<12}{\amname}%
  {%
    \ifthenelse{\value{HOUR}=12}{ \noon}{\pmname}%
  }%
}

```

However the `datetime2` version produces better results, especially where expansion is required (for example, in bookmarks or writing a time stamp to an external file). The `datetime2` version also performs extra checks for midnight where the `datetime` version produces ambiguous text. So the `ampmt` definition is simple but buggy.

The basic algorithm for `englishampm` is:

If the minute value is 0 (`\ifnum##2=0`):

 If the hour value is 12 (`\ifnum##1=12`):

 Print “noon”

 Otherwise (not noon but on the hour)

 If the hour value is 0 (`\ifnum##1=0`)

 Print “midnight”

 Otherwise (not 12:00 or 00:00 but on the hour)

 If the hour value is 24 (`\ifnum##1=24`)

 Print “midnight”

 Otherwise (not 12:00 or 00:00 or 24:00 but on the hour)

 If the hour value is less than 12 (`\ifnum##1<12`)

 Print the hour value (`\number##1`)

 Print “am”

 Otherwise

 Print the hour value less 12 (`\number\numexpr##1-12`)

 Print “pm”

 Otherwise (minute value isn’t zero)

 If the hour value is less than 13 (`\ifnum##1<13`)

 If the hour value is 0 (`\ifnum##1=0`)

 Print “12”

 Otherwise

 Print the hour (`\number##1`)

Print the separator between the hour and minute (`\DTMenglishtimesep`)

Print minute value (`\DTMtwodigits{##2}`)

If the hour value is 12 (`\ifnum##=12`)

Print “pm”

Otherwise (hour value less than 12)

Print “am”

Otherwise (hour value ≥ 13)

Print the hour value less 12 (`\number\numexpr##1-12`)

Print the time separator

Print minute value (`\DTMtwodigits{##2}`)

If the hour value is 24 (`\ifnum##1=24`)

Print “am”

Otherwise

Print “pm”

Most of the complication in this style isn't trying to determine the equivalent value of `\THEHOURXII` (`\number\numexpr##1-12` if `##1>12`) but in improving the algorithm to catch special cases that the datetime style misses. Additionally, `\DTMtexorpdfstring` is used to prevent any font formatting commands from being added to the bookmarks.

The `datetime2-en-fulltext` package provides an example of a time style that requires calculating the minutes to the hour and the next hour. Note that fragile commands are protected in this style. Since it contains fragile commands that require protection, it can't be used in an expandable context.

Remember that most of the styles provided by `datetime2` and its associated modules and dependent packages are configurable, so if your preferred style is only marginally different to a predefined style, you may be able to tweak that style to fit your requirements. For example, the dot time format can be obtained using the `default` style with the seconds suppressed and the separator change to a dot:

```
\DTMsettimestyle{default}  
\DTMsetup{timesep={.},showseconds=false}
```

The twelve hour format provided by the `english` module can also be adjusted. This style honours the package-wide time-related option `hourminsep` and the “am”, “pm”, “midnight” or “noon” part is formatted according to:

`datetime2-english`

```
\DTMenglishampfmt{<text>}
```

So a twelve hour format that uses a dot instead of a colon and has the text part in small caps can be obtained using:

```
\usepackage[english, hourminsep={.}]{datetime2}
\renewcommand*{\DTMenglishampfmt}[1]{\textsc{#1}}
```

11 The Code

11.1 datetime2.sty code

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{datetime2}[2018/07/20 v1.5.3 (NLCT) date and time formats]
```

Use tracklang to find out what languages have been loaded.

```
3 \RequirePackage{tracklang}
```

Also require etoolbox.

```
4 \RequirePackage{etoolbox}
```

Need xkeyval for *(key)=(value)* interface.

```
5 \RequirePackage{xkeyval}[2006/11/18]
```

`\pdfcreationdate` The `luatex85` package defines `\pdfcreationdate` in terms of `\pdffeedback`, but the parsing commands need a command whose replacement text is directly in the PDF date time format, so define a command with the full replacement text that can be used instead. This will allow for any possible future changes of `\pdfcreationdate` that require deeper levels of expansion.

```
6 \ifdef\pdfcreationdate
7 {%
8   \edef\dtm@pdfcreationdate{\pdfcreationdate}%
9 }%
10 {%
```

Check if newer version of LuaTeX is being used but `luatex85` hasn't been loaded.

```
11 \ifdef\pdffeedback
12 {%
13   \edef\dtm@pdfcreationdate{\pdffeedback creationdate}%
14 }%
15 {%
```

Neither `\pdfcreationdate` nor `\pdffeedback` are defined. (The TeX format is most probably Xe_{La}TeX.) If the locale package has been loaded, `\LocaleNowStamp` should be available.

```
16 \ifdef\LocaleNowStamp
17 {%
18   \ifx\LocaleNowStamp\empty
```

OS query failed, so there's no point trying directly with `texosquery`.

```
19   \else
20     \let\dtm@pdfcreationdate\LocaleNowStamp
21   \fi
22 }%
```

If `texosquery` has been loaded, then the current date and time can be fetched using `\TeXOSQueryNow` provided the shell escape has been enabled. (It might not be, so don't automatically load `texosquery`.) Since it might have been loaded using `\input` rather than `\usepackage`, just test if `\TeXOSQueryNow` has been defined.

```

23   {%
24   \ifdef\TeXOSQueryNow
25   {%
26   \TeXOSQueryNow{\dtm@pdfcreationdate}%
27   \ifdefempty\dtm@pdfcreationdate
28   {%
29 % \end{macrocode}
30 %Failed (maybe shell escape has been disabled).
31 % \begin{macrocode}
32   \undef\dtm@pdfcreationdate
33   }%
34   {}%
35   }%
36   {}%
37   }%
38 }%
39 }

```

`\dtm@yearmonthsep` Separator between year and month for numeric dates.

```
40 \newcommand*{\dtm@yearmonthsep}{-}
```

`\dtm@monthdaysep` Separator between month and day for numeric dates.

```
41 \newcommand*{\dtm@monthdaysep}{-}
```

`\dtm@dayyearsep` Separator between day and year for numeric middle-endian dates.

```
42 \newcommand*{\dtm@dayyearsep}{-}
```

`\dtm@hourminsep` Separator between the hour and minute for times.

```
43 \newcommand*{\dtm@hourminsep}{:}
```

`\dtm@minsecsep` Separator between the minute and second for times.

```
44 \newcommand*{\dtm@minsecsep}{:}
```

`\dtm@timezonesep` Separator between the date and time.

```
45 \newcommand*{\dtm@datetimesep}{\space}
```

`\dtm@timezonesep` Separator between the time and time zone.

```
46 \newcommand*{\dtm@timezonesep}{}
```

`datesep` Set year/month and month/day separator.

```

47 \define@key{datetime2.sty}{datesep}{%
48 \renewcommand*{\dtm@yearmonthsep}{#1}%
49 \renewcommand*{\dtm@monthdaysep}{#1}%
50 \renewcommand*{\dtm@dayyearsep}{#1}%
51 }

```

yearmonthsep Set year/month separator.

```

52 \define@key{datetime2.sty}{yearmonthsep}{%
53   \renewcommand*{\dtm@yearmonthsep}{#1}%
54 }

```

monthdaysep Set month/day separator.

```

55 \define@key{datetime2.sty}{monthdaysep}{%
56   \renewcommand*{\dtm@monthdaysep}{#1}%
57 }

```

dayyearsep Set day/year separator for middle-endian dates.

```

58 \define@key{datetime2.sty}{dayyearsep}{%
59   \renewcommand*{\dtm@dayyearsep}{#1}%
60 }

```

timesep Set hour/minute and minute/second separator.

```

61 \define@key{datetime2.sty}{timesep}{%
62   \renewcommand*{\dtm@hourminsep}{#1}%
63   \renewcommand*{\dtm@minsecsep}{#1}%
64 }

```

hourminsep Set hour/minute separator.

```

65 \define@key{datetime2.sty}{hourminsep}{%
66   \renewcommand*{\dtm@hourminsep}{#1}%
67 }

```

minsecsep Set minute/second separator.

```

68 \define@key{datetime2.sty}{minsecsep}{%
69   \renewcommand*{\dtm@minsecsep}{#1}%
70 }

```

timezonesep Set separator between the time and the time zone (used in `\DTMnow`).

```

71 \define@key{datetime2.sty}{timezonesep}{%
72   \renewcommand*{\dtm@timezonesep}{#1}%
73 }

```

datetimesep Set separator between the date and the time (used in `\DTMnow`).

```

74 \define@key{datetime2.sty}{datetimesep}{%
75   \renewcommand*{\dtm@datetimesep}{#1}%
76 }

```

showseconds Boolean key to determine whether or not to show the seconds.

```

77 \define@boolkey{datetime2.sty}[DTM]{showseconds}[true]{}

```

showdate Boolean key to determine whether or not to show the date in `\DTMdisplay` and `\DTMDisplay`.

```

78 \define@boolkey{datetime2.sty}[DTM]{showdate}[true]{}
79 \DTMshowdatetrue

```

`showzone` Boolean key to determine whether or not to show the time zone in `\DTMdisplay` and `\DTMDisplay`.

```
80 \define@boolkey{datetime2.sty}[DTM]{showzone}[true]{}
```

`showisoZ` Boolean key to determine whether or not to use Z instead of +00:00 for UTC in the default, iso or pdf styles. (Other styles may also use this.)

```
81 \define@boolkey{datetime2.sty}[DTM]{showisoZ}[true]{}  
82 \DTMshowisoZtrue
```

Switch off seconds and time zone if `\dtm@pdfcreationdate` isn't defined, otherwise switch on.

```
83 \ifdef\dtm@pdfcreationdate  
84 {%  
85   \DTMshowsecondstrue  
86   \DTMshowzonetrue  
87 }%  
88 {%  
89   \DTMshowsecondsfalse  
90   \DTMshowzonefalse  
91 }%
```

`showzoneminutes` Boolean key to determine whether or not to show the time zone minutes. (If `\DTMshowzonefalse` then this option is irrelevant.)

```
92 \define@boolkey{datetime2.sty}[DTM]{showzoneminutes}[true]{}  
93 \DTMshowzoneminutestrue
```

`DTMifcaseregional`

```
\DTMifcaseregional{<false>}{<text>}{<numeric>}
```

Determines if the user wants the language modules to set the regional format. The first argument *<false>* indicates that they don't want the regional format set, the second argument *<text>* indicates they want the textual format (e.g. 1st March, 2015 or March 1, 2005) and the third argument *<numeric>* indicates they want the numeric format (e.g. 1/3/2015 or 3/1/2015). A change in the setting will only have an affect when the module is loaded and when `\date<language>` is used to set the style. The default is false.

```
94 \newcommand*{\DTMifcaseregional}[3]{#1}
```

`useregional` Setting to determine whether or not to use the regional settings (if any are loaded).

```
95 \define@choicekey{datetime2.sty}{useregional}[\val\nr]%  
96 {false,text,numeric,num}[text]%  
97 {%  
98   \ifcase\nr\relax  
99     \renewcommand*{\DTMifcaseregional}[3]{##1}%  
100  \or  
101    \renewcommand*{\DTMifcaseregional}[3]{##2}%
```



```

102 \or
103 \renewcommand*\DTMifcaseregional}[3]{##3}%
104 \or
105 \renewcommand*\DTMifcaseregional}[3]{##3}%
106 \fi
107 }

```

@dtm@setusecalc

```

108 \newcommand*\@dtm@setusecalc{%
109 \renewcommand*\@dtm@usecalc{\RequirePackage{datetime2-calc}}%
110 }

```

\@dtm@usecalc

```

111 \newcommand*\@dtm@usecalc{}

```

Disable attempt to load datetime2-calc in the document.

```

112 \AtBeginDocument{%
113 \ifpackageloaded{datetime2-calc}%
114 {%
115 \renewcommand*\@dtm@setusecalc}{}%
116 }%
117 {%
118 \renewcommand*\@dtm@setusecalc{%
119 \PackageError{datetime2}{You must load ‘datetime2-calc’
120 package to use option ‘showdow’}{Try one of the following:^^J
121 pass ‘calc’ option to ‘datetime2’ package when you load it^^J
122 or move ‘showdow’ option to ‘datetime2’ package option list^^J
123 or move \string\DTLsetup\space to the preamble.}%
124 }%
125 }%
126 }

```

calc This option will load the datetime2-calc which uses the pgfcalendar package to compute the day of week and offsets. The package is loaded at the end of this one.

```

127 \DeclareOptionX{calc}{\@dtm@setusecalc}

```

showdow Boolean key to determine whether or not to show the day of week for the styles that can show the day of week. If this is switched on, then datetime2-calc is required. If this key is set later in the document with \DTMsetup, then the datetime2-calc package must previously be loaded for it to have an effect.

```

128 \define@boolkey{datetime2.sty}[DTM]{showdow}[true]{%
129 \ifDTMshowdow \@dtm@setusecalc \fi
130 }
131 \DTMshowdowfalse

```

\@dtm@warning Warning messages.

```

132 \newcommand*\@dtm@warning}[1]{%
133 \if@dtm@warn

```

```

134   \PackageWarning{datetime2}{#1}%
135   \fi
136 }

```

warn Allow user to suppress package warnings.

```

137 \define@boolkey{datetime2.sty}[@dtm@]{warn}[true]{%
138 \@dtm@warntrue

```

tm@initialstyle

```

139 \newcommand*{\@dtm@initialstyle}{}

```

style Set the style. This automatically sets `useregional=false`.

```

140 \define@key{datetime2.sty}{style}{%
141   \renewcommand*{\@dtm@initialstyle}{#1}%
142   \ifstrempy{#1}%
143   {}%
144   {%
145     \renewcommand*{\DTMifcaseregional}[3]{##1}%
146   }%
147 }

```

Pass any unknown options to `tracklang`. This will automatically switch the `useregional` setting to `text`.

```

148 \DeclareOptionX*{%
149   \ifcsundef{@tracklang@add@\CurrentOption}%
150   {%
151     \PackageError{datetime2}{‘\CurrentOption’ is not a recognised dialect.
152     \MessageBreak Perhaps you have misspelt it or the
153     \MessageBreak named dialect may be unsupported or
154     \MessageBreak perhaps you forgot to use the ‘style’ key}%
155     {Any options that aren’t described in the manual are assumed
156     \MessageBreak to be language or dialect names.}%
157   }%
158   {%
159     \TrackPredefinedDialect{\CurrentOption}%
160     \renewcommand*{\DTMifcaseregional}[3]{#2}%
161   }%
162 }

```

Process options passed to this package:

```

163 \ProcessOptionsX

```

Disable `calc` option. If it’s required, just load `datetime2-calc` with `\usepackage`.

```

164 \disable@keys{datetime2.sty}{calc}

```

Disable `style` option. If it’s required, just use `\DTMsetup`.

```

165 \disable@keys{datetime2.sty}{style}

```

Provide a way to set options after package has been loaded.

\DTMsetup

```
166 \newcommand*{\DTMsetup}[1]{%
167   \def\@dtm@usecalc{}%
168   \setkeys{datetime2.sty}{#1}%
169   \@dtm@usecalc
170 }
```

11.1.1 Defaults

This section sets up the defaults.

\@dtm@parsedate Parse date in the format $\langle year \rangle - \langle month \rangle - \langle day \rangle$. The arguments are expanded. (This is redefined by datetime2-calc.)

```
171 \def\@dtm@parsedate#1-#2-#3\@dtm@endparsedate{%
172   \edef\@dtm@year{\number#1}%
173   \edef\@dtm@month{\number#2}%
174   \edef\@dtm@day{\number#3}%
175   \def\@dtm@dow{-1}%
176 }
```

\@dtm@parsetime Define command to parse time in the format $\langle h \rangle : \langle m \rangle : \langle s \rangle$. The results are stored in \@dtm@hour, \@dtm@minute and \@dtm@second. The arguments are expanded.

```
177 \def\@dtm@parsetime#1:#2:#3\@dtm@endparsetime{%
178   \edef\@dtm@hour{\number#1}%
179   \edef\@dtm@minute{\number#2}%
180   \edef\@dtm@second{\number#3}%
181 }
```

\@dtm@parsetimezn Define command to parse time in the format $\langle h \rangle : \langle m \rangle : \langle s \rangle \langle znh \rangle : \langle znm \rangle$. The results are stored in \@dtm@hour, \@dtm@minute, \@dtm@second, \@dtm@timezonehour and \@dtm@timezoneminute. The arguments are expanded.

```
182 \def\@dtm@parsetimezn#1:#2:#3 #4\@dtm@endparsetimezn{%
183   \@dtm@parsetime#1:#2:#3\@dtm@endparsetime
184   \@dtm@parsezone{#4}%
185 }
```

\@dtm@parsezone Define command to parse time zone in the format Z or $\langle znh \rangle : \langle znm \rangle$. The results are stored in \@dtm@timezonehour and \@dtm@timezoneminute. The arguments are expanded in the event that registers are used.

```
186 \newcommand*{\@dtm@parsezone}[1]{%
187   \ifstrequal{#1}{Z}%
188   {%
189     \def\@dtm@timezonehour{+00}%
190     \def\@dtm@timezoneminute{00}%
191   }%
192   {%
193     \@dtm@parse@zone#1\@dtm@endparse@zone
```

```

194 }%
195 }
196 \def\@dtm@parse@zone#1:#2\@dtm@endparse@zone{%
197 \edef\@dtm@timezonehour{\number#1}%
198 \edef\@dtm@timezoneminute{\number#2}%
199 }

```

`@parsetimestamp` Parse date and time in ISO format `<YYYY>-<MM>-<DD>T<hh>:<mm>:<sec><time zone>` where `<time zone>` may be Z or in the form `<hh>:<mm>` (where `<hh>` includes the sign).

```

200 \def\@dtm@parsetimestamp#1-#2-#3T#4:#5:#6#7#8\@dtm@endparsetimestamp{%
201 \@dtm@parsedate#1-#2-#3\@dtm@endparsedate
202 \@dtm@parsetime#4:#5:#6#7\@dtm@endparsetime
203 \@dtm@parsezone{#8}%
204 }

```

`savefilemoddate` Not available for some engines.

```

205 \newcommand*\DTMsavefilemoddate}[2]{%
206 \@dtm@warning{Your TeX engine doesn't support accessing
207 file modification dates}%
208 \cslet{\@dtm@#1@year}{0}%
209 \cslet{\@dtm@#1@month}{0}%
210 \cslet{\@dtm@#1@day}{0}%
211 \cslet{\@dtm@#1@dow}{-1}%
212 \cslet{\@dtm@#1@hour}{0}%
213 \cslet{\@dtm@#1@minute}{0}%
214 \cslet{\@dtm@#1@second}{0}%
215 \cslet{\@dtm@#1@TZhour}{0}%
216 \cslet{\@dtm@#1@TZminute}{0}%
217 }

```

`savefrompdfdata` Save a date-time stamp that's specified in PDF format.

```

218 \newcommand*\DTMsavefrompdfdata}[2]{%
219 \edef\@dtm@tmp{#2}%
220 \expandafter\@dtm@parsepdfdatetime\@dtm@tmp\@dtm@endparsepdfdatetime
221 \cslet{\@dtm@#1@year}{\@dtm@year}%
222 \cslet{\@dtm@#1@month}{\@dtm@month}%
223 \cslet{\@dtm@#1@day}{\@dtm@day}%
224 \cslet{\@dtm@#1@dow}{\@dtm@dow}%
225 \cslet{\@dtm@#1@hour}{\@dtm@hour}%
226 \cslet{\@dtm@#1@minute}{\@dtm@minute}%
227 \cslet{\@dtm@#1@second}{\@dtm@second}%
228 \cslet{\@dtm@#1@TZhour}{\@dtm@timezonehour}%
229 \cslet{\@dtm@#1@TZminute}{\@dtm@timezoneminute}%
230 }

```

Find out the current time. If `\dtm@pdfcreationdate` is defined, it can be fetched from that.

```

231 \ifdef\dtm@pdfcreationdate
232 {%

```

Define commands to parse \dtm@pdfcreationdate

```
233 \def\@dtm@parsepdfdatetime#1:#2#3#4#5#6#7#8#9{%
234   \def\@dtm@year{#2#3#4#5}%
235   \def\@dtm@month{#6#7}%
236   \def\@dtm@day{#8#9}%
237   \@dtm@parsepdftime
238 }
239 \def\@dtm@parsepdftime#1#2#3#4#5#6#7\@dtm@endparsepdfdatetime{%
240   \def\@dtm@hour{#1#2}%
241   \def\@dtm@minute{#3#4}%
242   \def\@dtm@second{#5#6}%
243   \ifstrequal{#7}{Z}%
244   {%
245     \def\@dtm@timezonehour{00}%
246     \def\@dtm@timezoneminute{00}%
247   }%
248   {%
249     \@dtm@parsepdftimezone#7%
250   }%
251 }
252 \def\@dtm@parsepdftimezone#1'#2' {%
253   \def\@dtm@timezonehour{#1}%
254   \def\@dtm@timezoneminute{#2}%
255 }%
```

Now parse \dtm@pdfcreationdate

```
256 \expandafter\@dtm@parsepdfdatetime\dtm@pdfcreationdate\@dtm@endparsepdfdatetime
```

Save the values.

```
257 \let\@dtm@currentyear\@dtm@year
258 \let\@dtm@currentmonth\@dtm@month
259 \let\@dtm@currentday\@dtm@day
260 \let\@dtm@currenthour\@dtm@hour
261 \let\@dtm@currentminute\@dtm@minute
262 \let\@dtm@currentsecond\@dtm@second
263 \let\@dtm@currenttimezonehour\@dtm@timezonehour
264 \let\@dtm@currenttimezoneminute\@dtm@timezoneminute
265 %
```

LaTeX doesn't provide \pdffilemoddate.

```
266 \ifdef\pdffilemoddate
267 {%
268   \renewcommand*\DTMsavefilemoddate}[2]{%
269     \expandafter\@dtm@parsepdfdatetime\pdffilemoddate{#2}\@dtm@endparsepdfdatetime
270     \cslet{\@dtm@#1@year}{\@dtm@year}%
271     \cslet{\@dtm@#1@month}{\@dtm@month}%
272     \cslet{\@dtm@#1@day}{\@dtm@day}%
273     \cslet{\@dtm@#1@dow}{\@dtm@dow}%
274     \cslet{\@dtm@#1@hour}{\@dtm@hour}%
275     \cslet{\@dtm@#1@minute}{\@dtm@minute}%

```

```

276 \cslet{@dtm@#1@second}{\@dtm@second}%
277 \cslet{@dtm@#1@TZhour}{\@dtm@timezonehour}%
278 \cslet{@dtm@#1@TZminute}{\@dtm@timezoneminute}%
279 }
280 }%
281 {%
282 \ifdef\directlua
283 {

```

Lua time zone information provided by %z is OS dependent, so this might not work.

```

284 \renewcommand*{\DTMsavefilemoddate}[2]{%
285 \expandafter\@dtm@parseluadatetime
286 \directlua{tex.print(os.date(
287 "\expandafter\@gobble\string\%Y-%
288 \expandafter\@gobble\string\%m-%
289 \expandafter\@gobble\string\%d-%
290 \expandafter\@gobble\string\%w
291 \expandafter\@gobble\string\%H:%
292 \expandafter\@gobble\string\%M:%
293 \expandafter\@gobble\string\%S
294 \expandafter\@gobble\string\%z",
295 lfs.attributes("#2").modification))}%
296 \@dtm@endparseluadatetime
297 \cslet{@dtm@#1@year}{\@dtm@year}%
298 \cslet{@dtm@#1@month}{\@dtm@month}%
299 \cslet{@dtm@#1@day}{\@dtm@day}%
300 \cslet{@dtm@#1@dow}{\@dtm@dow}%
301 \cslet{@dtm@#1@hour}{\@dtm@hour}%
302 \cslet{@dtm@#1@minute}{\@dtm@minute}%
303 \cslet{@dtm@#1@second}{\@dtm@second}%
304 \cslet{@dtm@#1@TZhour}{\@dtm@TZhour}%
305 \cslet{@dtm@#1@TZminute}{\@dtm@TZminute}%
306 }
307 \def\@dtm@parseluadatetime#1-#2-#3-#4 #5:#6:#7 #8\@dtm@endparseluadatetime{%
308 \edef\@dtm@year{\number#1}%
309 \edef\@dtm@month{\number#2}%
310 \edef\@dtm@day{\number#3}%
311 \edef\@dtm@dow{\number#4}%
312 \edef\@dtm@hour{\number#5}%
313 \edef\@dtm@minute{\number#6}%
314 \edef\@dtm@second{\number#7}%
315 \@dtm@parseluatimezone#8000000\@dtm@endparseluatimezone
316 }
317 \def\@dtm@parseluatimezone#1#2#3#4#5#6{%
318 \ifstrequal{#1}{+}%
319 {%
320 \def\@dtm@TZhour{#1#2#3}%
321 \ifstrequal{#4}{:}%
322 {%
323 \def\@dtm@TZminute{#5#6}%

```

```

324     }%
325     {%
326         \def\@dtm@TZminute{#4#5}%
327     }%
328 }%
329 {%
330     \ifstrequal{#1}{-}%
331     {%
332         \def\@dtm@TZhour{#1#2#3}%
333         \ifstrequal{#4}{:}%
334         {%
335             \def\@dtm@TZminute{#5#6}%
336         }%
337     }%
338     \def\@dtm@TZminute{#4#5}%
339 }%
340 }%
341 {%
342     \ifstrequal{#1}{Z}%
343     {%
344         \def\@dtm@TZhour{0}%
345         \def\@dtm@TZminute{0}%
346     }%
347     {%
348         \def\@dtm@TZhour{#1#2}%
349         \ifstrequal{#3}{:}%
350         {%
351             \def\@dtm@TZminute{#4#5}%
352         }%
353     }%
354     \def\@dtm@TZminute{#3#4}%
355 }%
356 }%
357 }%
358 }%
359 \@dtm@parseluatimezone
360 }
361 \def\@dtm@parseluatimezone#1\@dtm@endparseluatimezone{%
362 }
363 }
364 {%

```

If `\directlua` isn't defined but `texosquery` has been loaded, we can use `\TeXOSQueryFileDate` if the shell escape is enabled.

```

365     \ifdef\TeXOSQueryFileDate
366     {
367         \renewcommand*{\DTMsavefilemoddate}[2]{%
368             \TeXOSQueryFileDate{\@dtm@tmp}{#2}%
369             \ifdefempty\@dtm@tmp
370             {%

```

OS query failed.

```
371         \@dtm@warning{Your TeX engine doesn't support accessing
372         file modification dates and the attempt to use texosquery
373         failed}%
374         \cslet{@dtm@#1@year}{0}%
375         \cslet{@dtm@#1@month}{0}%
376         \cslet{@dtm@#1@day}{0}%
377         \cslet{@dtm@#1@dow}{-1}%
378         \cslet{@dtm@#1@hour}{0}%
379         \cslet{@dtm@#1@minute}{0}%
380         \cslet{@dtm@#1@second}{0}%
381         \cslet{@dtm@#1@TZhour}{0}%
382         \cslet{@dtm@#1@TZminute}{0}%
383     }%
384     {%
385         \expandafter\@dtm@parsepdfdatetime\@dtm@tmp\@dtm@endparsepdfdatetime
386         \cslet{@dtm@#1@year}{\@dtm@year}%
387         \cslet{@dtm@#1@month}{\@dtm@month}%
388         \cslet{@dtm@#1@day}{\@dtm@day}%
389         \cslet{@dtm@#1@dow}{\@dtm@dow}%
390         \cslet{@dtm@#1@hour}{\@dtm@hour}%
391         \cslet{@dtm@#1@minute}{\@dtm@minute}%
392         \cslet{@dtm@#1@second}{\@dtm@second}%
393         \cslet{@dtm@#1@TZhour}{\@dtm@timezonehour}%
394         \cslet{@dtm@#1@TZminute}{\@dtm@timezoneminute}%
395     }%
396 }%
397 }
398 {}
399 }
400 }%
401 }%
402 {%
```

\pdfcreationdate not defined. By a process of elimination, the T_EX engine is either X_YT_EX or it's very old. (Or it may be a new version of LuaT_EX without luatex85.) In this case, the seconds and time zone can't be obtained. The hour and minute need to be calculated from T_EX's \time primitive.

```
403     \count@=\time\relax
404     \divide\count@ by 60\relax
405     \edef\@dtm@currenthour{\number\count@}%
406     \multiply\count@ by -60\relax
407     \advance\count@ by \time\relax
408     \edef\@dtm@currentminute{\number\count@}%
409     \newcommand*\@dtm@currentsecond{00}%
410     \newcommand\@dtm@currenttimezonehour{00}%
411     \newcommand\@dtm@currenttimezoneminute{00}%
```

Get the day, month and year from T_EX's primitives.

```
412     \edef\@dtm@currentyear{\number\year}%
```



```

413 \edef\@dtm@currentmonth{\number\month}%
414 \edef\@dtm@currentday{\number\day}%
415 }

```

Make \DTMsavefilemoddate robust.

```
416 \robustify\DTMsavefilemoddate
```

Current day of week defaults to -1 (that is, ignore it).

@dtm@currentdow

```
417 \newcommand*\@dtm@currentdow{-1}
```

Allow X_YLaTeX users a way of manually setting the current time zone.

Msetcurrentzone

```

418 \newcommand*\DTMsetcurrentzone[2]{%
419 \renewcommand\@dtm@currenttimezonehour{#1}%
420 \renewcommand\@dtm@currenttimezoneminute{#2}%
421 }

```

\DTMtoday Provided in case of conflict to obtain datettime2's version of \today.

```

422 \newcommand*\DTMtoday{%
423 \DTMdisplaydate
424 {\@dtm@currentyear}%
425 {\@dtm@currentmonth}%
426 {\@dtm@currentday}%
427 {\@dtm@currentdow}%
428 }

```

\today Version 1.4 dropped \renewcommand in case \today hasn't already been defined.

```
429 \let\today\DTMtoday
```

The scr_ltr2 class redefines \today at the start of the document, so check for this.

```
430 \@ifclassloaded{scrltr2}{\AtBeginDocument{\let\today\DTMtoday}}{}
```

\DTMToday First letter upper case version. Added to v1.4 to provide datettime2's version in case of conflict.

```

431 \newcommand*\DTMToday{%
432 \DTMDisplaydate
433 {\@dtm@currentyear}%
434 {\@dtm@currentmonth}%
435 {\@dtm@currentday}%
436 {\@dtm@currentdow}%
437 }

```

\Today

```
438 \let\Today\DTMToday
```

\DTMdisplaydate \DTMdisplaydate{\year}{\month}{\day}{\day of week}

Display the given date. If the day of week is negative, ignore it. The default style ignores it regardless.

```
439 \newcommand*\DTMdisplaydate[4]{%
440   \number#1\dtm@yearmonthsep\DTMtwdigits{#2}\dtm@monthdaysep\DTMtwdigits{#3}%
441 }%
```

`\DTMDisplaydate` First letter upper case version. Defaults to `\DTMdisplaydate`.

```
442 \newcommand*\DTMDisplaydate{\DTMdisplaydate}
```

`\DTMdate` Display date where date is specified in the format `<yyy>-<mm>-<dd>`. Use `\expandafter` in case argument is a control sequence containing the date. This command isn't expandable

```
443 \newrobustcmd*\DTMdate[1]{%
444   \expandafter\@dtm@parsedate#1\@dtm@endparsedate
445   \DTMdisplaydate{\@dtm@year}{\@dtm@month}{\@dtm@day}{\@dtm@dow}%
446 }
```

`\DTMDate` Upper case version.

```
447 \newrobustcmd*\DTMDate[1]{%
448   \expandafter\@dtm@parsedate#1\@dtm@endparsedate
449   \DTMdisplaydate{\@dtm@year}{\@dtm@month}{\@dtm@day}{\@dtm@dow}%
450 }
```

`\DTMcurrenttime` Display the current time.

```
451 \newcommand*\DTMcurrenttime{%
452   \DTMdisplaytime
453   {\@dtm@currenthour}%
454   {\@dtm@currentminute}%
455   {\@dtm@currentsecond}%
456 }
```

`\DTMdisplaytime` `\DTMdisplaytime{<hour>}{<minute>}{<sec>}`

Display the given time.

```
457 \newcommand*\DTMdisplaytime[3]{%
458   \DTMtwdigits{#1}\dtm@hourminsep\DTMtwdigits{#2}%
459   \ifDTMshowseconds\dtm@minsecsep\DTMtwdigits{#3}\fi
460 }%
```

`\DTMtime` Display date where time is specified in the format `<hour>:<minute>:<seconds>`. This uses `\expandafter` in case argument is a control sequence containing the time. Not expandable.

```
461 \newrobustcmd*\DTMtime[1]{%
462   \@dtm@parsetime#1\@dtm@endparsetime
463   \DTMdisplaytime{\@dtm@hour}{\@dtm@minute}{\@dtm@second}%
464 }
```

```

\DTMcurrentzone  Display the current time zone.
465 \newcommand*{\DTMcurrentzone}{%
466 \DTMdisplayzone
467 {\@dtm@currenttimezonehour}%
468 {\@dtm@currenttimezoneminute}%
469 }

```

```

\DTMdisplayzone  Display time zone.
470 \newcommand*{\DTMdisplayzone}[2]{%
471 \ifboolexpe
472 { bool{DTMshowisoZ}
473   and test{\ifnumequal{#1}{0}}
474   and test{\ifnumequal{#2}{0}}
475 }%
476 {%
477   Z%
478 }%
479 {%
480   \ifnum#1<0\else+\fi\DTMtwodigits{#1}%
481   \ifDTMshowzoneminutes\dtm@hourminsep\DTMtwodigits{#2}\fi
482 }%
483 }

```

```

\DTMnow  Current date, time and time zone.
484 \newcommand*{\DTMnow}{%
485 \DTMdisplay
486 {\@dtm@currentyear}
487 {\@dtm@currentmonth}
488 {\@dtm@currentday}
489 {\@dtm@currentdow}
490 {\@dtm@currenthour}%
491 {\@dtm@currentminute}%
492 {\@dtm@currentsecond}%
493 {\@dtm@currenttimezonehour}%
494 {\@dtm@currenttimezoneminute}%
495 }

```

```

\DTMNow  Current date, time and time zone.
496 \newcommand*{\DTMNow}{%
497 \DTMDisplay
498 {\@dtm@currentyear}
499 {\@dtm@currentmonth}
500 {\@dtm@currentday}
501 {\@dtm@currentdow}
502 {\@dtm@currenthour}%
503 {\@dtm@currentminute}%
504 {\@dtm@currentsecond}%
505 {\@dtm@currenttimezonehour}%
506 {\@dtm@currenttimezoneminute}%

```

507 }

```
\DTMdisplay \DTMdisplay{<YYYY>}{<MM>}{<DD>}{<DOW>}{<hh>}{<mm>}{<ss>}  
{<TZh>}{<TZm>}
```

Display the date and time.

```
508 \newcommand*{\DTMdisplay}[9]{%  
509 \ifDTMshowdate  
510 \DTMdisplaydate{#1}{#2}{#3}{#4}%  
511 \dtm@datetimesep  
512 \fi  
513 \DTMdisplaytime  
514 {#5}%  
515 {#6}%  
516 {#7}%  
517 \ifDTMshowzone  
518 \dtm@timezonesep  
519 \DTMdisplayzone  
520 {#8}%  
521 {#9}%  
522 \fi  
523 }
```

```
\DTMDisplay \DTMDisplay{<YYYY>}{<MM>}{<DD>}{<DOW>}{<hh>}{<mm>}{<ss>}  
{<TZh>}{<TZm>}
```

First letter upper case version. Defaults to \DTMdisplay.

```
524 \newcommand*{\DTMDisplay}{\DTMdisplay}
```

11.1.2 Styles

Provide user level commands for displaying number as two digits. (Truncate if over 99, to allow for converting year to two digits).

\DTMtwodigits

```
525 \newcommand*{\DTMtwodigits}[1]{%  
526 \ifnum#1<0  
527 -\DTMtwodigits{-#1}%  
528 \else  
529 \ifnum#1<100  
530 \ifnum#1<10  
531 0\number#1  
532 \else  
533 \number#1
```

```

534     \fi
535   \else
\numexpr rounds rather than truncates integer division, which is a little awkward to get
around in an expandable context.
536     \ifnum\numexpr#1-(#1/100)*100<0
537       \number\numexpr#1-((#1/100)-1)*100\relax
538     \else
539       \number\numexpr#1-(#1/100)*100\relax
540     \fi
541   \fi
542 \fi
543 }

```

`\DTMcentury` Expands to the given number divided by 100 rounded upwards (in absolute terms). Provided in case the user just wants the century.

```

544 \newcommand*\DTMcentury}[1]{%
545   \ifnum#1<0
546     -\DTMcentury{-#1}%
547   \else
548     \ifnum\numexpr#1-(#1/100)*100<1
549       \number\numexpr#1/100\relax
550     \else
551       \number\numexpr(#1/100)+1\relax
552     \fi
553   \fi
554 }

```

`\DTMdivhundred` Expands to the given number divided by 100.

```

555 \newcommand*\DTMdivhundred}[1]{%
556   \ifnum#1<0
557     -\DTMdivhundred{-#1}%
558   \else
559     \ifnum\numexpr#1-(#1/100)*100<0
560       \number\numexpr(#1)/100-1\relax
561     \else
562       \number\numexpr((#1)/100)\relax
563     \fi
564   \fi
565 }

```

`\Mtexorpdfstring` Provide user with a command that will use `hyperref`'s `\texorpdfstring` if `hyperref` has been loaded. If `hyperref` isn't loaded it just does the first argument.

```

566 \newcommand*\DTMtexorpdfstring}[2]{#1}
567 \AtBeginDocument{%
568   \@ifpackageloaded{hyperref}%
569   {%
570     \renewcommand*\DTMtexorpdfstring}{\texorpdfstring}%
571   }%

```

```
572 {}%
573 }
```

Access separator:

`\DTMsep`

```
574 \newcommand*{\DTMsep}[1]{\csname dtm@#1sep\endcsname}
```

Date-only styles are stored internally as `\@dtm@datestyle@<label>`, time-only styles are stored internally as `\@dtm@timestyle@<label>`, zone-only styles are stored internally as `\@dtm@zonestyle@<label>`.

`\DTMnewdatestyle` Define a new date-only style. This should only redefine `\DTMdisplaydate` and `\DTMDisplaydate`, which may or may not use the separators `\dtm@yearmonthsep` and `\dtm@monthdaysep`.

```
575 \newcommand*{\DTMnewdatestyle}[2]{%
576   \ifcsdef{@dtm@datestyle@#1}%
577   {%
578     \PackageError{datetime2}{Date style ‘#1’ already exists}{}%
579   }%
580   {%
581     \csdef{@dtm@datestyle@#1}{#2}%
582   }%
583 }
```

`\DTMrenewdatestyle` Redefine a date style. This should only redefine `\DTMdisplaydate` and `\DTMDisplaydate`, which may or may not use the separators `\dtm@yearmonthsep` and `\dtm@monthdaysep`. This may also be used to modify the date part of a full style.

```
584 \newcommand*{\DTMrenewdatestyle}[2]{%
585   \ifcsundef{@dtm@datestyle@#1}%
586   {%
587     \PackageError{datetime2}{Date style ‘#1’ doesn’t exist}{}%
588   }%
589   {%
590     \csdef{@dtm@datestyle@#1}{#2}%
591   }%
592 }
```

`\DTMprovidedatestyle` Define a date style if it doesn't already exist.

```
593 \newcommand*{\DTMprovidedatestyle}[2]{%
594   \ifcsdef{@dtm@datestyle@#1}%
595   {%
596   }%
597   {%
598     \csdef{@dtm@datestyle@#1}{#2}%
599   }%
600 }
```

`\DTMnewtimestyle` Define a new time-only style. This should only redefine `\DTMdisplaytime`, which may or may not use the separators `\dtm@hourminsep` and `\dtm@minsecsep`.

```

601 \newcommand*{\DTMnewtimestyle}[2]{%
602   \ifcsdef{@dtm@timestyle@#1}%
603   {%
604     \PackageError{datetime2}{Time style ‘#1’ already exists}{}%
605   }%
606   {%
607     \csdef{@dtm@timestyle@#1}{#2}%
608   }%
609 }

```

Mrenewtimestyle Redefine a time style. This should only redefine `\DTMdisplaytime`, which may or may not use the separators `\dtm@hourminsep` and `\dtm@minsecsep`. This may also be used to modify the time part of a full style.

```

610 \newcommand*{\DTMrenewtimestyle}[2]{%
611   \ifcsundef{@dtm@timestyle@#1}%
612   {%
613     \PackageError{datetime2}{Time style ‘#1’ doesn’t exist}{}%
614   }%
615   {%
616     \csdef{@dtm@timestyle@#1}{#2}%
617   }%
618 }

```

providetimestyle Define a time style if it doesn’t already exist.

```

619 \newcommand*{\DTMprovidetimestyle}[2]{%
620   \ifcsdef{@dtm@timestyle@#1}%
621   {%
622   }%
623   {%
624     \csdef{@dtm@timestyle@#1}{#2}%
625   }%
626 }

```

DTMnewzonestyle Define a new zone-only style. This should only redefine `\DTMdisplayzone`, which may or may not use the separator `\dtm@hourminsep`.

```

627 \newcommand*{\DTMnewzonestyle}[2]{%
628   \ifcsdef{@dtm@zonestyle@#1}%
629   {%
630     \PackageError{datetime2}{Zone style ‘#1’ already exists}{}%
631   }%
632   {%
633     \csdef{@dtm@zonestyle@#1}{#2}%
634   }%
635 }

```

Mrenewzonestyle Redefine a new zone style. This should only redefine `\DTMdisplayzone`, which may or may not use the separator `\dtm@hourminsep`. This may also be used to modify the zone part of a full style.

```

636 \newcommand*\DTMrenewzonestyle}[2]{%
637   \ifcsundef{@dtm@zonestyle@#1}%
638   {%
639     \PackageError{datetime2}{Zone style ‘#1’ doesn’t exist}{}%
640   }%
641   {%
642     \csdef{@dtm@zonestyle@#1}{#2}%
643   }%
644 }

```

`\DTMrenewzonestyle` Defines a new zone style if it doesn't already exist.

```

645 \newcommand*\DTMprovidezonestyle}[2]{%
646   \ifcsdef{@dtm@zonestyle@#1}%
647   {%
648   }%
649   {%
650     \csdef{@dtm@zonestyle@#1}{#2}%
651   }%
652 }

```

Zone styles may use mappings to use a regional time zone (such as GMT or BST). It's up to the language modules to define these mappings. A mapping for time zone $\langle TZh \rangle : \langle TZm \rangle$ is stored in $\backslash @dtm@zonemap@ \langle TZh \rangle : \langle TZm \rangle$.

```

\DTMdefzonemap \DTMdefzonemap{\langle TZh \rangle}{\langle TZm \rangle}{\langle map \rangle}

```

This will override any previous mapping for the given time zone.

```

653 \newcommand*\DTMdefzonemap}[3]{%
654   \csdef{@dtm@zonemap@DTMtwodigits{#1}:DTMtwodigits{#2}}{#3}%
655 }

```

`\DTMdefzonemap` Expands to the mapping or the default if not defined.

```

656 \newcommand*\DTMusedzonemapordefault}[2]{%
657   \ifcsundef{@dtm@zonemap@DTMtwodigits{#1}:DTMtwodigits{#2}}%
658   {%
659     \ifnum#1<0\else+ \fi
660     \DTMtwodigits{#1}%
661     \ifDTMshowzoneminutes\DTMsep{hourmin}\DTMtwodigits{#2} \fi
662   }%
663   {\csname @dtm@zonemap@DTMtwodigits{#1}:DTMtwodigits{#2}\endcsname}%
664 }

```

`\DTMusedzonemap` Expands to the mapping. (No check if defined.)

```

665 \newcommand*\DTMusedzonemap}[2]{%
666   \csname @dtm@zonemap@DTMtwodigits{#1}:DTMtwodigits{#2}\endcsname
667 }

```


`\DTMhaszonemap`

```
668 \newcommand*{\DTMhaszonemap}[4]{%
669   \ifcsundef{@dtm@zonemap@DTMtwodigits{#1}:\DTMtwodigits{#2}}{#4}{#3}%
670 }
```

`\DTMclearmap` Undefineds the given zone mapping. No check is made to determine if the map exists.

```
671 \newcommand*{\DTMclearmap}[2]{%
672   \csundef{@dtm@zonemap@DTMtwodigits{#1}:\DTMtwodigits{#2}}%
673 }
```

`\DTMshowmap` Debugging command.

```
674 \newcommand*{\DTMshowmap}[2]{%
675   \csshow{@dtm@zonemap@DTMtwodigits{#1}:\DTMtwodigits{#2}}%
676 }
```

`\DTMresetzones` Regional modules should use this before setting their local zones, so that users can unset previously defined zones that are outside the region if they require. By default this does nothing, so no modifications are made.

```
677 \newcommand*{\DTMresetzones}{}
```

`DTMNatoZoneMaps` Provide a command to set the time zone abbreviations to the military/NATO style.

```
678 \newcommand*{\DTMNatoZoneMaps}{%
679   \defzonemap{01}{00}{A}% Alpha time zone
680   \defzonemap{02}{00}{B}% Bravo time zone
681   \defzonemap{03}{00}{C}% Charlie time zone
682   \defzonemap{04}{00}{D}% Delta time zone
683   \defzonemap{05}{00}{E}% Echo time zone
684   \defzonemap{06}{00}{F}% Foxtrot time zone
685   \defzonemap{07}{00}{G}% Golf time zone
686   \defzonemap{08}{00}{H}% Hotel time zone
687   \defzonemap{09}{00}{I}% India time zone
688   \defzonemap{10}{00}{K}% Kilo time zone
689   \defzonemap{11}{00}{L}% Lima time zone
690   \defzonemap{12}{00}{M}% Mike time zone
691   \defzonemap{-01}{00}{N}% November time zone
692   \defzonemap{-02}{00}{O}% Oscar time zone
693   \defzonemap{-03}{00}{P}% Papa time zone
694   \defzonemap{-04}{00}{Q}% Quebec time zone
695   \defzonemap{-05}{00}{R}% Romeo time zone
696   \defzonemap{-06}{00}{S}% Sierra time zone
697   \defzonemap{-07}{00}{T}% Tango time zone
698   \defzonemap{-08}{00}{U}% Uniform time zone
699   \defzonemap{-09}{00}{V}% Victor time zone
700   \defzonemap{-10}{00}{W}% Whiskey time zone
701   \defzonemap{-11}{00}{X}% X-ray time zone
702   \defzonemap{-12}{00}{Y}% Yankee time zone
703   \defzonemap{00}{00}{Z}% Zulu time zone
704 }
```

```
\DTMifhasstyle \DTMifhasstyle{<label>}{<>true>}{<>false>}
```

Test to determine if style exists.

```
705 \newcommand*{\DTMifhasstyle}[3]{%  
706 \ifcsdef{@dtm@style@#1}{#2}{#3}%  
707 }
```

```
TMifhasdatestyle \DTMifhasdatestyle{<label>}{<>true>}{<>false>}
```

Test to determine if partial date style exists.

```
708 \newcommand*{\DTMifhasdatestyle}[3]{%  
709 \ifcsdef{@dtm@datestyle@#1}{#2}{#3}%  
710 }
```

```
TMifhastimestyle \DTMifhastimestyle{<label>}{<>true>}{<>false>}
```

Test to determine if time style exists.

```
711 \newcommand*{\DTMifhastimestyle}[3]{%  
712 \ifcsdef{@dtm@timestyle@#1}{#2}{#3}%  
713 }
```

```
TMifhaszonestyle \DTMifhaszonestyle{<label>}{<>true>}{<>false>}
```

Test to determine if time zone style exists.

```
714 \newcommand*{\DTMifhaszonestyle}[3]{%  
715 \ifcsdef{@dtm@zonestyle@#1}{#2}{#3}%  
716 }
```

```
\DTMnewstyle \DTMnewstyle{<label>}{<date style definition>}{<time style definition>}  
{<zone style definition>}{<full format definition>}
```

Define a new style. The full format redefines \DTMdisplay and \DTMDisplay.

```
717 \newcommand*{\DTMnewstyle}[5]{%  
718 \DTMifhasstyle{#1}%  
719 {%  
720 \PackageError{datetime2}{Style ‘#1’ already exists}{}}%
```

```

721 }%
722 {%
723   \DTMnewdatestyle{#1}{#2}%
724   \DTMnewtimestyle{#1}{#3}%
725   \DTMnewzonestyle{#1}{#4}%
726   \csdef{@dtm@style@#1}{%
727     \csuse{@dtm@datestyle@#1}%
728     \csuse{@dtm@timestyle@#1}%
729     \csuse{@dtm@zonestyle@#1}%
730     #5%
731   }%
732 }%
733 }

```

`\DTMrenewstyle` `\DTMrenewstyle{<label>}{<date style definition>}{<time style definition>}{<zone style definition>}{<full format definition>}`

Redefine a style. The full format redefines `\DTMdisplay` and `\DTMDisplay`.

```

734 \newcommand*{\DTMrenewstyle}[5]{%
735   \DTMifhasstyle{#1}%
736   {%
737     \DTMrenewdatestyle{#1}{#2}%
738     \DTMrenewtimestyle{#1}{#3}%
739     \DTMrenewzonestyle{#1}{#4}%
740     \csdef{@dtm@style@#1}{%
741       \csuse{@dtm@datestyle@#1}%
742       \csuse{@dtm@timestyle@#1}%
743       \csuse{@dtm@zonestyle@#1}%
744       #5%
745     }%
746   }%
747   {%
748     \PackageError{datetime2}{Style ‘#1’ doesn’t exist}{}%
749   }%
750 }

```

`\DTMprovidestyle` `\DTMprovidestyle{<label>}{<date style definition>}{<time style definition>}{<zone style definition>}{<full format definition>}`

Defines a full style if it doesn't already exist.

```

751 \newcommand*{\DTMprovidestyle}[5]{%
752   \DTMifhasstyle{#1}%
753   {%
754   }%

```

```

755  {%
756    \DTMprovideddatestyle{#1}{#2}%
757    \DTMprovidetimestyle{#1}{#3}%
758    \DTMprovidezonestyle{#1}{#4}%
759    \csdef{@dtm@style@#1}{%
760      \csuse{@dtm@datestyle@#1}%
761      \csuse{@dtm@timestyle@#1}%
762      \csuse{@dtm@zonestyle@#1}%
763      #5%
764    }%
765  }%
766 }

```

DTMsetdatestyle

```

767 \newrobustcmd*{\DTMsetdatestyle}[1]{%
768   \ifcsdef{@dtm@datestyle@#1}%
769   {\csuse{@dtm@datestyle@#1}}%
770   {%
771     \PackageError{datettime2}{Date style ‘#1’ not defined}{}%
772   }%
773 }

```

DTMsettimestyle

```

774 \newrobustcmd*{\DTMsettimestyle}[1]{%
775   \ifcsdef{@dtm@timestyle@#1}%
776   {\csuse{@dtm@timestyle@#1}}%
777   {%
778     \PackageError{datettime2}{Time style ‘#1’ not defined}{}%
779   }%
780 }

```

DTMsetzonestyle

```

781 \newrobustcmd*{\DTMsetzonestyle}[1]{%
782   \ifcsdef{@dtm@zonestyle@#1}%
783   {\csuse{@dtm@zonestyle@#1}}%
784   {%
785     \PackageError{datettime2}{Zone style ‘#1’ not defined}{}%
786   }%
787 }

```

```

\DTMtryregional \DTMtryregional[<lang>]{<lang code cs>}{<country code cs>}

```

Tries to see if the regional style can be set. Takes into account the useregional setting. The starred version expects the arguments to be control sequences. If both are blank or undefined nothing happens. The optional argument may be the root language label, which is used if no style exists for the ISO codes.

```

788 \newcommand*\DTMtryregional}{%
789   \ifstar\s@dtm@tryregional\@dtm@tryregional
790 }

```

dtm@tryregional

```

791 \newcommand*\@dtm@tryregional}[3] []{%
792   \edef\@dtm@langcode{#2}%
793   \edef\@dtm@countrycode{#3}%
794   \s@dtm@tryregional[#1]{\@dtm@langcode}{\@dtm@countrycode}%
795 }

```

dtm@tryregional

```

796 \newcommand*\s@dtm@tryregional}[3] []{%
797   \def\@dtm@thisstyle{}%
798   \edef\@dtm@root{#1}%
799   \ifdefempty{#2}{}%
800   {%
801     \let\@dtm@thisstyle#2%

```

Determine the root language name if not already provided in the optional argument.

```

802   \ifdefempty\@dtm@root
803   {%
804     \edef\@dtm@root{\TrackedLanguageFromIsoCode{639-1}{#2}}%
805     \ifdefempty\@dtm@root
806     {%
807       \edef\@dtm@root{\TrackedLanguageFromIsoCode{639-2}{#2}}%
808     }%
809   }%
810 }%
811 {}%
812 }%
813 \ifdefempty{#3}{}%
814 {%
815   \ifdefempty\@dtm@thisstyle
816   {\let\@dtm@thisstyle#3}%
817   {\eappto\@dtm@thisstyle{-#3}}%
818 }%
819 \ifdefempty\@dtm@thisstyle
820 {}%
821 {%
822   \DTMifcaseregional
823   {}%
824   {%
825     \DTMifhasstyle{\@dtm@thisstyle}%
826     {%
827       \csuse{\@dtm@style@\@dtm@thisstyle}%
828     }%
829     {%
830       \ifdefempty\@dtm@root
831       {}%

```

```

832     {%
833         \DTMifhasstyle{\@dtm@root}%
834     {%
835         \csuse{@dtm@style@\@dtm@root}%
836     }%
837     {}%
838     }%
839 }%
840 }%
841 {%
842     \DTMifhasstyle{\@dtm@thisstyle-numeric}%
843     {%
844         \csuse{@dtm@style@\@dtm@thisstyle-numeric}%
845     }%
846     {%
847         \ifdefempty\@dtm@root
848         {}%
849         {%
850             \DTMifhasstyle{\@dtm@root-numeric}%
851             {%
852                 \csuse{@dtm@style@\@dtm@root-numeric}%
853             }%
854             {}%
855         }%
856     }%
857 }%
858 }%
859 }

```

\DTMsetregional

```

860 \newcommand*{\DTMsetregional}[1][text]{%
861     \DTMsetup{useregional=#1}%
862     \ifstrequal{#1}{false}%
863     {%
864         \DTMsetstyle{default}%
865     }%
866     {%
867         \ifcsdef{date\languagename}
868         {%
869             \csuse{date\languagename}
870         }%
871         {%

```

Iterate through dialect list.

```

872         \ForEachTrackedDialect{\@dtm@thisdialect}%
873         {%
874             \edef\@dtm@lang{\TrackedLanguageFromDialect\@dtm@thisdialect}%
875             \edef\@dtm@langcode{\TrackedIsoCodeFromLanguage{639-1}{\@dtm@lang}}%
876             \ifdefempty\@dtm@langcode
877             {%

```

```

878     \edef\@dtm@langcode{\TrackedIsoCodeFromLanguage{639-2}{\@dtm@lang}}%
879     }%
880     {}%
881     \edef\@dtm@countrycode{%
882         \TrackedIsoCodeFromLanguage{3166-1}{\@dtm@thisdialect}}%
883     \s@dtm@tryregional[\@dtm@lang]{\@dtm@langcode}{\@dtm@countrycode}%
884     }%
885     }%
886     }%
887 }

```

\DTMsetstyle

```

888 \newrobustcmd*{\DTMsetstyle}[1]{%
889     \DTMifhasstyle{#1}%
890     {\csuse{@dtm@style@#1}}%
891     {%
892         \let\dtm@unknownstyle\@dtm@unknownstyle
893         \ifcsdef{@dtm@datestyle#1}%
894             {\csuse{@dtm@datestyle@#1}\let\dtm@unknownstyle\@dtm@unknown@style}%
895             {\@dtm@warning{No date style ‘#1’ defined}}%
896         \ifcsdef{@dtm@timestyle#1}%
897             {\csuse{@dtm@timestyle@#1}\let\dtm@unknownstyle\@dtm@unknown@style}%
898             {\@dtm@warning{No time style ‘#1’ defined}}%
899         \ifcsdef{@dtm@zonestyle#1}%
900             {\csuse{@dtm@zonestyle@#1}\let\dtm@unknownstyle\@dtm@unknown@style}%
901             {\@dtm@warning{No zone style ‘#1’ defined}}%
902         \dtm@unknownstyle{#1}%
903     }%
904 }

```

tm@unknownstyle

```

905 \newcommand*{\@dtm@unknownstyle}[1]{%
906     \PackageError{datetime2}{Unknown style ‘#1’}{}%
907 }

```

m@unknown@style

```

908 \newcommand*{\@dtm@unknown@style}[1]{%
909     \@dtm@warning{No full style ‘#1’ defined}{}%
910 }

```

Define default style:

```

911 \DTMnewstyle
912 {default}%label
913 {% date style
914     \renewcommand*\DTMdisplaydate[4]{%
915         \number##1\DTMsep{yearmonth}\DTMtwodigits{##2}%
916         \DTMsep{monthday}\DTMtwodigits{##3}%
917     }%
918     \renewcommand*\DTMdisplaydate{\DTMdisplaydate}%

```

```

919 }%
920 {% time style
921   \renewcommand*\DTMdisplaytime[3]{%
922     \DTMtwodigits{##1}\DTMsep{hourmin}\DTMtwodigits{##2}%
923     \ifDTMshowseconds\DTMsep{minsec}\DTMtwodigits{##3}\fi
924   }%
925 }%
926 {% zone style
927   \renewcommand*\DTMdisplayzone}[2]{%
928     \ifboolexpe
929     { bool{DTMshowisoZ}
930       and test{\ifnumequal{##1}{0}}
931       and test{\ifnumequal{##2}{0}}
932     }%
933     {%
934       Z%
935     }%
936     {%

```

A space is needed after the conditional to prevent an unwanted `\relax` from being inserted if the time zone is negative.

```

937     \ifnum##1<0 \else+\fi\DTMtwodigits{##1}%
938     \ifDTMshowzoneminutes\DTMsep{hourmin}\DTMtwodigits{##2}\fi
939   }%
940 }%
941 }%
942 {% full style
943   \renewcommand*\DTMdisplay}[9]{%
944     \ifDTMshowdate
945     \DTMdisplaydate{##1}{##2}{##3}{##4}%
946     \DTMsep{datetime}%
947     \fi
948     \DTMdisplaytime
949     {##5}%
950     {##6}%
951     {##7}%
952     \ifDTMshowzone
953     \DTMsep{timezone}%
954     \DTMdisplayzone
955     {##8}%
956     {##9}%
957     \fi
958   }%
959   \renewcommand*\DTMDisplay}{\DTMdisplay}%
960 }

```

Define iso style which ignores the separator settings:

```

961 \DTMnewstyle
962 {iso}%label
963 {% date style

```



```

964 \renewcommand*\DTMdisplaydate[4]{%
965 \number##1-\DTMtwodigits{##2}-\DTMtwodigits{##3}%
966 }%
967 \renewcommand*\DTMdisplaydate{\DTMdisplaydate}%
968 }%
969 {% time style
970 \renewcommand*\DTMdisplaytime[3]{%
971 \DTMtwodigits{##1}:\DTMtwodigits{##2}%
972 \ifDTMshowseconds:\DTMtwodigits{##3}\fi
973 }%
974 }%
975 {% zone style
976 \renewcommand*\DTMdisplayzone}[2]{%
977 \ifboolexpe
978 { bool{DTMshowisoZ}
979 and test{\ifnumequal{##1}{0}}
980 and test{\ifnumequal{##2}{0}}
981 }%
982 {%
983 Z%
984 }%
985 {%

```

A space is needed after the conditional to prevent an unwanted `\relax` from being inserted if the time zone is negative.

```

986 \ifnum##1<0 \else+\fi\DTMtwodigits{##1}%
987 \ifDTMshowzoneminutes:\DTMtwodigits{##2}\fi
988 }%
989 }%
990 }%
991 {% full style
992 \renewcommand*\DTMdisplay}[9]{%
993 \ifDTMshowdate
994 \DTMdisplaydate{##1}{##2}{##3}{##4}%
995 T%
996 \fi
997 \DTMdisplaytime
998 {##5}%
999 {##6}%
1000 {##7}%
1001 \ifDTMshowzone
1002 \DTMdisplayzone
1003 {##8}%
1004 {##9}%
1005 \fi
1006 }%
1007 \renewcommand*\DTMdisplay{\DTMdisplay}%
1008 }

```

Define pdf style which converts into a format that can be used in `\pdfinfo`:

```

1009 \DTMnewstyle
1010 {pdf}%label
1011 {% date style
1012   \renewcommand*\DTMdisplaydate[4]{%
1013     D:\number##1 % space intended
1014     \DTMtwodigits{##2}\DTMtwodigits{##3}%
1015   }%
1016   \renewcommand*\DTMdisplaydate{\DTMdisplaydate}%
1017 }%
1018 {% time style
1019   \renewcommand*\DTMdisplaytime[3]{%
1020     \DTMtwodigits{##1}\DTMtwodigits{##2}\DTMtwodigits{##3}%
1021   }%
1022 }%
1023 {% zone style
1024   \renewcommand*\DTMdisplayzone[2]{%
1025     \ifboolexpe
1026     { bool{DTMshowisoZ}
1027       and test{\ifnumequal{##1}{0}}
1028       and test{\ifnumequal{##2}{0}}
1029     }%
1030     {%
1031       Z%
1032     }%
1033     {%

```

A space is needed after the conditional to prevent an unwanted `\relax` from being inserted if the time zone is negative.

```

1034     \ifnum##1<0 \else+\fi\DTMtwodigits{##1}'\DTMtwodigits{##2}'%
1035   }%
1036 }%
1037 }%
1038 {% full style
1039   \renewcommand*\DTMdisplay[9]{%
1040     \DTMdisplaydate{##1}{##2}{##3}{##4}%
1041     \DTMdisplaytime{##5}{##6}{##7}%
1042     \DTMdisplayzone{##8}{##9}%
1043   }%
1044   \renewcommand*\DTMdisplay{\DTMdisplay}%
1045 }

```

Define `yyyymd` style:

```

1046 \DTMnewstyle
1047 {yyyymd}%label
1048 {% date style
1049   \renewcommand*\DTMdisplaydate[4]{%
1050     \number##1
1051     \DTMsep{yearmonth}%
1052     \number##2
1053     \DTMsep{monthday}%

```

```

1054     \number##3
1055 }%
1056 \renewcommand*{\DTMdisplaydate}{\DTMdisplaydate}%
1057 }%
1058 {% time style
1059 \renewcommand*\DTMdisplaytime[3]{%
1060     \DTMtwodigits{##1}\DTMsep{hourmin}\DTMtwodigits{##2}%
1061     \ifDTMshowseconds\DTMsep{minsec}\DTMtwodigits{##3}\fi
1062 }%
1063 }%
1064 {% zone style
1065 \renewcommand*{\DTMdisplayzone}[2]{%
1066     \ifboolexpe
1067     { bool{DTMshowisoZ}
1068       and test{\ifnumequal{##1}{0}}
1069       and test{\ifnumequal{##2}{0}}
1070     }%
1071     {%
1072     Z%
1073     }%
1074     {%

```

A space is needed after the conditional to prevent an unwanted `\relax` from being inserted if the time zone is negative.

```

1075     \ifnum##1<0 \else+\fi\DTMtwodigits{##1}%
1076     \ifDTMshowzoneminutes\DTMsep{hourmin}\DTMtwodigits{##2}\fi
1077     }%
1078 }%
1079 }%
1080 {% full style
1081 \renewcommand*{\DTMdisplay}[9]{%
1082 \ifDTMshowdate
1083 \DTMdisplaydate{##1}{##2}{##3}{##4}%
1084 \DTMsep{datetime}%
1085 \fi
1086 \DTMdisplaytime
1087 {##5}%
1088 {##6}%
1089 {##7}%
1090 \ifDTMshowzone
1091 \DTMsep{timezone}%
1092 \DTMdisplayzone
1093 {##8}%
1094 {##9}%
1095 \fi
1096 }%
1097 \renewcommand*{\DTMdisplay}{\DTMdisplay}%
1098 }

```

Define `ddmmyyy` style:

```

1099 \DTMnewstyle
1100 {ddmmyyyy}%label
1101 {% date style
1102   \renewcommand*\DTMdisplaydate[4]{%
1103     \DTMtwdigits{##3}\DTMsep{monthday}%
1104     \DTMtwdigits{##2}\DTMsep{yearmonth}%
1105     \number##1
1106   }%
1107   \renewcommand*\DTMdisplaydate{\DTMdisplaydate}%
1108 }%
1109 {% time style
1110   \renewcommand*\DTMdisplaytime[3]{%
1111     \DTMtwdigits{##1}\DTMsep{hourmin}\DTMtwdigits{##2}%
1112     \ifDTMshowseconds\DTMsep{minsec}\DTMtwdigits{##3}\fi
1113   }%
1114 }%
1115 {% zone style
1116   \renewcommand*\DTMdisplayzone[2]{%
1117     \ifboolexpe
1118     { bool{DTMshowisoZ}
1119       and test{\ifnumequal{##1}{0}}
1120       and test{\ifnumequal{##2}{0}}
1121     }%
1122     {%
1123       Z%
1124     }%
1125     {%

```

A space is needed after the conditional to prevent an unwanted `\relax` from being inserted if the time zone is negative.

```

1126     \ifnum##1<0 \else+\fi\DTMtwdigits{##1}%
1127     \ifDTMshowzoneminutes\DTMsep{hourmin}\DTMtwdigits{##2}\fi
1128   }%
1129 }%
1130 }%
1131 {% full style
1132   \renewcommand*\DTMdisplay[9]{%
1133     \ifDTMshowdate
1134     \DTMdisplaydate{##1}{##2}{##3}{##4}%
1135     \DTMsep{datetime}%
1136     \fi
1137     \DTMdisplaytime
1138     {##5}%
1139     {##6}%
1140     {##7}%
1141     \ifDTMshowzone
1142     \DTMsep{timezone}%
1143     \DTMdisplayzone
1144     {##8}%
1145     {##9}%

```

```

1146   \fi
1147 }%
1148 \renewcommand*\DTMdisplay}{\DTMdisplay}%
1149 }

```

Define dmyyyy style:

```

1150 \DTMnewstyle
1151 {dmyyyy}%label
1152 {% date style
1153   \renewcommand*\DTMdisplaydate[4]{%
1154     \number##3
1155     \DTMsep{monthday}%
1156     \number##2
1157     \DTMsep{yearmonth}%
1158     \number##1
1159   }%
1160   \renewcommand*\DTMdisplaydate}{\DTMdisplaydate}%
1161 }%
1162 {% time style
1163   \renewcommand*\DTMdisplaytime[3]{%
1164     \DTMtwodigits{##1}\DTMsep{hourmin}\DTMtwodigits{##2}%
1165     \ifDTMshowseconds\DTMsep{minsec}\DTMtwodigits{##3}\fi
1166   }%
1167 }%
1168 {% zone style
1169   \renewcommand*\DTMdisplayzone}[2]{%
1170     \ifboolexpe
1171     { bool{DTMshowisoZ}
1172       and test{\ifnumequal{##1}{0}}
1173       and test{\ifnumequal{##2}{0}}
1174     }%
1175     {%
1176       Z%
1177     }%
1178     {%

```

A space is needed after the conditional to prevent an unwanted `\relax` from being inserted if the time zone is negative.

```

1179     \ifnum##1<0 \else+\fi\DTMtwodigits{##1}%
1180     \ifDTMshowzoneminutes\DTMsep{hourmin}\DTMtwodigits{##2}\fi
1181   }%
1182 }%
1183 }%
1184 {% full style
1185   \renewcommand*\DTMdisplay}[9]{%
1186     \ifDTMshowdate
1187     \DTMdisplaydate{##1}{##2}{##3}{##4}%
1188     \DTMsep{datetime}%
1189     \fi
1190     \DTMdisplaytime

```

```

1191     {##5}%
1192     {##6}%
1193     {##7}%
1194     \ifDTMshowzone
1195     \DTMsep{timezone}%
1196     \DTMdisplayzone
1197     {##8}%
1198     {##9}%
1199     \fi
1200 }%
1201 \renewcommand*\DTMDisplay{\DTMdisplay}%
1202 }

```

Define dmyy style:

```

1203 \DTMnewstyle
1204 {dmyy}%label
1205 {% date style
1206     \renewcommand*\DTMdisplaydate[4]{%
1207         \number##3 % space intended
1208         \DTMsep{monthday}%
1209         \number##2 % space intended
1210         \DTMsep{yearmonth}%
1211         \DTMtwodigits{##1}%
1212     }%
1213     \renewcommand*\DTMDisplaydate{\DTMdisplaydate}%
1214 }%
1215 {% time style
1216     \renewcommand*\DTMdisplaytime[3]{%
1217         \DTMtwodigits{##1}\DTMsep{hourmin}\DTMtwodigits{##2}%
1218         \ifDTMshowseconds\DTMsep{minsec}\DTMtwodigits{##3}\fi
1219     }%
1220 }%
1221 {% zone style
1222     \renewcommand*\DTMdisplayzone[2]{%
1223         \ifboolexpe
1224         { bool{DTMshowisoZ}
1225             and test{\ifnumequal{##1}{0}}
1226             and test{\ifnumequal{##2}{0}}
1227         }%
1228         {%
1229             Z%
1230         }%
1231         {%

```

A space is needed after the conditional to prevent an unwanted `\relax` from being inserted if the time zone is negative.

```

1232         \ifnum##1<0 \else+\fi\DTMtwodigits{##1}%
1233         \ifDTMshowzoneminutes\DTMsep{hourmin}\DTMtwodigits{##2}\fi
1234     }%
1235 }%

```

```

1236 }%
1237 {% full style
1238 \renewcommand*\DTMdisplay}[9]{%
1239 \ifDTMshowdate
1240 \DTMdisplaydate{##1}{##2}{##3}{##4}%
1241 \DTMsep{datetime}%
1242 \fi
1243 \DTMdisplaytime
1244 {##5}%
1245 {##6}%
1246 {##7}%
1247 \ifDTMshowzone
1248 \DTMsep{timezone}%
1249 \DTMdisplayzone
1250 {##8}%
1251 {##9}%
1252 \fi
1253 }%
1254 \renewcommand*\DTMDisplay}{\DTMdisplay}%
1255 }

```

Define ddmyy style:

```

1256 \DTMnewstyle
1257 {ddmyy}%label
1258 {% date style
1259 \renewcommand*\DTMdisplaydate[4]{%
1260 \DTMtwodigits{##3}\DTMsep{monthday}%
1261 \DTMtwodigits{##2}\DTMsep{yearmonth}%
1262 \DTMtwodigits{##1}%
1263 }%
1264 \renewcommand*\DTMDisplaydate}{\DTMdisplaydate}%
1265 }%
1266 {% time style
1267 \renewcommand*\DTMdisplaytime[3]{%
1268 \DTMtwodigits{##1}\DTMsep{hourmin}\DTMtwodigits{##2}%
1269 \ifDTMshowseconds\DTMsep{minsec}\DTMtwodigits{##3}\fi
1270 }%
1271 }%
1272 {% zone style
1273 \renewcommand*\DTMdisplayzone}[2]{%
1274 \ifboolexpe
1275 { bool{DTMshowisoZ}
1276 and test{\ifnumequal{##1}{0}}
1277 and test{\ifnumequal{##2}{0}}
1278 }%
1279 {%
1280 Z%
1281 }%
1282 {%

```

A space is needed after the conditional to prevent an unwanted `\relax` from being inserted if the time zone is negative.

```

1283     \ifnum##1<0 \else+ \fi\DTMtwdigits{##1}%
1284     \ifDTMshowzoneminutes\DTMsep{hourmin}\DTMtwdigits{##2}\fi
1285     }%
1286   }%
1287 }%
1288 {% full style
1289 \renewcommand*\DTMdisplay}[9]{%
1290 \ifDTMshowdate
1291 \DTMdisplaydate{##1}{##2}{##3}{##4}%
1292 \DTMsep{datetime}%
1293 \fi
1294 \DTMdisplaytime
1295 {##5}%
1296 {##6}%
1297 {##7}%
1298 \ifDTMshowzone
1299 \DTMsep{timezone}%
1300 \DTMdisplayzone
1301 {##8}%
1302 {##9}%
1303 \fi
1304 }%
1305 \renewcommand*\DTMDisplay{\DTMdisplay}%
1306 }

```

Define `mmddyyyy` style:

```

1307 \DTMnewstyle
1308 {mmddyyyy}%label
1309 {% date style
1310 \renewcommand*\DTMdisplaydate[4]{%
1311 \DTMtwdigits{##2}\DTMsep{monthday}%
1312 \DTMtwdigits{##3}\DTMsep{dayyear}%
1313 \number##1
1314 }%
1315 \renewcommand*\DTMDisplaydate{\DTMdisplaydate}%
1316 }%
1317 {% time style
1318 \renewcommand*\DTMdisplaytime[3]{%
1319 \DTMtwdigits{##1}\DTMsep{hourmin}\DTMtwdigits{##2}%
1320 \ifDTMshowseconds\DTMsep{minsec}\DTMtwdigits{##3}\fi
1321 }%
1322 }%
1323 {% zone style
1324 \renewcommand*\DTMdisplayzone}[2]{%
1325 \ifboolexpe
1326 { bool{DTMshowisoZ}
1327 and test{\ifnumequal{##1}{0}}

```



```

1328     and test{\ifnumequal{##2}{0}}
1329     }%
1330     {%
1331     Z%
1332     }%
1333     {%

```

A space is needed after the conditional to prevent an unwanted `\relax` from being inserted if the time zone is negative.

```

1334     \ifnum##1<0 \else\fi\DTMtwodigits{##1}%
1335     \ifDTMshowzoneminutes\DTMsep{hourmin}\DTMtwodigits{##2}\fi
1336     }%
1337     }%
1338 }%
1339 {% full style
1340 \renewcommand*\DTMdisplay}[9]{%
1341 \ifDTMshowdate
1342 \DTMdisplaydate{##1}{##2}{##3}{##4}%
1343 \DTMsep{datetime}%
1344 \fi
1345 \DTMdisplaytime
1346 {##5}%
1347 {##6}%
1348 {##7}%
1349 \ifDTMshowzone
1350 \DTMsep{timezone}%
1351 \DTMdisplayzone
1352 {##8}%
1353 {##9}%
1354 \fi
1355 }%
1356 \renewcommand*\DTMDisplay}{\DTMdisplay}%
1357 }

```

Define `mdyyyy` style:

```

1358 \DTMnewstyle
1359 {mdyyyy}%label
1360 {% date style
1361 \renewcommand*\DTMdisplaydate[4]{%
1362 \number##2 % space intended
1363 \DTMsep{monthday}%
1364 \number##3 % space intended
1365 \DTMsep{dayyear}%
1366 \number##1 % space intended
1367 }%
1368 \renewcommand*\DTMDisplaydate}{\DTMdisplaydate}%
1369 }%
1370 {% time style
1371 \renewcommand*\DTMdisplaytime[3]{%
1372 \DTMtwodigits{##1}\DTMsep{hourmin}\DTMtwodigits{##2}%

```

```

1373     \ifDTMshowseconds\DTMsep{minsec}\DTMtwodigits{##3}\fi
1374 }%
1375 }%
1376 {% zone style
1377 \renewcommand*\DTMdisplayzone}[2]{%
1378   \ifboolexpe
1379   { bool{DTMshowisoZ}
1380     and test{\ifnumequal{##1}{0}}
1381     and test{\ifnumequal{##2}{0}}
1382   }%
1383   {%
1384     Z%
1385   }%
1386   {%

```

A space is needed after the conditional to prevent an unwanted `\relax` from being inserted if the time zone is negative.

```

1387     \ifnum##1<0 \else+\fi\DTMtwodigits{##1}%
1388     \ifDTMshowzoneminutes\DTMsep{hourmin}\DTMtwodigits{##2}\fi
1389   }%
1390 }%
1391 }%
1392 {% full style
1393 \renewcommand*\DTMdisplay}[9]{%
1394   \ifDTMshowdate
1395     \DTMdisplaydate{##1}{##2}{##3}{##4}%
1396     \DTMsep{datetime}%
1397   \fi
1398   \DTMdisplaytime
1399     {##5}%
1400     {##6}%
1401     {##7}%
1402   \ifDTMshowzone
1403     \DTMsep{timezone}%
1404     \DTMdisplayzone
1405       {##8}%
1406       {##9}%
1407   \fi
1408 }%
1409 \renewcommand*\DTMdisplay}{\DTMdisplay}%
1410 }

```

Define `mdyy` style:

```

1411 \DTMnewstyle
1412 {mdyy}%label
1413 {% date style
1414   \renewcommand*\DTMdisplaydate[4]{%
1415     \number##2 % space intended
1416     \DTMsep{monthday}%
1417     \number##3 % space intended

```

```

1418     \DTMsep{dayyear}%
1419     \DTMtwodigits{##1}%
1420 }%
1421 \renewcommand*\DTMdisplaydate{\DTMdisplaydate}%
1422 }%
1423 {% time style
1424     \renewcommand*\DTMdisplaytime[3]{%
1425         \DTMtwodigits{##1}\DTMsep{hourmin}\DTMtwodigits{##2}%
1426         \ifDTMshowseconds\DTMsep{minsec}\DTMtwodigits{##3}\fi
1427     }%
1428 }%
1429 {% zone style
1430 \renewcommand*\DTMdisplayzone}[2]{%
1431     \ifboolexpe
1432     { bool{DTMshowisoZ}
1433       and test{\ifnumequal{##1}{0}}
1434       and test{\ifnumequal{##2}{0}}
1435     }%
1436     {%
1437         Z%
1438     }%
1439     {%

```

A space is needed after the conditional to prevent an unwanted `\relax` from being inserted if the time zone is negative.

```

1440     \ifnum##1<0 \else+\fi\DTMtwodigits{##1}%
1441     \ifDTMshowzoneminutes\DTMsep{hourmin}\DTMtwodigits{##2}\fi
1442 }%
1443 }%
1444 }%
1445 {% full style
1446 \renewcommand*\DTMdisplay}[9]{%
1447     \ifDTMshowdate
1448     \DTMdisplaydate{##1}{##2}{##3}{##4}%
1449     \DTMsep{datetime}%
1450     \fi
1451     \DTMdisplaytime
1452     {##5}%
1453     {##6}%
1454     {##7}%
1455     \ifDTMshowzone
1456     \DTMsep{timezone}%
1457     \DTMdisplayzone
1458     {##8}%
1459     {##9}%
1460     \fi
1461 }%
1462 \renewcommand*\DTMdisplay{\DTMdisplay}%
1463 }

```

Define mmddy style:

```
1464 \DTMnewstyle
1465 {mmddy}%label
1466 {% date style
1467   \renewcommand*\DTMdisplaydate[4]{%
1468     \DTMtwodigits{##2}\DTMsep{monthday}%
1469     \DTMtwodigits{##3}\DTMsep{dayyear}%
1470     \DTMtwodigits{##1}%
1471   }%
1472   \renewcommand*\DTMdisplaydate{\DTMdisplaydate}%
1473 }%
1474 {% time style
1475   \renewcommand*\DTMdisplaytime[3]{%
1476     \DTMtwodigits{##1}\DTMsep{hourmin}\DTMtwodigits{##2}%
1477     \ifDTMshowseconds\DTMsep{minsec}\DTMtwodigits{##3}\fi
1478   }%
1479 }%
1480 {% zone style
1481   \renewcommand*\DTMdisplayzone[2]{%
1482     \ifboolexpe
1483       { bool{DTMshowisoZ}
1484         and test{\ifnumequal{##1}{0}}
1485         and test{\ifnumequal{##2}{0}}
1486       }%
1487     {%
1488       Z%
1489     }%
1490   }%
```

A space is needed after the conditional to prevent an unwanted `\relax` from being inserted if the time zone is negative.

```
1491     \ifnum##1<0 \else+\fi\DTMtwodigits{##1}%
1492     \ifDTMshowzoneminutes\DTMsep{hourmin}\DTMtwodigits{##2}\fi
1493   }%
1494 }%
1495 }%
1496 {% full style
1497   \renewcommand*\DTMdisplay}[9]{%
1498     \ifDTMshowdate
1499       \DTMdisplaydate{##1}{##2}{##3}{##4}%
1500       \DTMsep{datetime}%
1501     \fi
1502     \DTMdisplaytime
1503     {##5}%
1504     {##6}%
1505     {##7}%
1506     \ifDTMshowzone
1507       \DTMsep{timezone}%
1508     \DTMdisplayzone
```

```

1509     {##8}%
1510     {##9}%
1511     \fi
1512 }%
1513 \renewcommand*\DTMdisplay{\DTMdisplay}%
1514 }

    Define hmmmss time style
1515 \DTMnewtimestyle
1516 {hmmmss}% label
1517 {%
1518     \renewcommand*\DTMdisplaytime[3]{%
1519         \number##1
1520         \DTMsep{hourmin}\DTMtwodigits{##2}%
1521         \ifDTMshowseconds\DTMsep{minsec}\DTMtwodigits{##3}\fi
1522     }%
1523 }%

```

```

    Define map zone style
1524 \DTMnewzonestyle
1525 {map}% label
1526 {%
1527     \renewcommand*\DTMdisplaytime[3]{%
1528         \DTMusezonemapordefault{##1}{##2}%
1529     }%
1530 }%

```

```

    Define hhmm zone style
1531 \DTMnewzonestyle
1532 {hhmm}% label
1533 {%
1534     \renewcommand*\DTMdisplaytime[3]{%

```

A space is needed after the conditional to prevent an unwanted `\relax` from being inserted if the time zone is negative.

```

1535     \ifnum##1<0 \else+\fi\DTMtwodigits{##1}%
1536     \ifDTMshowzoneminutes\DTMsep{hourmin}\DTMtwodigits{##2}\fi
1537 }%
1538 }

```

11.1.3 Saving and Using Dates

Date and time information is stored in control sequences in the form `\@dtm@<label>@<tag>`, where *<label>* is the label uniquely identifying the information and *<tag>* is the element (year, month, day, dow, hour, minute, second, TZhour and TZminute). Missing information is stored as 0 (except for the day of week, which is stored as -1).

`\DTMsavedate` Save the date specified in the format `<yyyy>-<mm>-<dd>`. `\expandafter` is used in case the argument is a control sequence storing the date. This will redefine an existing saved date with the same label. The first argument is the label.

```

1539 \newrobustcmd*{\DTMsavedate}[2]{%
1540 \expandafter\@dtm@parsedate#2\@dtm@endparsedate
1541 \cslet{\@dtm@#1@year}{\@dtm@year}%
1542 \cslet{\@dtm@#1@month}{\@dtm@month}%
1543 \cslet{\@dtm@#1@day}{\@dtm@day}%
1544 \cslet{\@dtm@#1@dow}{\@dtm@dow}%
1545 \ifcsundef{\@dtm@#1@hour}{\csdef{\@dtm@#1@hour}{0}}{}%
1546 \ifcsundef{\@dtm@#1@minute}{\csdef{\@dtm@#1@minute}{0}}{}%
1547 \ifcsundef{\@dtm@#1@second}{\csdef{\@dtm@#1@second}{0}}{}%
1548 \ifcsundef{\@dtm@#1@TZhour}{\csdef{\@dtm@#1@TZhour}{0}}{}%
1549 \ifcsundef{\@dtm@#1@TZminute}{\csdef{\@dtm@#1@TZminute}{0}}{}%
1550 }

```

`\savenoparsedate` Save the date without parsing the $\langle YYYY \rangle$ - $\langle MM \rangle$ - $\langle DD \rangle$ format.

```

1551 \newrobustcmd*{\DTMsavenoparsedate}[5]{%
1552 \csedef{\@dtm@#1@year}{\number#2}%
1553 \csedef{\@dtm@#1@month}{\number#3}%
1554 \csedef{\@dtm@#1@day}{\number#4}%
1555 \csedef{\@dtm@#1@dow}{\number#5}%
1556 \ifcsundef{\@dtm@#1@hour}{\csdef{\@dtm@#1@hour}{0}}{}%
1557 \ifcsundef{\@dtm@#1@minute}{\csdef{\@dtm@#1@minute}{0}}{}%
1558 \ifcsundef{\@dtm@#1@second}{\csdef{\@dtm@#1@second}{0}}{}%
1559 \ifcsundef{\@dtm@#1@TZhour}{\csdef{\@dtm@#1@TZhour}{0}}{}%
1560 \ifcsundef{\@dtm@#1@TZminute}{\csdef{\@dtm@#1@TZminute}{0}}{}%
1561 }

```

`\DTMsavetime` Save the time specified in the format $\langle hh \rangle$: $\langle mm \rangle$: $\langle ss \rangle$. `\expandafter` is used in case the argument is a control sequence storing the date. This will redefine an existing saved date with the same label. The first argument is the label.

```

1562 \newrobustcmd*{\DTMsavetime}[2]{%
1563 \expandafter\@dtm@parsetime#2\@dtm@endparsetime
1564 \cslet{\@dtm@#1@hour}{\@dtm@hour}%
1565 \cslet{\@dtm@#1@minute}{\@dtm@minute}%
1566 \cslet{\@dtm@#1@second}{\@dtm@second}%
1567 \ifcsundef{\@dtm@#1@year}{\csdef{\@dtm@#1@year}{0}}{}%
1568 \ifcsundef{\@dtm@#1@month}{\csdef{\@dtm@#1@month}{0}}{}%
1569 \ifcsundef{\@dtm@#1@day}{\csdef{\@dtm@#1@day}{0}}{}%
1570 \ifcsundef{\@dtm@#1@dow}{\csdef{\@dtm@#1@dow}{-1}}{}%
1571 \ifcsundef{\@dtm@#1@TZhour}{\csdef{\@dtm@#1@TZhour}{0}}{}%
1572 \ifcsundef{\@dtm@#1@TZminute}{\csdef{\@dtm@#1@TZminute}{0}}{}%
1573 }

```

`\DTMsavetimezn` Save the time (including zone) specified in the format $\langle hh \rangle$: $\langle mm \rangle$: $\langle ss \rangle$ $\langle tzh \rangle$: $\langle tzm \rangle$. `\expandafter` is used in case the argument is a control sequence storing the date. This will redefine an existing saved date with the same label. The first argument is the label.

```

1574 \newrobustcmd*{\DTMsavetimezn}[2]{%
1575 \expandafter\@dtm@parsetimezn#2\@dtm@endparsetimezn
1576 \cslet{\@dtm@#1@hour}{\@dtm@hour}%

```

```

1577 \cslet{@dtm@#1@minute}{\@dtm@minute}%
1578 \cslet{@dtm@#1@second}{\@dtm@second}%
1579 \cslet{@dtm@#1@TZhour}{\@dtm@timezonehour}%
1580 \cslet{@dtm@#1@TZminute}{\@dtm@timezoneminute}%
1581 \ifcsundef{@dtm@#1@year}{\csdef{@dtm@#1@year}{0}}{}%
1582 \ifcsundef{@dtm@#1@month}{\csdef{@dtm@#1@month}{0}}{}%
1583 \ifcsundef{@dtm@#1@day}{\csdef{@dtm@#1@day}{0}}{}%
1584 \ifcsundef{@dtm@#1@dow}{\csdef{@dtm@#1@dow}{-1}}{}%
1585 }

```

`\DTMsavetimestamp` Save the time (including zone) specified in the format `<YYYY>-<MM>-<DD>T<hh>:<mm>:<ss><time zone>`

The first argument is the label.

```

1586 \newrobustcmd*{\DTMsavetimestamp}[2]{%
1587 \expandafter\@dtm@parsetimestamp#2\@dtm@endparsetimestamp
1588 \cslet{@dtm@#1@year}{\@dtm@year}%
1589 \cslet{@dtm@#1@month}{\@dtm@month}%
1590 \cslet{@dtm@#1@day}{\@dtm@day}%
1591 \cslet{@dtm@#1@dow}{\@dtm@dow}%
1592 \cslet{@dtm@#1@hour}{\@dtm@hour}%
1593 \cslet{@dtm@#1@minute}{\@dtm@minute}%
1594 \cslet{@dtm@#1@second}{\@dtm@second}%
1595 \cslet{@dtm@#1@TZhour}{\@dtm@timezonehour}%
1596 \cslet{@dtm@#1@TZminute}{\@dtm@timezoneminute}%
1597 }

```

`\DTMsavenow` Save the date-time stamp in PDF format.

```

1598 \newrobustcmd*{\DTMsavepdftimestamp}[2]{%
1599 \edef\@dtm@tmp{#2}%
1600 \ifx\@dtm@tmp\empty
1601 \cslet{@dtm@#1@year}{0}%
1602 \cslet{@dtm@#1@month}{0}%
1603 \cslet{@dtm@#1@day}{0}%
1604 \cslet{@dtm@#1@dow}{0}%
1605 \cslet{@dtm@#1@hour}{0}%
1606 \cslet{@dtm@#1@minute}{0}%
1607 \cslet{@dtm@#1@second}{0}%
1608 \cslet{@dtm@#1@TZhour}{0}%
1609 \cslet{@dtm@#1@TZminute}{0}%
1610 \else
1611 \expandafter\@dtm@parsepdfdatetime\@dtm@tmp\@dtm@endparsepdfdatetime
1612 \cslet{@dtm@#1@year}{\@dtm@year}%
1613 \cslet{@dtm@#1@month}{\@dtm@month}%
1614 \cslet{@dtm@#1@day}{\@dtm@day}%
1615 \cslet{@dtm@#1@dow}{\@dtm@dow}%
1616 \cslet{@dtm@#1@hour}{\@dtm@hour}%
1617 \cslet{@dtm@#1@minute}{\@dtm@minute}%
1618 \cslet{@dtm@#1@second}{\@dtm@second}%
1619 \cslet{@dtm@#1@TZhour}{\@dtm@timezonehour}%

```

```

1620 \cslet{@dtm@#1@TZminute}{\@dtm@timezoneminute}%
1621 \fi
1622 }

```

`\DTMsavenow` Save the current time.

```

1623 \newrobustcmd{\DTMsavenow}[1]{%
1624 \cslet{@dtm@#1@year}{\@dtm@currentyear}%
1625 \cslet{@dtm@#1@month}{\@dtm@currentmonth}%
1626 \cslet{@dtm@#1@day}{\@dtm@currentday}%
1627 \cslet{@dtm@#1@dow}{\@dtm@currentdow}%
1628 \cslet{@dtm@#1@hour}{\@dtm@currenthour}%
1629 \cslet{@dtm@#1@minute}{\@dtm@currentminute}%
1630 \cslet{@dtm@#1@second}{\@dtm@currentsecond}%
1631 \cslet{@dtm@#1@TZhour}{\@dtm@currenttimezonehour}%
1632 \cslet{@dtm@#1@TZminute}{\@dtm@currenttimezoneminute}%
1633 }

```

`\DTMmakeglobal` Globally set the stored information.

```

1634 \newrobustcmd{\DTMmakeglobal}[1]{%
1635 \global\csletcs{@dtm@#1@year}{@dtm@#1@year}%
1636 \global\csletcs{@dtm@#1@month}{@dtm@#1@month}%
1637 \global\csletcs{@dtm@#1@day}{@dtm@#1@day}%
1638 \global\csletcs{@dtm@#1@dow}{@dtm@#1@dow}%
1639 \global\csletcs{@dtm@#1@hour}{@dtm@#1@hour}%
1640 \global\csletcs{@dtm@#1@minute}{@dtm@#1@minute}%
1641 \global\csletcs{@dtm@#1@second}{@dtm@#1@second}%
1642 \global\csletcs{@dtm@#1@TZhour}{@dtm@#1@TZhour}%
1643 \global\csletcs{@dtm@#1@TZminute}{@dtm@#1@TZminute}%
1644 }

```

Expandable ways of fetching saved data. (No check for existence performed.) The argument is the label.

`\DTMfetchyear`

```

1645 \newcommand*{\DTMfetchyear}[1]{\csname @dtm@#1@year\endcsname}

```

`\DTMfetchmonth`

```

1646 \newcommand*{\DTMfetchmonth}[1]{\csname @dtm@#1@month\endcsname}

```

`\DTMfetchday`

```

1647 \newcommand*{\DTMfetchday}[1]{\csname @dtm@#1@day\endcsname}

```

`\DTMfetchdow`

```

1648 \newcommand*{\DTMfetchdow}[1]{\csname @dtm@#1@dow\endcsname}

```

`\DTMfetchhour`

```

1649 \newcommand*{\DTMfetchhour}[1]{\csname @dtm@#1@hour\endcsname}

```


`\DTMfetchminute`

```
1650 \newcommand*\DTMfetchminute[1]{\csname @dtm@#1@minute\endcsname}
```

`\DTMfetchsecond`

```
1651 \newcommand*\DTMfetchsecond[1]{\csname @dtm@#1@second\endcsname}
```

`\DTMfetchTZhour`

```
1652 \newcommand*\DTMfetchTZhour[1]{\csname @dtm@#1@TZhour\endcsname}
```

`\DTMfetchTZminute`

```
1653 \newcommand*\DTMfetchTZminute[1]{\csname @dtm@#1@TZminute\endcsname}
```

`\DTMusedate` `\DTMusedate{<label>}`

Displays the previously saved date using `\DTMdisplaydate`.

```
1654 \newcommand*\DTMusedate[1]{%
1655   \ifcsundef{@dtm@#1@year}%
1656   {%
1657     \PackageError{datetime2}{Undefined date ‘#1’}{%
1658     }%
1659   }%
1660   \DTMdisplaydate
1661     {\csname @dtm@#1@year\endcsname}%
1662     {\csname @dtm@#1@month\endcsname}%
1663     {\csname @dtm@#1@day\endcsname}%
1664     {\csname @dtm@#1@dow\endcsname}%
1665   }%
1666 }%
```

`\DTMUsedate` `\DTMUsedate{<label>}`

Displays the previously saved date using `\DTMDisplaydate`.

```
1667 \newcommand*\DTMUsedate[1]{%
1668   \ifcsundef{@dtm@#1@year}%
1669   {%
1670     \PackageError{datetime2}{Undefined date ‘#1’}{%
1671     }%
1672   }%
1673   \DTMDisplaydate
1674     {\csname @dtm@#1@year\endcsname}%
1675     {\csname @dtm@#1@month\endcsname}%
1676     {\csname @dtm@#1@day\endcsname}%
1677     {\csname @dtm@#1@dow\endcsname}%
1678   }%
```

```
1678 }%
1679 }%
```

`\DTMusetime` `\DTMusetime{<label>}`

Displays the previously saved time using `\DTMdisplaytime`.

```
1680 \newcommand*\DTMusetime[1]{%
1681   \ifcsundef{@dtm@#1@hour}%
1682   {%
1683     \PackageError{datetime2}{Undefined time ‘#1’}{}%
1684   }%
1685   {%
1686     \DTMdisplaytime
1687     {\csname @dtm@#1@hour\endcsname}%
1688     {\csname @dtm@#1@minute\endcsname}%
1689     {\csname @dtm@#1@second\endcsname}%
1690   }%
1691 }%
```

`\DTMusezone` `\DTMusezone{<label>}`

Displays the previously saved date using `\DTMdisplayzone`.

```
1692 \newcommand*\DTMusezone[1]{%
1693   \ifcsundef{@dtm@#1@TZhour}%
1694   {%
1695     \PackageError{datetime2}{Undefined time ‘#1’}{}%
1696   }%
1697   {%
1698     \DTMdisplayzone
1699     {\csname @dtm@#1@TZhour\endcsname}%
1700     {\csname @dtm@#1@TZminute\endcsname}%
1701   }%
1702 }%
```

`\DTMuse` `\DTMuse{<label>}`

Displays the previously saved date and time.

```
1703 \newcommand*\DTMuse[1]{%
1704   \ifcsundef{@dtm@#1@year}%
1705   {%
1706     \PackageError{datetime2}{Undefined date-time ‘#1’}{}%
```

```

1707 }%
1708 {%
1709 \DTMdisplay
1710 {\csname @dtm@#1@year\endcsname}%
1711 {\csname @dtm@#1@month\endcsname}%
1712 {\csname @dtm@#1@day\endcsname}%
1713 {\csname @dtm@#1@dow\endcsname}%
1714 {\csname @dtm@#1@hour\endcsname}%
1715 {\csname @dtm@#1@minute\endcsname}%
1716 {\csname @dtm@#1@second\endcsname}%
1717 {\csname @dtm@#1@TZhour\endcsname}%
1718 {\csname @dtm@#1@TZminute\endcsname}%
1719 }%
1720 }%

```

`\DTMUse` `\DTMUse{<label>}`

Displays the previously saved date and time.

```

1721 \newcommand*\DTMUse[1]{%
1722 \ifcsundef{@dtm@#1@year}%
1723 {%
1724 \PackageError{datetime2}{Undefined date-time ‘#1’}{}%
1725 }%
1726 {%
1727 \DTMdisplay
1728 {\csname @dtm@#1@year\endcsname}%
1729 {\csname @dtm@#1@month\endcsname}%
1730 {\csname @dtm@#1@day\endcsname}%
1731 {\csname @dtm@#1@dow\endcsname}%
1732 {\csname @dtm@#1@hour\endcsname}%
1733 {\csname @dtm@#1@minute\endcsname}%
1734 {\csname @dtm@#1@second\endcsname}%
1735 {\csname @dtm@#1@TZhour\endcsname}%
1736 {\csname @dtm@#1@TZminute\endcsname}%
1737 }%
1738 }%

```

`\DTMifsaveddate` Determine if the given label has been assigned to a date, time and zone.

```

1739 \newcommand{\DTMifsaveddate}[3]{%
1740 \ifcsundef{@dtm@#1@year}{#3}{#2}%
1741 }

```

11.1.4 Language Module Loading

Define commands to load regional settings.

m@requiremodule Use tracklang interface to find the associated file for the given dialect.

```
1742 \newcommand*{\@dtm@requiremodule}[1]{%
1743   \IfTrackedLanguageFileExists{#1}%
1744   {datetime2-}% prefix
1745   {.ldf}% suffix
1746   {%
1747     \RequireDateTimeModule{\CurrentTrackedTag}%
1748   }%
1749   {%
1750     \@dtm@warning{Date-Time Language Module ‘#1’ not installed}%
1751   }%
1752 }
```

m@loadedregions List of loaded datetime2 language modules.

```
1753 \newcommand*{\@dtm@loadedregions}{}
```

eDateTimeModule Input the language file, if not already loaded. Should only be used with \@dtm@requiremodule which sets commands like \CurrentTrackedDialect. Since the language modules are loaded within \@dtm@requiremodule they may use this command to load dependent modules.

```
1754 \newcommand*{\RequireDateTimeModule}[1]{%
1755   \ifundef\CurrentTrackedDialect
1756   {%
1757     \PackageError{datetime2}%
1758     {\string\RequireDateTimeModule\space not permitted here}%
1759     {This command is only permitted inside datetime2 language
1760     modules.}%
1761   }%
1762   {%
1763     \ifcsundef{ver@datetime2-#1.ldf}%
1764     {%
1765       \input{datetime2-#1.ldf}%
1766       \ifdefempty\@dtm@loadedregions
1767       {%
1768         \edef\@dtm@loadedregions{#1}%
1769       }%
1770     }%
1771     \edef\@dtm@loadedregions{\@dtm@loadedregions,#1}%
1772   }%
```

In case a synonym is also used, add a mapping from the module name to the current tracked dialect.

```
1773   \csedef{@dtm@moddialectmap@#1}{\CurrentTrackedDialect}%
1774   }%
1775   {%
```

The module has already been loaded, but the current tracked dialect might be a synonym for a different language label that might've already loaded the module. If \date<*dialect*> exists, this needs to be set.

```

1776 \ifcsdef{date\CurrentTrackedDialect}
1777 {%
1778 \letcs{\@dtm@otherdialect}{@dtm@moddialectmap@#1}%
1779 \edef\@dtm@thisdialect{\CurrentTrackedDialect}%
1780 \ifdefequal\@dtm@thisdialect\@dtm@otherdialect
1781 {}%
1782 {%
1783 \ifcsdef{date\@dtm@otherdialect}%
1784 {%
1785 \csletcs{date\@dtm@thisdialect}{date\@dtm@otherdialect}%
1786 }%
1787 {}%
1788 }%
1789 }%
1790 {}%
1791 }%

```

In case it's needed, create a mapping between the dialect name and the module name.

```

1792 \csedef{@dtm@dialectmodmap@\CurrentTrackedDialect}{#1}%
1793 }%
1794 }

```

alecttomodulemap `\DTMdialecttomodulemap{<dialect>}`

Expands to name of the module loaded with the given dialect name or `\relax` if no module has been loaded for the given dialect.

```

1795 \newcommand*{\DTMdialecttomodulemap}[1]{%
1796 \ifcsdef{ver@datetime2-#1.ldf}%
1797 {#1}%
1798 {\csname @dtm@dialectmodmap@#1\endcsname}%
1799 }

```

sDateTimeModule For use in language module to identify itself.

```

1800 \newcommand*{\ProvidesDateTimeModule}[1]{%
1801 \ProvidesFile{datetime2-#1.ldf}%
1802 }

```

\DTMusementmodule Provided for packages or documents that need to load a module. This shouldn't be used inside the `.ldf` files.

```

1803 \newcommand*{\DTMusementmodule}[2]{%
1804 \ifcsdef{@tracklang@add@#1}%
1805 {%
1806 \TrackPredefinedDialect{#1}%
1807 }%
1808 {}%
1809 \let\@dtm@org@dialect\CurrentTrackedDialect

```

```

1810 \def\CurrentTrackedDialect{#1}%
1811 \RequireDateTimeModule{#2}%
1812 \let\CurrentTrackedDialect\@dtm@org@dialect
1813 }

```

`\DTMdefkey` `\DTMdefkey{<region>}{<key>}[<default>]{<func>}`

Used by language modules to define a key.

```
1814 \newcommand*{\DTMdefkey}[1]{\define@key[dtm]{#1}}
```

`\DTMdefchoicekey` `\DTMdefchoicekey{<region>}{<key>}[<bin>]{<choice list>}{<default>}{<func>}`

Used by language modules to define a choice key.

```
1815 \newcommand*{\DTMdefchoicekey}[1]{\define@choicekey[dtm]{#1}}
```

`\DTMdefboolkey` `\DTMdefboolkey{<region>}[<mp>]{<key>}[<default>]{<func>}`

Used by language modules to define a boolean key.

```
1816 \newcommand*{\DTMdefboolkey}[1]{\define@boolkey[dtm]{#1}}
```

`\DTMifbool` `\DTMifbool{<region>}{<key>}{<>true>}{<>false>}`

Test boolean key that was defined using `\DTMdefboolkey`

```
1817 \newcommand*{\DTMifbool}[4]{\ifbool{dtm@#1@#2}{#3}{#4}}
```

`\DTMsetbool` `\DTMsetbool{<region>}{<key>}{<value>}`

Set boolean key that was defined using `\DTMdefboolkey`

```
1818 \newcommand*{\DTMsetbool}[3]{\setbool{dtm@#1@#2}{#3}}
```

`\DTMlangsetup` Set up options for language modules. The optional argument is a list of language/regions. If omitted all loaded regions are iterated over. (I'm not sure why `\setkeys` doesn't work if the same key is present in multiple families, so this iterates over the families instead.) The starred version doesn't warn on unknown keys.

```

1819 \newcommand*{\DTMlangsetup}{%
1820   \ifstar\s@DTMlangsetup\@DTMlangsetup}

  Unstarred version:
1821 \newcommand*{\@DTMlangsetup}[2][\@dtm@loadedregions]{%
1822   \@for\@dtm@region:=#1\do{%
1823     \setkeys*+[dtm]{\@dtm@region}{#2}%
1824     \ifdefempty\XKV@rm{%
1825       {%
1826         \@dtm@warning{Region ‘\@dtm@region’ has ignored
1827           \MessageBreak the following settings:\MessageBreak
1828           \XKV@rm
1829           ^^J}%
1830       }%
1831   }%
1832 }

```

Starred version:

```

1833 \newcommand*{\s@DTMlangsetup}[2][\@dtm@loadedregions]{%
1834   \@for\@dtm@region:=#1\do{%
1835     \setkeys*+[dtm]{\@dtm@region}{#2}%
1836   }%
1837 }

```

Now load all the required modules (if installed) using the tracklang interface. (Language packages, such as babel or polyglossia must be loaded before this.)

```

1838 \AnyTrackedLanguages
1839 {%
1840   \ForEachTrackedDialect{\this@dialect}%
1841   {%
1842     \@dtm@requiremodule\this@dialect
1843   }%
1844 }
1845 {%

```

No tracked languages. The default is already set up, so nothing to do here.

```
1846 }
```

Load datetime2-calc if required.

```
1847 \@dtm@usecalc
```

Use the style package option, if set.

```
1848 \ifdefempty\@dtm@initialstyle{\DTMsetstyle{\@dtm@initialstyle}}
```

11.2 datetime2-calc.sty code

```

1849 \NeedsTeXFormat{LaTeX2e}
1850 \ProvidesPackage{datetime2-calc}[2018/07/20 v1.5.3 (NLCT)]

```

Load other required packages

```
1851 \RequirePackage{pgfkeys}
1852 \RequirePackage{pgfcalendar}
```

11.2.1 Conversions and Calculations

`\@dtm@julianday` Register for storing Julian day number.

```
1853 \newcount\@dtm@julianday
```

`\@dtm@parsedate` Redefine `\@dtm@parsedate` so that it uses `pgfcalendar` to compute the required information. This allows for offsets, the use of last and also determine the day of week.

```
1854 \def\@dtm@parsedate#1-#2-#3\@dtm@endparsedate{%
1855   \pgfcalendardatetojulian{#1-#2-#3}\@dtm@julianday}%
1856   \pgfcalendarjuliantodate{\@dtm@julianday}\@dtm@year}\@dtm@month}\@dtm@day}%
1857   \pgfcalendarjuliantoweekday{\@dtm@julianday}\count@}%
1858   \edef\@dtm@dow{\number\count@}%
1859 }
```

Set the current day of week

`@dtm@currentdow`

```
1860 \pgfcalendardatetojulian
1861 {\@dtm@currentyear-\@dtm@currentmonth-\@dtm@currentday}%
1862 {\@dtm@julianday}%
1863 \pgfcalendarjuliantoweekday{\@dtm@julianday}\count@}%
1864 \edef\@dtm@currentdow{\number\count@}%
```

`DTMsavejulianday` Save the date obtained from the Julian day number.

```
1865 \newrobustcmd*{\DTMsavejulianday}[2]{%
1866   \pgfcalendarjuliantodate{#2}\@dtm@year}\@dtm@month}\@dtm@day}%
1867   \pgfcalendarjuliantoweekday{#2}\count@}%
1868   \csdef{@dtm@#1@dow}{\number\count@}%
1869   \cslet{@dtm@#1@year}\@dtm@year}%
1870   \cslet{@dtm@#1@month}\@dtm@month}%
1871   \cslet{@dtm@#1@day}\@dtm@day}%
1872   \ifcsundef{@dtm@#1@hour}{\csdef{@dtm@#1@hour}{0}}{}%
1873   \ifcsundef{@dtm@#1@minute}{\csdef{@dtm@#1@minute}{0}}{}%
1874   \ifcsundef{@dtm@#1@second}{\csdef{@dtm@#1@second}{0}}{}%
1875   \ifcsundef{@dtm@#1@TZhour}{\csdef{@dtm@#1@TZhour}{0}}{}%
1876   \ifcsundef{@dtm@#1@TZminute}{\csdef{@dtm@#1@TZminute}{0}}{}%
1877 }
```

`datetojulianday` Converts a saved date to a Julian day number. The first argument is the name referencing the saved date, the second is a count register in which to store the result.

```
1878 \newrobustcmd*{\DTMsaveddatetojulianday}[2]{%
1879   \ifcsundef{@dtm@#1@year}%
1880   {%
1881     \PackageError{datetime2-calc}{Unknown date ‘#1’}{}%
1882   }%
1883   {%
```



```

1884 \pgfcalendaratetojulian
1885   {\csname @dtm@#1@year\endcsname
1886   -\csname @dtm@#1@month\endcsname
1887   -\csname @dtm@#1@day\endcsname}
1888   {#2}%
1889 }%
1890 }

```

`offsettojulianday` Converts an offset from the saved date to a Julian day number. The first argument is the name referencing the saved date, the second is the offset increment and the third is a count register in which to store the result.

```

1891 \newrobustcmd*{\DTMsaveddateoffsettojulianday}[3]{%
1892   \ifcsundef{@dtm@#1@year}%
1893   {%
1894     \PackageError{datetime2-calc}{Unknown date '#1'}{}}%
1895   }%
1896   {%
1897     \pgfcalendaratetojulian
1898     {\csname @dtm@#1@year\endcsname
1899     -\csname @dtm@#1@month\endcsname
1900     -\csname @dtm@#1@day\endcsname
1901     +#2}
1902     {#3}%
1903   }%
1904 }

```

`\DTMifdate` Test a saved date using `\pgfcalendarifdate`

```

1905 \newrobustcmd*{\DTMifdate}[4]{%
1906   \ifcsundef{@dtm@#1@year}%
1907   {%
1908     \PackageError{datetime2-calc}{Unknown date '#1'}{}}%
1909   }%
1910   {%
1911     \pgfcalendarifdate
1912     {\csname @dtm@#1@year\endcsname
1913     -\csname @dtm@#1@month\endcsname
1914     -\csname @dtm@#1@day\endcsname}
1915     {#2}{#3}{#4}%
1916   }%
1917 }

```

`DTMsaveddatediff` Computes the difference between two saved dates. The result is stored in the third argument, which should be a count register.

```

1918 \newrobustcmd*{\DTMsaveddatediff}[3]{%
1919   \ifcsundef{@dtm@#1@year}%
1920   {%
1921     \PackageError{datetime2-calc}{Unknown date '#1'}{}}%
1922   }%
1923   {%

```

```

1924 \ifcsundef{@dtm@#2@year}%
1925 {%
1926 \PackageError{datetime2-calc}{Unknown date '#1'}{}}%
1927 }%
1928 {%
1929 \pgfcalendaratetotjulian
1930 {\csname @dtm@#1@year\endcsname
1931 -\csname @dtm@#1@month\endcsname
1932 -\csname @dtm@#1@day\endcsname}
1933 {#3}%
1934 \pgfcalendaratetotjulian
1935 {\csname @dtm@#2@year\endcsname
1936 -\csname @dtm@#2@month\endcsname
1937 -\csname @dtm@#2@day\endcsname}
1938 {\@dtm@julianday}%
1939 \advance#3 by -\@dtm@julianday\relax
1940 }
1941 }%
1942 }

```

`\DTMtozulu` Converts the datetime data referenced by the first argument into Zulu time and saves it to data referenced by the second argument.

```

1943 \newrobustcmd*{\DTMtozulu}[2]{%
1944 \ifcsundef{@dtm@#1@year}%
1945 {%
1946 \PackageError{datetime2-calc}{Unknown date '#1'}{}}%
1947 }%
1948 {%
1949 \DTMsaveaszulutime{#2}%
1950 {\DTMfetchyear{#1}}%
1951 {\DTMfetchmonth{#1}}%
1952 {\DTMfetchday{#1}}%
1953 {\DTMfetchhour{#1}}%
1954 {\DTMfetchminute{#1}}%
1955 {\DTMfetchsecond{#1}}%
1956 {\DTMfetchTZhour{#1}}%
1957 {\DTMfetchTZminute{#1}}%
1958 }%
1959 }

```

`\DTMsaveaszulutime` Converts the given datetime into Zulu (+00:00) and saves the result.

```

\DTMsavetozulutime{<name>}{<year>}{<month>}{<day>}{<hour>}
{<minute>}{<second>}{<tzh>}{<tzm>}

```

```

1960 \newrobustcmd*{\DTMsaveaszulutime}[9]{%
1961 \edef\@dtm@year{\number#2}%

```

```

1962 \edef\@dtm@month{\number#3}%
1963 \edef\@dtm@day{\number#4}%
1964 \edef\@dtm@hour{\number#5}%
1965 \edef\@dtm@minute{\number#6}%
1966 \edef\@dtm@second{\number#7}%
1967 \edef\@dtm@TZhour{\number#8}%
1968 \edef\@dtm@TZminute{\number#9}%
1969 \pgfcalendardatetojulian{\@dtm@year-\@dtm@month-\@dtm@day}{\@dtm@julianday}%

```

First adjust the minute offset if non-zero

```

1970 \ifnum\@dtm@TZminute=0\relax
1971 \else
1972 \count@=\@dtm@minute\relax

```

Add or subtract the offset minute

```

1973 \ifnum\@dtm@TZhour<0\relax
1974 \advance\count@ by \@dtm@TZminute\relax
1975 \else
1976 \advance\count@ by -\@dtm@TZminute\relax
1977 \fi
1978 \edef\@dtm@minute{\number\count@}%

```

Does the hour need adjusting?

```

1979 \ifnum\count@<0\relax
1980 \advance\count@ by 60\relax
1981 \edef\@dtm@minute{\number\count@}%

```

Need to subtract 1 from the hour but does the day need adjusting?

```

1982 \ifnum\@dtm@hour=0\relax
1983 \def\@dtm@hour{23}%

```

Day needs adjusting.

```

1984 \advance\@dtm@julianday by -1\relax
1985 \else

```

Subtract 1 from the hour

```

1986 \count@ = \@dtm@hour\relax
1987 \advance\count@ by -1\relax
1988 \edef\@dtm@hour{\number\count@}%
1989 \fi
1990 \else

```

Minute isn't negative. Is it ≥ 60 ?

```

1991 \ifnum\count@>59\relax
1992 \advance\count@ by -60\relax
1993 \edef\@dtm@minute{\number\count@}%

```

Add 1 to the hour

```

1994 \count@ = \@dtm@hour\relax
1995 \advance\count@ by 1\relax
1996 \edef\@dtm@hour{\number\count@}%

```

Does the day need adjusting?

```
1997     \ifnum\@dtm@hour=24\relax
1998     \def\@dtm@hour{00}%
1999     \advance\@dtm@julianday by 1\relax
2000     \fi
2001     \fi
2002     \fi
2003     \fi
```

Now adjust the hour offset if non-zero

```
2004     \ifnum\@dtm@TZhour=0\relax
2005     \else
2006     \count@=\@dtm@hour\relax
2007     \advance\count@ by -\@dtm@TZhour\relax
```

Does the day need adjusting?

```
2008     \ifnum\count@<0\relax
2009     \advance\count@ by 24\relax
2010     \edef\@dtm@hour{\number\count@}%
2011     \advance\@dtm@julianday by -1\relax
2012     \else
2013     \ifnum\count@>23\relax
2014     \advance\count@ by -24\relax
2015     \edef\@dtm@hour{\number\count@}%
2016     \advance\@dtm@julianday by 1\relax
2017     \else
2018     \edef\@dtm@hour{\number\count@}%
2019     \fi
2020     \fi
2021     \fi
2022     \pgfcalendarjuliantodate{\@dtm@julianday}{\@dtm@year}{\@dtm@month}{\@dtm@day}%
2023     \pgfcalendarjuliantoweekday{\@dtm@julianday}{\count@}%
```

Save the results.

```
2024     \csedef{\@dtm@#1@dow}{\number\count@}%
2025     \cslet{\@dtm@#1@year}{\@dtm@year}%
2026     \cslet{\@dtm@#1@month}{\@dtm@month}%
2027     \cslet{\@dtm@#1@day}{\@dtm@day}%
2028     \cslet{\@dtm@#1@hour}{\@dtm@hour}%
2029     \cslet{\@dtm@#1@minute}{\@dtm@minute}%
2030     \cslet{\@dtm@#1@second}{\@dtm@second}%
2031     \csdef{\@dtm@#1@TZhour}{0}%
2032     \csdef{\@dtm@#1@TZminute}{0}%
2033 }
```

11.2.2 Month and Weekday Names

These commands *should not* be used in date styles. One of the reasons for replacing `datetime` with `datetime2` was caused by styles using language-variable names in language-specific syntax resulting in a mismatch with the syntax of one language (or region) with a translation of

the month (and possibly weekday) name. These commands are provided for standalone use outside of styles. (Additionally, they're not expandable, which also makes them inappropriate for the styles that are expected to provide expandable dates.)

ifdianameexists

```

2034 \newcommand*{\dtm@ifdianameexists}[3]{%
2035 \IfTrackedDialect{\language}%
2036 {%
2037 \ifcsdef{DTM\TrackedLanguageFromDialect{\language}#1}%
2038 {#2}%
2039 {#3}%
2040 }%
2041 {#3}%
2042 }

```

\DTMmonthname

```

2043 \newrobustcmd{\DTMmonthname}[1]{%
    First check if \DTM<language>monthname exists.
2044 \ifcsdef{DTM\language monthname}%
2045 {%
    It exists, so use it.
2046 \csuse{DTM\language monthname}#{1}%
2047 }%
2048 {%
    Try obtaining the language name from the dialect using tracklang's interface.
2049 \dtm@ifdianameexists{monthname}%
2050 {%
    It exists, so use it.
2051 \csuse{DTM\TrackedLanguageFromDialect{\language}monthname}#{1}%
2052 }%
2053 {%
    Can't determine the language name macro. This may be because the actual name can't be
    determined or it could be because the relevant language module can't be loaded so use pgf's
    command instead, which also has limited language support.
2054 \dtmnamewarning{\DTMmonthname}%
2055 \pgfcalendarmonthname{#1}%
2056 }%
2057 }%
2058 }

```

\DTMMonthname

```

2059 \newrobustcmd{\DTMMonthname}[1]{%
    First check if \DTM<language>Monthname exists.
2060 \ifcsdef{DTM\language Monthname}%
2061 {%

```

It exists, so use it.

```
2062 \csuse{DTM\language Monthname}{#1}%  
2063 }%  
2064 {%
```

Try obtaining the language name from the dialect using tracklang's interface.

```
2065 \dtm@ifdianameexists{Monthname}%  
2066 {%
```

It exists, so use it.

```
2067 \csuse{DTM\TrackedLanguageFromDialect{\language}Monthname}{#1}%  
2068 }%  
2069 {%
```

Can't determine the language name macro. This could be because there's no upper case macro as the names always start with a capital (like English).

```
2070 \ifcsdef{DTM\language monthname}%  
2071 {%  
2072 \csuse{DTM\language monthname}{#1}%  
2073 }%  
2074 {%  
2075 \dtm@ifdianameexists{monthname}%  
2076 {%  
2077 \csuse{DTM\TrackedLanguageFromDialect{\language}monthname}{#1}%  
2078 }%  
2079 {%
```

Can't find no-case change version either, so use pgfcalendar command instead (which will need a case-change applied).

```
2080 \dtmnamewarning{DTMMonthname}%
```

If mfirstuc has been loaded, use it.

```
2081 \ifdef\emakefirstuc  
2082 {%  
2083 \emakefirstuc{\pgfcalendarmonthname{#1}}%  
2084 }%  
2085 {%
```

Hasn't been loaded, so just expand and apply \MakeUppercase:

```
2086 \protected@edef\dtm@tmp@monthname{\pgfcalendarmonthname{#1}}%  
2087 \expandafter\MakeUppercase\dtm@tmp@monthname  
2088 }%  
2089 }%  
2090 }%  
2091 }%  
2092 }%  
2093 }
```

Mshortmonthname

```
2094 \newrobustcmd{\DTMshortmonthname}[1]{%
```

First check if `\DTM<language>shortmonthname` exists.

```
2095 \ifcsdef{DTM\language shortmonthname}%  
2096 {%
```

It exists, so use it.

```
2097 \csuse{DTM\language shortmonthname}{#1}%  
2098 }%  
2099 {%
```

Try obtaining the language name from the dialect using `tracklang`'s interface.

```
2100 \dtm@ifdianameexists{shortmonthname}%  
2101 {%
```

It exists, so use it.

```
2102 \csuse{DTM\TrackedLanguageFromDialect{\language}shortmonthname}{#1}%  
2103 }%  
2104 {%
```

Can't determine the language name macro. This may be because the actual name can't be determined or it could be because the relevant language module can't be loaded so use `pgf`'s command instead, which also has limited language support.

```
2105 \dtmnamewarning{\DTMshortmonthname}%  
2106 \pgfcalendarmonthshortname{#1}%  
2107 }%  
2108 }%  
2109 }
```

MshortMonthname

```
2110 \newrobustcmd{\DTMshortMonthname}[1]{%
```

First check if `\DTM<language>shortMonthname` exists.

```
2111 \ifcsdef{DTM\language shortMonthname}%  
2112 {%
```

It exists, so use it.

```
2113 \csuse{DTM\language shortMonthname}{#1}%  
2114 }%  
2115 {%
```

Try obtaining the language name from the dialect using `tracklang`'s interface.

```
2116 \dtm@ifdianameexists{shortMonthname}%  
2117 {%
```

It exists, so use it.

```
2118 \csuse{DTM\TrackedLanguageFromDialect{\language}shortMonthname}{#1}%  
2119 }%  
2120 {%
```

Can't determine the language name macro. This could be because there's no upper case macro as the names always start with a capital (like English).

```
2121 \ifcsdef{DTM\language shortmonthname}%  
2122 {%
```

```

2123     \csuse{DTM\languagename shortmonthname}{#1}%
2124 }%
2125 {%
2126     \dtm@ifdianameexists{shortmonthname}%
2127     {%
2128         \csuse{DTM\TrackedLanguageFromDialect{\languagename}shortmonthname}%
2129         {#1}%
2130     }%
2131     {%

```

Can't find no-case change version either, so use pgfcalendar command instead (which will need a case-change applied).

```

2132     \dtmnamewarning{\DTMshortMonthname}%

```

If mfirstuc has been loaded, use it.

```

2133     \ifdef\emakefirstuc
2134     {%
2135         \emakefirstuc{\pgfcalendarmonthshortname{#1}}%
2136     }%
2137     {%

```

Hasn't been loaded, so just expand and apply \MakeUppercase:

```

2138     \protected@edef\dtm@tmp@monthname{\pgfcalendarmonthshortname{#1}}%
2139     \expandafter\MakeUppercase\dtm@tmp@monthname
2140     }%
2141     }%
2142     }%
2143     }%
2144     }%
2145 }

```

```

\dtmdayofweekindex \DTMcomputedayofweekindex{<date>}{<cs>}

```

This is for standalone use and shouldn't be used in any date styles (since the day of week index is already supplied). The result is stored in the supplied control sequence.

```

2146 \newrobustcmd*{\DTMcomputedayofweekindex}[2]{%
2147   \pgfcalendardatetojulian{#1}{\@dtm@julianday}%
2148   \pgfcalendarjuliantoweekday{\@dtm@julianday}{\count@}%
2149   \edef#2{\number\count@}%
2150 }

```

These are shared with the locale package, so they may already be defined.

```

\dtmMondayIndex

```

```

2151 \def\dtmMondayIndex{0}

```

```

\dtmTuesdayIndex

```

```

2152 \def\dtmTuesdayIndex{1}

```


mWednesdayIndex

```
2153 \def\dtmWednesdayIndex{2}
```

tmThursdayIndex

```
2154 \def\dtmThursdayIndex{3}
```

\dtmFridayIndex

```
2155 \def\dtmFridayIndex{4}
```

tmSaturdayIndex

```
2156 \def\dtmSaturdayIndex{5}
```

\dtmSundayIndex

```
2157 \def\dtmSundayIndex{6}
```

\DTMweekdayname

```
2158 \newrobustcmd{\DTMweekdayname}[1]{%
```

First check if `\DTM<language>weekdayname` exists.

```
2159 \ifcsdef{DTM\language weekdayname}%
```

```
2160 {%
```

It exists, so use it.

```
2161 \csuse{DTM\language weekdayname}{#1}%
```

```
2162 }%
```

```
2163 {%
```

Try obtaining the language name from the dialect using `tracklang`'s interface.

```
2164 \dtm@ifdianameexists{weekdayname}%
```

```
2165 {%
```

It exists, so use it.

```
2166 \csuse{DTM\TrackedLanguageFromDialect{\language}weekdayname}{#1}%
```

```
2167 }%
```

```
2168 {%
```

Can't determine the language name macro. This may be because the actual name can't be determined or it could be because the relevant language module can't be loaded so use `pgf`'s command instead, which also has limited language support.

```
2169 \dtmnamewarning{\DTMweekdayname}%
```

```
2170 \pgfcalendarweekdayname{#1}%
```

```
2171 }%
```

```
2172 }%
```

```
2173 }
```

\DTMWeekdayname

```
2174 \newrobustcmd{\DTMWeekdayname}[1]{%
```

First check if `\DTM<language>Weekdayname` exists.

```
2175 \ifcsdef{DTM\language Weekdayname}%
```

```
2176 {%
```

It exists, so use it.

```
2177 \csuse{DTM\language Weekdayname}{#1}%  
2178 }%  
2179 {%
```

Try obtaining the language name from the dialect using tracklang's interface.

```
2180 \dtm@ifdianameexists{Weekdayname}%  
2181 {%
```

It exists, so use it.

```
2182 \csuse{DTM\TrackedLanguageFromDialect{\language}Weekdayname}{#1}%  
2183 }%  
2184 {%
```

Can't determine the language name macro. This could be because there's no upper case macro as the names always start with a capital (like English).

```
2185 \ifcsdef{DTM\language weekdayname}%  
2186 {%  
2187 \csuse{DTM\language weekdayname}{#1}%  
2188 }%  
2189 {%  
2190 \dtm@ifdianameexists{weekdayname}%  
2191 {%  
2192 \csuse{DTM\TrackedLanguageFromDialect{\language}weekdayname}{#1}%  
2193 }%  
2194 {%
```

Can't find no-case change version either, so use pgfcalendar command instead (which will need a case-change applied).

```
2195 \dtmnamewarning{\DTMWeekdayname}%
```

If mfirstuc has been loaded, use it.

```
2196 \ifdef\emakefirstuc  
2197 {%  
2198 \emakefirstuc{\pgfcalendarweekdayname{#1}}%  
2199 }%  
2200 {%
```

Hasn't been loaded, so just expand and apply \MakeUppercase:

```
2201 \protected@edef\dtm@tmp@weekdayname{\pgfcalendarweekdayname{#1}}%  
2202 \expandafter\MakeUppercase\dtm@tmp@weekdayname  
2203 }%  
2204 }%  
2205 }%  
2206 }%  
2207 }%  
2208 }
```

hortweekdayname

```
2209 \newrobustcmd{\DTMshortweekdayname}[1]{%
```

First check if `\DTM<language>shortweekdayname` exists.

```
2210 \ifcsdef{DTM\language shortweekdayname}%  
2211 {%
```

It exists, so use it.

```
2212 \csuse{DTM\language shortweekdayname}{#1}%  
2213 }%  
2214 {%
```

Try obtaining the language name from the dialect using `tracklang`'s interface.

```
2215 \dtm@ifdianameexists{shortweekdayname}%  
2216 {%
```

It exists, so use it.

```
2217 \csuse{DTM\TrackedLanguageFromDialect{\language}shortweekdayname}{#1}%  
2218 }%  
2219 {%
```

Can't determine the language name macro. This may be because the actual name can't be determined or it could be because the relevant language module can't be loaded so use `pgf`'s command instead, which also has limited language support.

```
2220 \dtmnamewarning{\DTMshortweekdayname}%  
2221 \pgfcalendarweekdaysshortname{#1}%  
2222 }%  
2223 }%  
2224 }
```

hortWeekdayname

```
2225 \newrobustcmd{\DTMshortWeekdayname}[1]{%
```

First check if `\DTM<language>shortWeekdayname` exists.

```
2226 \ifcsdef{DTM\language shortWeekdayname}%  
2227 {%
```

It exists, so use it.

```
2228 \csuse{DTM\language shortWeekdayname}{#1}%  
2229 }%  
2230 {%
```

Try obtaining the language name from the dialect using `tracklang`'s interface.

```
2231 \dtm@ifdianameexists{shortWeekdayname}%  
2232 {%
```

It exists, so use it.

```
2233 \csuse{DTM\TrackedLanguageFromDialect{\language}shortWeekdayname}{#1}%  
2234 }%  
2235 {%
```

Can't determine the language name macro. This could be because there's no upper case macro as the names always start with a capital (like English).

```
2236 \ifcsdef{DTM\language shortweekdayname}%  
2237 {%
```

```

2238     \csuse{DTM\languagename shortweekdayname}{#1}%
2239 }%
2240 {%
2241     \dtm@ifdianameexists{shortweekdayname}%
2242     {%
2243         \csuse{DTM\TrackedLanguageFromDialect{\languagename}shortweekdayname}%
2244         {#1}%
2245     }%
2246     {%

```

Can't find no-case change version either, so use pgfcalendar command instead (which will need a case-change applied).

```

2247     \dtmnamewarning{\DTMshortWeekdayname}%

```

If mfirstuc has been loaded, use it.

```

2248     \ifdef\emakefirstuc
2249     {%
2250         \emakefirstuc{\pgfcalendarweekdaysshortname{#1}}%
2251     }%
2252     {%

```

Hasn't been loaded, so just expand and apply \MakeUppercase:

```

2253         \protected@edef\dtm@tmp@weekdayname{%
2254             \pgfcalendarweekdaysshortname{#1}}%
2255         \expandafter\MakeUppercase\dtm@tmp@weekdayname
2256     }%
2257 }%
2258 }%
2259 }%
2260 }%
2261 }

```

\DTMordinal

```

2262 \newrobustcmd{\DTMordinal}[1]{%

```

First check if \DTM<*language*>ordinal exists.

```

2263 \ifcsdef{DTM\languagename ordinal}%
2264 {%

```

It exists, so use it.

```

2265     \csuse{DTM\languagename ordinal}{#1}%
2266 }%
2267 {%

```

Try obtaining the language name from the dialect using tracklang's interface.

```

2268     \dtm@ifdianameexists{ordinal}%
2269     {%

```

It exists, so use it.

```

2270     \csuse{DTM\TrackedLanguageFromDialect{\languagename}ordinal}{#1}%
2271 }%
2272 {%

```

Can't determine the language name macro. This may be because the actual name can't be determined or it could be because the relevant language module can't be loaded so just display the number.

```
2273     \number#1
2274   }%
2275 }%
2276 }
```

`\dtmnamewarning` Issue warning unless warnings have been suppressed.

```
2277 \newcommand*{\dtmnamewarning}[1]{%
2278   \if@dtm@warn
2279     \PackageWarning{datetime2-calc}%
2280     {Can't find underlying language macro for \MessageBreak
2281       \string#1\space(language: \languagename); \MessageBreak
2282       using pgfcalendar macro instead}%
2283   \fi
2284 }
```

Change History

1.0 (2015-03-24)	
General: Initial release	109
1.03 (2016-01-20)	
\DTMusemodule: new	157
1.1 (2015-09-15)	
\DTMsaveaszulutime: fixed bug in incorrect conversion	162
fixed bug in misnamed \@dtm@hour	164
1.2 (2015-11-10)	
\DTMprovidedatestyle: new	126
\DTMprovidestyle: new	131
\DTMprovidetimestyle: new	127
\DTMprovidezonestyle: new	128
\DTMrenewdatestyle: new	126
\DTMrenewstyle: new	131
\DTMrenewtimestyle: new	127
\DTMrenewzonestyle: new	127
1.3 (2016-01-22)	
General: added ddmmyy style	143
added mmdyy style	148
added predefined dialect test	114
\dtm@ifdianameexists: new	165
\DTMdialecttomodulemap: new	157
\DTMlangsetup: added starred version	158
\DTMMonthname: new	165
\DTMmonthname: new	165
\dtmnamewarning: new	173
\DTMordinal: new	172
\DTMshortMonthname: new	167
\DTMshortmonthname: new	166
\DTMshortWeekdayname: new	171
\DTMshortweekdayname: new	170
\DTMWeekdayname: new	169
\DTMweekdayname: new	169
\RequireDateTimeModule: added	
dialect to module map	157
added module to dialect map	156
1.5 (2016-06-04)	
\dtm@pdfcreationdate: new	109
1.5.1 (2016-06-05)	
\dtm@pdfcreationdate: added check for \pdffeedback	109
1.5.2 (2016-07-12)	
\dtm@pdfcreationdate: added check for \TeXOSQueryNow	110
\DTMifhasdatestyle: new	130
\DTMifhasstyle: new	130
\DTMifhastimestyle: new	130
\DTMifhaszonestyle: new	130
\DTMsavefrompdfdata: new	116
\DTMsetregional: new	134
\DTMtryregional: new	132
1.5.3 (2018-07-20)	
General: added space after test	136–143, 145–149
1.5.3 (??)	
\dtmFridayIndex: new	169
\dtmMondayIndex: new	168
\dtmSaturdayIndex: new	169
\DTMsavenow: new	151
\dtmSundayIndex: new	169
\dtmThursdayIndex: new	169
\dtmTuesdayIndex: new	168
\dtmWednesdayIndex: new	169
2016-02-11 (1.4)	
\DTMcomputedayofweekindex: new	168
\DTMToday: new	121
\DTMtoday: new	121
\today: no longer using \renewcommand	121

Index

Symbols	
<code>\@dtm@currentdow</code>	121, 160
<code>\@dtm@initialstyle</code>	114
<code>\@dtm@julianday</code>	160
<code>\@dtm@loadedregions</code>	156
<code>\@dtm@parsedate</code>	115, 160
<code>\@dtm@parsetime</code>	115
<code>\@dtm@parsetimestamp</code>	116
<code>\@dtm@parsetimezn</code>	115
<code>\@dtm@parsezone</code>	115
<code>\@dtm@requiremodule</code>	156
<code>\@dtm@setusecalc</code>	113
<code>\@dtm@tryregional</code>	133
<code>\@dtm@unknown@style</code>	135
<code>\@dtm@unknownstyle</code>	135
<code>\@dtm@usecalc</code>	113
<code>\@dtm@warning</code>	113
B	
babel package	5, 7–9, 12, 15, 17, 18, 34–40, 45, 46, 59, 60, 68, 84, 86, 87, 159
C	
calc (option)	113
D	
date style	22, 23, 28
date-time style	15
datesep (option)	110
datetime package commands	
<code>\currenttime</code>	69
<code>\dateseparator</code>	70, 90
<code>\dayofweekname</code>	77
<code>\dayofweeknameid</code>	75, 77
<code>\dayofweeknameid<lang></code>	74
<code>\dayofweeknameidenglish</code>	74, 75
<code>\dayofweeknameidfrench</code>	74
<code>\ddmmyydate</code>	91
<code>\ddmmyyyydate</code>	91
<code>\displaydate</code>	81
<code>\dmyydate</code>	91
<code>\dmyyyydate</code>	91
<code>\formatdate</code>	69
<code>\formattime</code>	69
<code>\getdateday</code>	81
<code>\getdatemonth</code>	81
<code>\getdateyear</code>	82
<code>\ifshowdow</code>	77
<code>\longdate</code>	90
<code>\mdyydate</code>	93
<code>\mdyyyydate</code>	92
<code>\mddydate</code>	93
<code>\mddyyyydate</code>	92
<code>\monthname</code>	71, 72
<code>\monthnameenglish</code>	72
<code>\monthnamefrench</code>	72
<code>\newdate</code>	80
<code>\newdateformat</code>	94
<code>\newtimeformat</code>	102
<code>\ordinaldate</code>	77
<code>\pdfdate</code>	70
<code>\setdefaultdate</code>	86
<code>\settimeformat</code>	93
<code>\shortdate</code>	90
<code>\shortdayofweekname</code>	77
<code>\shortmonthname</code>	73
<code>\textdate</code>	91
<code>\THEDAY</code>	94
<code>\THEHOUR</code>	102
<code>\THEHOURXII</code>	102, 104
<code>\THEMINUTE</code>	102
<code>\THEMONTH</code>	94
<code>\THESECOND</code>	102
<code>\THETOHOUR</code>	102, 104
<code>\THETOMINUTE</code>	102, 104
<code>\THEYEAR</code>	94
<code>\timeseparator</code>	70
<code>\twodigit</code>	80
<code>\usdate</code>	92
<code>\yyyymmdddate</code>	89
datetime package	164

datetime2 package	6, 121, 164	\DTMenglishordinal	77
datetime2-calc package	113–115, 159	\DTMenglishshortmonthname	73
datetime2-en-fulltext package		\DTMenglishweekdayname	77
.....	63, 68, 85, 92, 94, 107	\DTMfetchday	20, 82, 152
datetime2-english package	85	\DTMfetchdow	20, 77, 82, 152
datetime2-french package	85, 86	\DTMfetchhour	20, 152
datetimesep (option)	111	\DTMfetchminute	21, 153
dayyearsep (option)	111	\DTMfetchmonth	20, 82, 152
ddmmyy style	26, 91, 143	\DTMfetchsecond	21, 153
ddmmyyyy style	26, 91, 139	\DTMfetchTZhous	21, 153
default style	6, 8,	\DTMfetchTZminute	21, 153
24–27, 34–36, 38, 54, 90, 93, 107, 112, 135		\DTMfetchyear	20, 82, 152
\directlua	119	\DTMfrenchordinal	78
display style	10	\dtmFridayIndex	169
dmyy style	26, 91, 142	\DTMgermanordinal	78
dmyyyy style	26, 91, 141	\DTMhaszonemap	33, 129
document (environment)	12, 31, 46, 47	\DTMifbool	158
\dtm@dayyearsep	110	\DTMifcaseregional	112
\dtm@hourminsep	110	\DTMifdate	52, 161
\dtm@ifdianameexists	165	\DTMifhasdatestyle	30, 130
\dtm@minsecsep	110	\DTMifhasstyle	30, 130
\dtm@monthdaysep	110	\DTMifhastimestyle	30, 130
\dtm@pdfcreationdate	109	\DTMifhaszonestyle	30, 130
\dtm@timezonesep	110	\DTMifsaveddate	20, 155
\dtm@yearmonthsep	110	\DTMlangsetup	38, 48, 158
\DTMbretonfmtordinal	78	\DTMlangsetup options	
\DTMbretonordinal	78	abbr	90
\DTMcentury	31, 125	monthyearsep	90
\DTMclearmap	32, 129	ord	78, 90
\DTMcomputedayofweekindex	56, 168	\DTMmakeglobal	19, 152
\DTMcurrenttime	14, 69, 122	\dtmMondayIndex	168
\DTMcurrentzone	14, 123	\DTMmonthname	55, 165
\DTMdate	13, 122	\DTMmonthname	41, 54, 72, 97, 165
\DTMdate	12, 69, 122	\dtmnamewarning	76, 173
\DTMdefboolkey	158	\DTMNatoZoneMaps	32, 129
\DTMdefchoicexkey	158	\DTMnewdatestyle	28, 94, 126
\DTMdefkey	158	\DTMnewstyle	28, 130
\DTMdefzonemap	32, 128	\DTMnewtimestyle	28, 102, 126
\DTMdialecttomodulemap	40, 157	\DTMnewzonestyle	28, 127
\DTMDisplay	15, 124	\DTMnorskordinal	79
\DTMdisplay	15, 69, 124	\DTMNow	16, 123
\DTMdisplaydate	11, 77, 95, 122	\DTMnow	15, 123
\DTMdisplaydate	10, 36, 77, 95, 121	\DTMordinal	58, 79, 172
\DTMdisplaytime	14, 69, 102, 122	\DTMprovidedatestyle	29, 126
\DTMdisplayzone	14, 123	\DTMprovidestyle	30, 131
\DTMdivhundred	31, 125	\DTMprovidetimestyle	29, 127
\DTMenglishampfmt	107	\DTMprovidezonestyle	30, 128
\DTMenglishfmtordsuffix	78	\DTMrenewdatestyle	29, 126
\DTMenglishmonthname	71	\DTMrenewstyle	29, 131

<code>\DTMrenewtimestyle</code>	29, 127	<code>\DTMuseumzonemap</code>	32, 128
<code>\DTMrenewzonestyle</code>	29, 127	<code>\DTMuseumzonemapordefault</code>	33, 128
<code>\DTMresetzones</code>	32, 129	<code>\dtmWednesdayIndex</code>	169
<code>\dtmSaturdayIndex</code>	169	<code>\DTMWeekdayname</code>	57, 169
<code>\DTMsaveaszulutime</code>	53, 162	<code>\DTMweekdayname</code>	56, 76, 77, 169
<code>\DTMsavedate</code>	17, 81, 149	<code>\DTMwelshfmtordinal</code>	78
<code>\DTMsaveddatediff</code>	52, 161	<code>\DTMwelshordinal</code>	78
<code>\DTMsaveddateoffsettojulianday</code>	52, 161		
<code>\DTMsaveddatetojuliandate</code>	51	E	
<code>\DTMsaveddatetojulianday</code>	160	<code>en-FullText style</code>	94
<code>\DTMsavefilemoddate</code>	19, 116	<code>en-GB style</code>	7, 36–38, 50, 65, 86, 90, 94, 101
<code>\DTMsavefrompdfdata</code>	18, 116	<code>en-GB-numeric style</code>	7, 36–38, 49, 50
<code>\DTMsavejulianday</code>	51, 160	<code>en-GG style</code>	90
<code>\DTMsavenoparsedate</code>	17, 150	<code>en-US style</code>	78
<code>\DTMsavenow</code>	18, 83, 151, 152	<code>english style</code>	38
<code>\DTMsavetime</code>	17, 150	<code>englishshampm style</code>	104, 106
<code>\DTMsavetimestamp</code>	18, 151	environments:	
<code>\DTMsavetimezn</code>	18, 150	<code>document</code>	12, 31, 46, 47
<code>\DTMsep</code>	31, 126	<code>etoolbox package</code>	109
<code>\DTMsetbool</code>	158		
<code>\DTMsetcurrentzone</code>	121	F	
<code>\DTMsetdatestyle</code>	22, 89, 132	<code>fmtcount package</code>	58, 77, 79, 84, 85, 97
<code>\DTMsetregional</code>	46, 134	<code>full style</code>	15, 22, 23, 28
<code>\DTMsetstyle</code>	22, 89, 93, 135		
<code>\DTMsettimestyle</code>	22, 93, 132	H	
<code>\DTMsetup</code>	12, 25, 36, 48, 115	<code>hhmm style</code>	28, 149
<code>\DTMsetzonestyle</code>	22, 132	<code>hmmss style</code>	27, 149
<code>\DTMshortMonthname</code>	56, 167	<code>hourminsep (option)</code>	111
<code>\DTMshortmonthname</code>	56, 73, 166	<code>hyperref package</code>	31, 99, 125
<code>\DTMshortWeekdayname</code>	58, 171		
<code>\DTMshortweekdayname</code>	57, 170	I	
<code>\DTMshowmap</code>	129	<code>\ifDTMshowdow</code>	77
<code>\dtmSundayIndex</code>	169	<code>iso style</code>	25, 44, 45, 61, 70, 112, 136
<code>\DTMtexorpdfstring</code>	31, 107, 125		
<code>\dtmThursdayIndex</code>	169	L	
<code>\DTMtime</code>	15, 70, 122	<code>locale package</code>	109, 168
<code>\DTMToday</code>	12, 121	<code>luatex85 package</code>	109, 120
<code>\DTMtoday</code>	12, 121		
<code>\DTMtozulu</code>	53, 162	M	
<code>\DTMtryregional</code>	23, 24, 132	<code>map style</code>	27, 28, 149
<code>\dtmTuesdayIndex</code>	168	<code>mdyy style</code>	26, 93, 146
<code>\DTMtwodigits</code>	30, 80, 96, 124	<code>mdyyyy style</code>	26, 92, 145
<code>\DTMUse</code>	20, 155	<code>mfirstuc package</code>	11, 166, 168, 170, 172
<code>\DTMUse</code>	20, 154	<code>minsecsep (option)</code>	111
<code>\DTMUsedate</code>	19, 153	<code>mmddy style</code>	26, 93, 148
<code>\DTMusedate</code>	19, 81, 153	<code>mmddyyyy style</code>	26, 92, 144
<code>\DTMUsemodule</code>	9, 39, 157	<code>\month</code>	72
<code>\DTMusetime</code>	19, 154	<code>monthdaysep (option)</code>	111
<code>\DTMUsezone</code>	20, 154		
		N	
		<code>\number</code>	95

O	
options (definitions):	
calc	113
datesep	110
datetimesep	111
dayyearsep	111
hourminsep	111
minsecsep	111
monthdaysep	111
showdate	111
showdown	113
showisoZ	112
showseconds	111
showzone	112
showzoneminutes	112
style	114
timesep	111
timezonesep	111
useregional	112
warn	114
yearmonthsep	111
UKenglish	38
usenumerical	
false	46, 47
text	46, 47
useregional	
. 7, 9, 23, 24, 34, 36–38, 45–47, 59, 87, 132	
false	24, 46, 47, 114
true	38
warn	47, 55
yearmonthsep	25, 32, 33, 44
partial style	23, 28
pdf style	26, 27, 44, 45, 71, 112, 137
\pdfcreationdate	15, 70, 109
\pdfinfo	26, 70
pgf package	165, 167, 169, 171
pgfcalendar package	
. 12, 13, 42, 43, 47, 51, 54–57, 72,	
73, 75, 77, 84, 113, 160, 166, 168, 170, 172	
\pgfcalendardatetojulian	77
\pgfcalendarjuliantoweekday	77
\pgfcalendarmonthname	72
\pgfcalendarmonthshortname	73
\pgfcalendarweekdayname	75
polyglossia package . 9, 12, 34–39, 45, 46, 84, 159	
\ProvidesDateTimeModule	157
P	
package options:	
british	38, 46, 47
calc	12, 17, 47, 48, 51, 72
datesep	25, 26, 44, 48–50, 70, 90
datetimesep	25, 45
dayyearsep	26, 44
en-GB	38, 90, 101
english	38
hourminsep	25, 28, 33, 44, 107
mapzone	32
minsecsep	25, 44
monthdaysep	25, 26, 44
scottish	38
showdate	25, 45
showdown 8, 10–12, 17, 43, 47, 66, 72, 90	
true	12, 47, 51
showisoZ	25, 27, 28, 33, 45
false	6
showseconds	15, 25, 45, 93
false	6
showzone	15, 25, 45
false	6
showzoneminutes	25, 28, 33, 45
style	34, 46–48, 159
timesep	44, 70
timezonesep	25, 45
R	
\RequireDateTimeModule	39, 156
S	
\s@dtm@tryregional	133
scottish style	38
scottish-numeric style	38
scrlltr2 class	121
showdate (option)	111
showdown (option)	113
showisoZ (option)	112
showseconds (option)	111
showzone (option)	112
showzoneminutes (option)	112
style (option)	114
styles:	
ddmmyy	26, 91, 143
ddmmyyyy	26, 91, 139
default	6, 8,
24–27, 34–36, 38, 54, 90, 93, 107, 112, 135	
dmyy	26, 91, 142
dmyyyy	26, 91, 141
en-FullText	94
en-GB	7, 36–38, 50, 65, 86, 90, 94, 101

en-GB-numeric	7, 36–38, 49, 50	timesep (option)	111
en-GG	90	timezonesep (option)	111
en-US	78	\Today	11, 121
english	38	\today	11, 68, 121
englishampm	104, 106	tracklang package .	34, 35, 37–39, 47, 54–57, 76, 109, 114, 156, 159, 165–167, 169–172
hhmm	28, 149	translator package	42, 72, 76
hmmss	27, 149		
iso	25, 44, 45, 61, 70, 112, 136	U	
map	27, 28, 149	ukdate package	5
mdyy	26, 93, 146	useregional (option)	112
mdyyyy	26, 92, 145		
mmddyy	26, 93, 148	W	
mmddyyyy	26, 92, 144	warn (option)	114
pdf	26, 27, 44, 45, 71, 112, 137		
scottish	38	X	
scottish-numeric	38	xkeyval package	48, 109
yyyymd	26, 138		
		Y	
T		yearmonthsep (option)	111
texosquery package	6, 15, 19, 34, 45, 109, 110, 119	yyyymd style	26, 138
\TeXOSQueryNow	15, 110		
\time	15	Z	
time style	14, 22, 23, 28	zone style	14, 22, 23, 28