datetime2 v1.5.7: date and time formats

Nicola L. C. Talbot
http://www.dickimaw-books.com/

2021-03-21

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

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 PDFTEX 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 LATEX 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.

---

[1] Of 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 XƎLaTeX in which case the seconds and time zone are omitted. (XƎLaTeX 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[useregional]{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[useregional,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).

Some regional styles may end the date with a period (full stop). This can cause a problem if the date occurs at the end of a sentence. If the style has used `\DTMfinaldot` then you can use `\DTMdate*` to suppress the terminating period if required. (See below.)

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⟩}
```

As from v1.5.5 there are also starred versions of these commands:

`\DTMdate*`

```
\DTMdate*{⟨date⟩}
```

which essentially behaves like:

```
{\let\DTMfinaldot\empty\DTMdate{⟨date⟩}}
```

and

`\DTMDate*`

```
\DTMDate*{⟨date⟩}
```

which essentially behaves like:

```
{\let\DTMfinaldot\empty\DTMDate{⟨date⟩}}
```

These commands are provided for date styles that use `\DTMfinaldot` to produce a period (full stop) at the end of the date. These two star-commands may be used at the end of a sentence to prevent a double period. Note that there is no equivalent starred form of `\DTMdisplaydate` or `\DTMDisplaydate`. (It would break their ability to fully expand, which was their principle design feature.)

In these cases (starred and unstarred) 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 negative[1] 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).

---

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

- ⟨*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}
```

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 PDFTEX primitive `\pdfcreationdate` is defined, the current time information is obtained from that, which includes the seconds and time zone. LuaTEX also defines this command (now replaced with `\pdffeedback␣creationdate`) but XƎTEX doesn't, and in that case the only way to determine the current time is from TEX's `\time` primitive which only contains the number of minutes since midnight, which means that the seconds and time zone are unavailable. Therefore if XƎTEX 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 showdow 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 TeX 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 PDFLaTeX, this command will use the PDFTeX primitive `\pdffilemoddate`. If you use LuaTeX 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 XeLaTeX 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:

\DTMUsedate

```
\DTMUsedate{⟨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*⟩-⟨*county 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*[⟨lang name⟩]{⟨lang code cs⟩}{⟨country code cs⟩}`

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

⟨*YYYY*⟩⟨*YMsep*⟩⟨*MM*⟩⟨*MDsep*⟩⟨*DD*⟩

where the month ⟨*MM*⟩ and day of the month ⟨*DD*⟩ numbers are formatted as two digits. The separators ⟨*YMsep*⟩ and ⟨*MDsep*⟩ 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:

⟨*hh*⟩⟨*HMsep*⟩⟨*mm*⟩⟨*MSsep*⟩⟨*ss*⟩

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

The zone is displayed in form

⟨*TZh*⟩⟨*HMsep*⟩⟨*TZm*⟩

or just `Z` if the option showisoZ is set to `true` and both ⟨*TZh*⟩ and ⟨*TZm*⟩ are zero. The separator ⟨*HMsep*⟩ is the same as used for the time format. The final ⟨*HMsep*⟩⟨*TZm*⟩ is omitted if the option showzoneminutes is set to `false`. The hour offset ⟨*TZh*⟩ is formatted as two digits proceeded 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

⟨*date*⟩⟨*DTsep*⟩⟨*time*⟩⟨*TZsep*⟩⟨*zone*⟩

The ⟨*date*⟩⟨*DTsep*⟩ part is omitted if the option showdate is set to `false`, and the ⟨*TZsep*⟩⟨*zone*⟩ part is omitted if the option showzone is set to `false`. The separator between the date and time ⟨*DTsep*⟩ defaults to `\space` but may be changed using the datetimesep option. The separator between the time and zone ⟨*TZsep*⟩ 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.

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

⟨*DD*⟩⟨*MDsep*⟩⟨*MM*⟩⟨*YMsep*⟩⟨*YYYY*⟩

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.

`dmyyyy` This is like the `ddmmyyyy` 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

⟨*DD*⟩⟨*MDsep*⟩⟨*MM*⟩⟨*YMsep*⟩⟨*YY*⟩

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.

`mmddyyyy` This is like the `ddmmyyyy` 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.

`mmddyy` 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.

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

`mdyy` This is like the `mdyyyy` 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:⟨*YYYY*⟩⟨*MM*⟩⟨*DD*⟩

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

⟨*TZh*⟩⟨*HMsep*⟩⟨*TZm*⟩

where ⟨*HMsep*⟩ is given by the hourminsep option. This style honours the showzoneminutes 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{⟨name⟩}{⟨definition⟩}
```

This defines a partial style that should only modify `\DTMdisplaydate` and `\DTMDisplaydate`. The redefinition of these commands should be placed in ⟨*definition*⟩. If the date should end with a period (full stop) use:

\DTMfinaldot

```
\DTMfinaldot
```

at the end of the definitions of `\DTMdisplaydate` and `\DTMDisplaydate`.

This command is simply defined as "`.`" but it's locally redefined to do nothing by the starred versions of `\DTMdate` and `\DTMDate` allowing the dot to be discarded.

A new time style can be defined using:

\DTMnewtimestyle

```
\DTMnewtimestyle{⟨name⟩}{⟨definition⟩}
```

This defines a partial style that should only modify `\DTMdisplaytime`. The redefinition should be placed in ⟨*definition*⟩.

A new zone style can be defined using:

\DTMnewzonestyle

```
\DTMnewzonestyle{⟨name⟩}{⟨definition⟩}
```

This defines a partial style that should only modify `\DTMdisplayzone`. The redefinition should be placed in ⟨*definition*⟩.

A new full style can be defined using:

\DTMnewstyle

```
\DTMnewstyle{⟨name⟩}{⟨date style definition⟩}{⟨time style definition⟩}
{⟨zone style definition⟩}{⟨full style definition⟩}
```

This does

`\DTMnewdatestyle{`⟨*name*⟩`}{`⟨*date style definition*⟩`}`

`\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 it 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 LaTeX's `\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 `\datetime`'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 XƎLaTeX 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.

The document languages are detected using tracklang's interface. This has limitations. In particular, it can't detect the region with polyglossia nor can it detect languages specified after tracklang has been loaded, and it can't pick up babel's new `\babelprovide` command. See the tracklang documentation for further details. Please check that you have an up-to-date version of tracklang.

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 `\languagename`) 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 provided 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, LuaTeX and XeTeX 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 LuaLaTeX and XeLaTeX 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 LuaLaTeX and XeLaTeX 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 XƎTEX 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 XƎTEX 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 `\languagename`). 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 usenumerical=text but the style option then implements usenumerical=false.)

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

In the above useregional is `text`. (The style option implements usenumerical=false but the british option then implements usenumerical=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 2021.03.21 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 2021.03.21 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 2021-03-21 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[useregional=numeric]{datetime2}
\DTMlangsetup{datesep=.}
\begin{document}
\today
\end{document}
```

The above displays the date in the form 21.3.2021 since the `en-GB-numeric` style is in use and `datesep` is used in `\DTMlangsetup`.

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

The above displays the date in the form 21st March 2021 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[useregional=numeric]{datetime2}
\DTMsetup{datesep=.}
\begin{document}
\today
\end{document}
```

The above displays the date in the form 21/3/2021 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 ⟨*name*⟩) to a Julian day. The result is stored in ⟨*register*⟩ 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 ⟨*offset*⟩. The result is stored in ⟨*register*⟩. This is equivalent to

```
\pgfcalendardatetojulian{⟨y⟩-⟨m⟩-⟨d⟩+⟨offset⟩}{⟨register⟩}
```

where ⟨*y*⟩, ⟨*m*⟩ and ⟨*d*⟩ are the year, month and day fetched from the saved date. A negative ⟨*offset*⟩ 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 ⟨*name*⟩). 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 ⟨*test*⟩ 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`

```
\DTMsavedatediff{⟨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 ⟨*name1*⟩ 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 Zulu[1] 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 `\DTMuse` 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: \DTMuse{mydate}.

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

---

[1] That'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 `\languagename`. If so, that macro is used. For example:

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

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

If the test with `\languagename` didn't work, `\DTMmonthname` then tries with ⟨*lang*⟩ set to

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

(`\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 `\languagename` 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 `\languagename` and then where ⟨*lang*⟩ is given by

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

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 ⟨*lang*⟩ for either `\DTM⟨lang⟩Monthname` or `\DTM⟨lang⟩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{⟨n⟩}
```

This behaves in a similar way to `\DTMmonthname` but tries to determine if `\DTM⟨lang⟩shortmonthname` exists (where ⟨*lang*⟩ is either `\languagename` or obtained from `\languagename` 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{⟨n⟩}
```

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

`\DTMcomputedayofweekindex`

```
\DTMcomputedayofweekindex{⟨date⟩}{⟨cs⟩}
```

where ⟨*date*⟩ is in the form ⟨*YYYY*⟩-⟨*MM*⟩-⟨*DD*⟩ and ⟨*cs*⟩ 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{⟨n⟩}
```

This checks if `\DTM⟨lang⟩weekdayname` exists where ⟨*lang*⟩ is given by `\languagename`. 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 fmtcount'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 it 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[useregional]{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 useregional 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 20$^{\text{th}}$ 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 20$^{\text{th}}$ 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 ⟨*YYYY*⟩-⟨*MM*⟩-⟨*DD*⟩ 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 ⟨*YYYY*⟩/⟨*MM*⟩/⟨*DD*⟩ dates. For example:

```
\documentclass{article}
\usepackage[yyyymmdd]{datetime}
\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 ⟨*DD*⟩/⟨*MM*⟩/⟨*YYYY*⟩ dates. For example:

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

dmyyyy This option was designed for ⟨*D*⟩/⟨*M*⟩/⟨*YYYY*⟩ 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 ⟨*DD*⟩/⟨*MM*⟩/⟨*YY*⟩ 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 ⟨*D*⟩/⟨*M*⟩/⟨*YY*⟩ 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}
```

mmddyyyy This option was designed for ⟨*MM*⟩/⟨*DD*⟩/⟨*YYYY*⟩ dates. For example:

```
\documentclass{article}
\usepackage[mmddyyyy]{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 ⟨*M*⟩/⟨*D*⟩/⟨*YYYY*⟩ 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 ⟨*MM*⟩/⟨*DD*⟩/⟨*YY*⟩ 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 ⟨*M*⟩/⟨*D*⟩/⟨*YY*⟩ 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 20$^{\text{th}}$ January, 2016

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

```
\documentclass{article}
\usepackage[en-GB,showdow]{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 20$^{\text{th}}$ 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: 20$^{\text{th}}$ 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 ⟨*HH*⟩:⟨*MM*⟩:⟨*SS*⟩. 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 ⟨*HH*⟩:⟨*MM*⟩. 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]{datetime}

\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 is 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 ⟨*hh*⟩ is the hour, ⟨*mm*⟩ is the minutes and ⟨*ss*⟩ 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 PDFTEX 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 PDFTEX):

```
\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 showdow 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 datetime, 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 ⟨*dow*⟩ 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{⟨dow⟩}`

This attempts to use `\DTM⟨lang⟩weekdayname` if it exists, where ⟨*lang*⟩ is either `\languagename` 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 ⟨*n*⟩, 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 ⟨*n*⟩ 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 ⟨*n*⟩.

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$^{\text{er}}$ in the first case (through `\ordinaldatefrench`) and 1$^{\text{st}}$ 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$^{\text{er}}$ 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`):

`dt-american.def`
`dt-australian.def`
`dt-austrian.def`
`dt-bahasa.def`
`dt-basque.def`
`dt-breton.def`
`dt-british.def`
`dt-bulgarian.def`
`dt-canadian.def`
`dt-catalan.def`
`dt-croatian.def`
`dt-czech.def`
`dt-danish.def`
`dt-dutch.def`
`dt-esperanto.def`
`dt-estonian.def`
`dt-finnish.def`
`dt-french.def`
`dt-galician.def`
`dt-german.def`
`dt-greek.def`
`dt-hebrew.def`
`dt-icelandic.def`
`dt-irish.def`
`dt-italian.def`
`dt-latin.def`
`dt-lsorbian.def`
`dt-magyar.def`
`dt-naustrian.def`
`dt-newzealand.def`
`dt-ngerman.def`
`dt-norsk.def`
`dt-polish.def`
`dt-portuges.def`
`dt-romanian.def`
`dt-russian.def`
`dt-samin.def`
`dt-scottish.def`
`dt-serbian.def`
`dt-slovak.def`
`dt-slovene.def`
`dt-spanish.def`
`dt-swedish.def`
`dt-turkish.def`
`dt-UKenglish.def`
`dt-ukraineb.def`
`dt-USenglish.def`
`dt-usorbian.def`
`dt-welsh.def`

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 fmtcount, 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:

`datetime2-bahasai`
`datetime2-basque`
`datetime2-breton`
`datetime2-bulgarian`
`datetime2-catalan`
`datetime2-croatian`
`datetime2-czech`
`datetime2-danish`
`datetime2-dutch`
`datetime2-english`
`datetime2-esperanto`
`datetime2-estonian`
`datetime2-finnish`
`datetime2-french`
`datetime2-galician`
`datetime2-german`
`datetime2-greek`
`datetime2-hebrew`
`datetime2-icelandic`
`datetime2-irish`
`datetime2-italian`
`datetime2-latin`
`datetime2-lsorbian`
`datetime2-magyar`
`datetime2-norsk`
`datetime2-polish`
`datetime2-portuges`
`datetime2-romanian`
`datetime2-russian`
`datetime2-samin`
`datetime2-scottish`
`datetime2-serbian`
`datetime2-slovak`
`datetime2-slovene`
`datetime2-spanish`
`datetime2-swedish`
`datetime2-turkish`
`datetime2-ukrainian`
`datetime2-usorbian`
`datetime2-welsh`

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[showdow,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[showdow,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}

\yyyymmdddate

\begin{document}
\selectlanguage{french}
\today
```

```
\selectlanguage{british}
\today

\end{document}
```

This produces:

> 21 janvier 2016
>
> Thursday 21$^{\text{st}}$ 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 previous 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 showdow and en-GB package options and the ord and monthyearsep options for the `en-GB` style. For example:

```
\usepackage[showdow,en-GB]{datetime2}
\DTMlangsetup[en-GB]{ord=raise,monthyearsep={,\space}}
```

Another possibility is:

```
\usepackage[british]{babel}
\usepackage[showdow]{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 8$^{\text{th}}$ Mar, 2000. With datetime2, this is like the above but additionally needs the abbr option for the `en-GB` style:

```
\usepackage[showdow,en-GB]{datetime2}
\DTMlangsetup[en-GB]{ord=raise,monthyearsep={,\space},abbr}
```

or

```
\usepackage[british]{babel}
\usepackage[showdow]{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 `\ddmmyydate`

This produces a date in the form 08/03/00. This can be reproduced with just datetime2 using the `ddmmyy` 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[useregional]{datetime2}
\DTMsetdatestyle{en-US}
```

datetime `\mmddyyyydate`

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 `mmddyyyy` style with the date separator set to a slash:

```
\DTMsetup{datesep=/}
\DTMsetdatestyle{mmddyyyy}
```

datetime `\mdyyyydate`

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 `mdyyyy` style with the date separator set to a slash:

```
\DTMsetup{datesep=/}
\DTMsetdatestyle{mdyyyy}
```

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 `\mmddyydate`

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 `mmddyy` style with the date separator set to a slash:

```
\DTMsetup{datesep=/}
\DTMsetdatestyle{mmddyy}
```

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. LaTeX 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 1$^{\mathrm{st}}$, 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$^{\text{er}}$, 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 1*st*, 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`, `\THETOHOUR` 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]{%
```

---

[1] Bug fix from datetime2-english v1.03 included

```
\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
          {\DTMenglishampmfmt{\DTMenglisham}}%
          {\DTMenglisham}%
        \else
          \number\numexpr##1-12\relax
          \DTMtexorpdfstring
          {\DTMenglishampmfmt{\DTMenglishpm}}%
          {\DTMenglishpm}%
        \fi
        \fi
      \fi
    \fi
  \fi
\else
  \ifnum##1<13
    \ifnum##1=0
      12%
    \else
      \number##1
    \fi
    \DTMenglishtimesep\DTMtwodigits{##2}%
    \ifnum##1=12
      \DTMtexorpdfstring
      {\DTMenglishampmfmt{\DTMenglishpm}}%
      {\DTMenglishpm}%
    \else
      \DTMtexorpdfstring
      {\DTMenglishampmfmt{\DTMenglisham}}%
      {\DTMenglisham}%
    \fi
  \else
```

```
          \number\numexpr##1-12\relax
          \DTMenglishtimesep\DTMtwodigits{##2}%
          \ifnum##1=24
            \DTMtexorpdfstring
            {\DTMenglishampmfmt{\DTMenglisham}}%
            {\DTMenglisham}%
          \else
            \DTMtexorpdfstring
            {\DTMenglishampmfmt{\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 `ampmtime` 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 `\DTMenglishampmfmt{⟨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*{\DTMenglishampmfmt}[1]{\textsc{#1}}
```

11 The Code

11.1 datetime2.sty code

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{datetime2}[2021/03/21 v1.5.7 (NLCT) date and time formats]
```

Use tracklang to find out what languages have been loaded.

```
\RequirePackage{tracklang}
```

Also require etoolbox.

```
\RequirePackage{etoolbox}
```

Need xkeyval for ⟨*key*⟩=⟨*value*⟩ interface.

```
\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.

```
\ifdef\pdfcreationdate
{%
  \edef\dtm@pdfcreationdate{\pdfcreationdate}%
}%
{%
```

Check if newer version of LuaTEX is being used but luatex85 hasn't been loaded.

```
  \ifdef\pdffeedback
  {%
    \edef\dtm@pdfcreationdate{\pdffeedback creationdate}%
  }%
  {%
```

Neither `\pdfcreationdate` nor `\pdffeedback` are defined. (The TEX format is most probably XƎLATEX.) If the locale package has been loaded, `\LocaleNowStamp` should be available.

```
    \ifdef\LocaleNowStamp
    {%
      \ifx\LocaleNowStamp\empty
```

OS query failed, so there's no point trying directly with texosquery.

```
      \else
        \let\dtm@pdfcreationdate\LocaleNowStamp
      \fi
    }%
```

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.

```
    {%
      \ifdef\TeXOSQueryNow
      {%
        \TeXOSQueryNow{\dtm@pdfcreationdate}%
        \ifdefempty\dtm@pdfcreationdate
        {%
%   \end{macrocode}
%Failed (maybe shell escape has been disabled).
%   \begin{macrocode}
          \undef\dtm@pdfcreationdate
        }%
        {}%
      }%
      {}%
    }%
  }%
}
```

`tm@yearmonthsep` Separator between year and month for numeric dates.

```
\newcommand*{\dtm@yearmonthsep}{-}
```

`dtm@monthdaysep` Separator between month and day for numeric dates.

```
\newcommand*{\dtm@monthdaysep}{-}
```

`\dtm@dayyearsep` Separator between day and year for numeric middle-endian dates.

```
\newcommand*{\dtm@dayyearsep}{-}
```

`\dtm@hourminsep` Separator between the hour and minute for times.

```
\newcommand*{\dtm@hourminsep}{:}
```

`\dtm@minsecsep` Separator between the minute and second for times.

```
\newcommand*{\dtm@minsecsep}{:}
```

`dtm@timezonesep` Separator between the date and time.

```
\newcommand*{\dtm@datetimesep}{\space}%
```

`dtm@timezonesep` Separator between the time and time zone.

```
\newcommand*{\dtm@timezonesep}{}
```

`datesep` Set year/month and month/day separator.

```
\define@key{datetime2.sty}{datesep}{%
  \renewcommand*{\dtm@yearmonthsep}{#1}%
  \renewcommand*{\dtm@monthdaysep}{#1}%
  \renewcommand*{\dtm@dayyearsep}{#1}%
}
```

`yearmonthsep` Set year/month separator.

```
\define@key{datetime2.sty}{yearmonthsep}{%
  \renewcommand*{\dtm@yearmonthsep}{#1}%
}
```

`monthdaysep` Set month/day separator.

```
\define@key{datetime2.sty}{monthdaysep}{%
  \renewcommand*{\dtm@monthdaysep}{#1}%
}
```

`dayyearsep` Set day/year separator for middle-endian dates.

```
\define@key{datetime2.sty}{dayyearsep}{%
  \renewcommand*{\dtm@dayyearsep}{#1}%
}
```

`timesep` Set hour/minute and minute/second separator.

```
\define@key{datetime2.sty}{timesep}{%
  \renewcommand*{\dtm@hourminsep}{#1}%
  \renewcommand*{\dtm@minsecsep}{#1}%
}
```

`hourminsep` Set hour/minute separator.

```
\define@key{datetime2.sty}{hourminsep}{%
  \renewcommand*{\dtm@hourminsep}{#1}%
}
```

`minsecsep` Set minute/second separator.

```
\define@key{datetime2.sty}{minsecsep}{%
  \renewcommand*{\dtm@minsecsep}{#1}%
}
```

`timezonesep` Set separator between the time and the time zone (used in `\DTMnow`).

```
\define@key{datetime2.sty}{timezonesep}{%
  \renewcommand*{\dtm@timezonesep}{#1}%
}
```

`datetimesep` Set separator between the date and the time (used in `\DTMnow`).

```
\define@key{datetime2.sty}{datetimesep}{%
  \renewcommand*{\dtm@datetimesep}{#1}%
}
```

`showseconds` Boolean key to determine whether or not to show the seconds.

```
\define@boolkey{datetime2.sty}[DTM]{showseconds}[true]{}
```

`showdate` Boolean key to determine whether or not to show the date in `\DTMdisplay` and `\DTMDisplay`.

```
\define@boolkey{datetime2.sty}[DTM]{showdate}[true]{}
\DTMshowdatetrue
```

`showzone` Boolean key to determine whether or not to show the time zone in `\DTMdisplay` and `\DTMDisplay`.

```
\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.)

```
\define@boolkey{datetime2.sty}[DTM]{showisoZ}[true]{}
\DTMshowisoZtrue
```

Switch off seconds and time zone if `\dtm@pdfcreationdate` isn't defined, otherwise switch on.

```
\ifdef\dtm@pdfcreationdate
{%
  \DTMshowsecondstrue
  \DTMshowzonetrue
}%
{%
  \DTMshowsecondsfalse
  \DTMshowzonefalse
}%
```

`showzoneminutes` Boolean key to determine whether or not to show the time zone minutes. (If `\DTMshowzonefalse` then this option is irrelevant.)

```
\define@boolkey{datetime2.sty}[DTM]{showzoneminutes}[true]{}
\DTMshowzoneminutestrue
```

`Mifcaseregional`

```
\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.

```
\newcommand*{\DTMifcaseregional}[3]{#1}
```

`useregional` Setting to determine whether or not to use the regional settings (if any are loaded).

```
\define@choicekey{datetime2.sty}{useregional}[\@dtm@useregional@val\@dtm@useregional@nr]%
 {false,text,numeric,num}[text]%
 {%
   \ifcase\@dtm@useregional@nr\relax
     \renewcommand*{\DTMifcaseregional}[3]{##1}%
   \or
     \renewcommand*{\DTMifcaseregional}[3]{##2}%
   \or
     \renewcommand*{\DTMifcaseregional}[3]{##3}%
   \or
     \renewcommand*{\DTMifcaseregional}[3]{##3}%
   \fi
 }
```

@dtm@setusecalc

```
\newcommand*{\@dtm@setusecalc}{%
  \renewcommand*{\@dtm@usecalc}{\RequirePackage{datetime2-calc}}%
}
```

\@dtm@usecalc

```
\newcommand*{\@dtm@usecalc}{}
```

Disable attempt to load datetime2-calc in the document.

```
\AtBeginDocument{%
  \@ifpackageloaded{datetime2-calc}%
  {%
    \renewcommand*{\@dtm@setusecalc}{}%
  }%
  {%
    \renewcommand*{\@dtm@setusecalc}{%
      \PackageError{datetime2}{You must load ‘datetime2-calc’
      package to use option ‘showdow’}{Try one of the following:^^J
      pass ‘calc’ option to ‘datetime2’ package when you load it^^J
      or move ‘showdow’ option to ‘datetime2’ package option list^^J
      or move \string\DTLsetup\space to the preamble.}%
    }%
  }%
}
```

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.

```
\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.

```
\define@boolkey{datetime2.sty}[DTM]{showdow}[true]{%
  \ifDTMshowdow \@dtm@setusecalc \fi
}
\DTMshowdowfalse
```

\@dtm@warning Warning messages.

```
\newcommand*{\@dtm@warning}[1]{%
  \if@dtm@warn
```

```
    \PackageWarning{datetime2}{#1}%
  \fi
}
```

`warn` Allow user to suppress package warnings.

```
\define@boolkey{datetime2.sty}[@dtm@]{warn}[true]{}
\@dtm@warntrue
```

`tm@initialstyle`

```
\newcommand*{\@dtm@initialstyle}{}
```

`style` Set the style. This automatically sets useregional=false.

```
\define@key{datetime2.sty}{style}{%
  \renewcommand*{\@dtm@initialstyle}{#1}%
  \ifstrempty{#1}%
  {}%
  {%
    \renewcommand*{\DTMifcaseregional}[3]{##1}%
  }%
}
```

Pass any unknown options to tracklang. This will automatically switch the `useregional` setting to `text`.

```
\DeclareOptionX*{%
  \ifcsundef{@tracklang@add@\CurrentOption}%
  {%
    \ifundef\TrackIfKnownLanguage
    {%
       \PackageError{datetime2}{Unrecognised dialect ‘\CurrentOption’.
         If you are using a valid ISO language code
         please update tracklang.sty to at least v1.3.9}%
         {Any options that aren’t described in the manual are assumed
          \MessageBreak to be language or dialect names.}%
    }%
    {%
       \TrackIfKnownLanguage{\CurrentOption}%
       {\renewcommand*{\DTMifcaseregional}[3]{#2}}%
       {%
         \PackageError{datetime2}{‘\CurrentOption’ is not a recognised dialect
          \MessageBreak and doesn’t contain a known language code.
          \MessageBreak Perhaps you have misspelt it or the
          \MessageBreak named dialect may be unsupported or
          \MessageBreak perhaps you forgot the ‘<key>=’ part
          \MessageBreak for example, ‘style=\CurrentOption’}%
          {Any options that aren’t described in the manual are assumed
           \MessageBreak to be language or dialect names.}%
       }%
    }%
  }%
  {%
    \TrackPredefinedDialect{\CurrentOption}%
    \renewcommand*{\DTMifcaseregional}[3]{#2}%
  }%
}
```

Process options passed to this package:

```
\ProcessOptionsX
```

Disable `calc` option. If it's required, just load datetime2-calc with `\usepackage`.

```
\disable@keys{datetime2.sty}{calc}
```

Disable `style` option. If it's required, just use `\DTMsetup`.

```
\disable@keys{datetime2.sty}{style}
```

Provide a way to set options after package has been loaded.

`\DTMsetup`

```
\newcommand*{\DTMsetup}[1]{%
 \def\@dtm@usecalc{}%
 \setkeys{datetime2.sty}{#1}%
 \@dtm@usecalc
}
```

11.1.1 Defaults

This section sets up the defaults.

`\@dtm@parsedate` Parse date in the format ⟨*year*⟩-⟨*month*⟩-⟨*day*⟩. The arguments are expanded. (This is redefined by datetime2-calc.)

```
\def\@dtm@parsedate#1-#2-#3\@dtm@endparsedate{%
  \edef\@dtm@year{\number#1}%
  \edef\@dtm@month{\number#2}%
  \edef\@dtm@day{\number#3}%
  \def\@dtm@dow{-1}%
}
```

`\@dtm@parsetime` Define command to parse time in the format ⟨*h*⟩:⟨*m*⟩:⟨*s*⟩. The results are stored in `\@dtm@hour`, `\@dtm@minute` and `\@dtm@second`. The arguments are expanded.

```
\def\@dtm@parsetime#1:#2:#3\@dtm@endparsetime{%
  \edef\@dtm@hour{\number#1}%
  \edef\@dtm@minute{\number#2}%
  \edef\@dtm@second{\number#3}%
}
```

`dtm@parsetimezn` Define command to parse time in the format ⟨*h*⟩:⟨*m*⟩:⟨*s*⟩ ⟨*znh*⟩:⟨*znm*⟩. The results are stored in `\@dtm@hour`, `\@dtm@minute`, `\@dtm@second`, `\@dtm@timezonehour` and `\@dtm@timezoneminute`. The arguments are expanded.

```
\def\@dtm@parsetimezn#1:#2:#3 #4\@dtm@endparsetimezn{%
  \@dtm@parsetime#1:#2:#3\@dtm@endparsetime
  \@dtm@parsezone{#4}%
}
```

\@dtm@parsezone Define command to parse time zone in the format Z or ⟨*znh*⟩:⟨*znm*⟩. The results are stored in `\@dtm@timezonehour` and `\@dtm@timezoneminute`. The arguments are expanded in the event that registers are used.

```
\newcommand*{\@dtm@parsezone}[1]{%
  \ifstrequal{#1}{Z}%
  {%
    \def\@dtm@timezonehour{+00}%
    \def\@dtm@timezoneminute{00}%
  }%
  {%
    \@dtm@parse@zone#1\@dtm@endparse@zone
  }%
}
\def\@dtm@parse@zone#1:#2\@dtm@endparse@zone{%
  \edef\@dtm@timezonehour{\number#1}%
  \edef\@dtm@timezoneminute{\number#2}%
}
```

@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).

```
\def\@dtm@parsetimestamp#1-#2-#3T#4:#5:#6#7#8\@dtm@endparsetimestamp{%
  \@dtm@parsedate#1-#2-#3\@dtm@endparsedate
  \@dtm@parsetime#4:#5:#6#7\@dtm@endparsetime
  \@dtm@parsezone{#8}%
}
```

savefilemoddate Not available for some engines.

```
\newcommand*{\DTMsavefilemoddate}[2]{%
  \@dtm@warning{Your TeX engine doesn’t support accessing
  file modification dates}%
  \cslet{@dtm@#1@year}{0}%
  \cslet{@dtm@#1@month}{0}%
  \cslet{@dtm@#1@day}{0}%
  \cslet{@dtm@#1@dow}{-1}%
  \cslet{@dtm@#1@hour}{0}%
  \cslet{@dtm@#1@minute}{0}%
  \cslet{@dtm@#1@second}{0}%
  \cslet{@dtm@#1@TZhour}{0}%
  \cslet{@dtm@#1@TZminute}{0}%
}
```

savefrompdfdata Save a date-time stamp that’s specified in PDF format.

```
\newcommand*{\DTMsavefrompdfdata}[2]{%
  \edef\@dtm@tmp{#2}%
  \expandafter\@dtm@parsepdfdatetime\@dtm@tmp\@dtm@endparsepdfdatetime
  \cslet{@dtm@#1@year}{\@dtm@year}%
  \cslet{@dtm@#1@month}{\@dtm@month}%
  \cslet{@dtm@#1@day}{\@dtm@day}%
  \cslet{@dtm@#1@dow}{\@dtm@dow}%
  \cslet{@dtm@#1@hour}{\@dtm@hour}%
  \cslet{@dtm@#1@minute}{\@dtm@minute}%
  \cslet{@dtm@#1@second}{\@dtm@second}%
  \cslet{@dtm@#1@TZhour}{\@dtm@timezonehour}%
  \cslet{@dtm@#1@TZminute}{\@dtm@timezoneminute}%
}
```

Find out the current time. If `\dtm@pdfcreationdate` is defined, it can be fetched from that.

```
\ifdef\dtm@pdfcreationdate
{%
```

Define commands to parse `\dtm@pdfcreationdate`

```
  \def\@dtm@parsepdfdatetime#1:#2#3#4#5#6#7#8#9{%
    \def\@dtm@year{#2#3#4#5}%
    \def\@dtm@month{#6#7}%
    \def\@dtm@day{#8#9}%
    \@dtm@parsepdftime
  }
  \def\@dtm@parsepdftime#1#2#3#4#5#6#7\@dtm@endparsepdfdatetime{%
    \def\@dtm@hour{#1#2}%
    \def\@dtm@minute{#3#4}%
    \def\@dtm@second{#5#6}%
    \ifstrequal{#7}{Z}%
    {%
      \def\@dtm@timezonehour{00}%
      \def\@dtm@timezoneminute{00}%
    }%
    {%
      \@dtm@parsepdftimezone#7%
    }%
  }
  \def\@dtm@parsepdftimezone#1'#2'{%
    \def\@dtm@timezonehour{#1}%
    \def\@dtm@timezoneminute{#2}%
  }%
```

Now parse `\dtm@pdfcreationdate`

```
  \expandafter\@dtm@parsepdfdatetime\dtm@pdfcreationdate\@dtm@endparsepdfdatetime
```

Save the values.

```
  \let\@dtm@currentyear\@dtm@year
  \let\@dtm@currentmonth\@dtm@month
  \let\@dtm@currentday\@dtm@day
  \let\@dtm@currenthour\@dtm@hour
  \let\@dtm@currentminute\@dtm@minute
```

```
\let\@dtm@currentsecond\@dtm@second
\let\@dtm@currenttimezonehour\@dtm@timezonehour
\let\@dtm@currenttimezoneminute\@dtm@timezoneminute
%
```

LuaTEX doesn’t provide `\pdffilemoddate`.

```
\ifdef\pdffilemoddate
{%
  \renewcommand*{\DTMsavefilemoddate}[2]{%
    \expandafter\@dtm@parsepdfdatetime\pdffilemoddate{#2}\@dtm@endparsepdfdatetime
    \cslet{@dtm@#1@year}{\@dtm@year}%
    \cslet{@dtm@#1@month}{\@dtm@month}%
    \cslet{@dtm@#1@day}{\@dtm@day}%
    \cslet{@dtm@#1@dow}{\@dtm@dow}%
    \cslet{@dtm@#1@hour}{\@dtm@hour}%
    \cslet{@dtm@#1@minute}{\@dtm@minute}%
    \cslet{@dtm@#1@second}{\@dtm@second}%
    \cslet{@dtm@#1@TZhour}{\@dtm@timezonehour}%
    \cslet{@dtm@#1@TZminute}{\@dtm@timezoneminute}%
  }
}%
{%
  \ifdef\directlua
  {
```

Lua time zone information provided by `%z` is OS dependent, so this might not work.

```
    \renewcommand*{\DTMsavefilemoddate}[2]{%
      \expandafter\@dtm@parseluadatetime
        \directlua{tex.print(os.date(
           "\expandafter\@gobble\string\%Y-%
            \expandafter\@gobble\string\%m-%
            \expandafter\@gobble\string\%d-%
            \expandafter\@gobble\string\%w
            \expandafter\@gobble\string\%H:%
            \expandafter\@gobble\string\%M:%
            \expandafter\@gobble\string\%S
            \expandafter\@gobble\string\%z",
          lfs.attributes("#2").modification))}%
       \@dtm@endparseluadatetime
       \cslet{@dtm@#1@year}{\@dtm@year}%
       \cslet{@dtm@#1@month}{\@dtm@month}%
       \cslet{@dtm@#1@day}{\@dtm@day}%
       \cslet{@dtm@#1@dow}{\@dtm@dow}%
       \cslet{@dtm@#1@hour}{\@dtm@hour}%
       \cslet{@dtm@#1@minute}{\@dtm@minute}%
       \cslet{@dtm@#1@second}{\@dtm@second}%
       \cslet{@dtm@#1@TZhour}{\@dtm@TZhour}%
       \cslet{@dtm@#1@TZminute}{\@dtm@TZminute}%
    }
    \def\@dtm@parseluadatetime#1-#2-#3-#4 #5:#6:#7 #8\@dtm@endparseluadatetime{%
```

```
   \edef\@dtm@year{\number#1}%
   \edef\@dtm@month{\number#2}%
   \edef\@dtm@day{\number#3}%
   \edef\@dtm@dow{\number#4}%
   \edef\@dtm@hour{\number#5}%
   \edef\@dtm@minute{\number#6}%
   \edef\@dtm@second{\number#7}%
   \@dtm@parseluatimezone#8000000\@dtm@endparseluatimezone
}
\def\@dtm@parseluatimezone#1#2#3#4#5#6{%
   \ifstrequal{#1}{+}%
   {%
     \def\@dtm@TZhour{#1#2#3}%
     \ifstrequal{#4}{:}%
     {%
        \def\@dtm@TZminute{#5#6}%
     }%
     {%
        \def\@dtm@TZminute{#4#5}%
     }%
   }%
   {%
     \ifstrequal{#1}{-}%
     {%
        \def\@dtm@TZhour{#1#2#3}%
        \ifstrequal{#4}{:}%
        {%
           \def\@dtm@TZminute{#5#6}%
        }%
        {%
           \def\@dtm@TZminute{#4#5}%
        }%
     }%
     {%
       \ifstrequal{#1}{Z}%
       {%
         \def\@dtm@TZhour{0}%
         \def\@dtm@TZminute{0}%
       }%
       {%
         \def\@dtm@TZhour{#1#2}%
         \ifstrequal{#3}{:}%
         {%
            \def\@dtm@TZminute{#4#5}%
         }%
         {%
            \def\@dtm@TZminute{#3#4}%
         }%
       }%
```

```
           }%
         }%
         \@@dtm@parseluatimezone
      }
      \def\@@dtm@parseluatimezone#1\@dtm@endparseluatimezone{%
      }
    }
    {%
```

If `\directlua` isn't defined but texosquery has been loaded, we can use `\TeXOSQueryFileDate` if the shell escape is enabled.

```
      \ifdef\TeXOSQueryFileDate
      {
        \renewcommand*{\DTMsavefilemoddate}[2]{%
          \TeXOSQueryFileDate{\@dtm@tmp}{#2}%
          \ifdefempty\@dtm@tmp
          {%
```

OS query failed.

```
           \@dtm@warning{Your TeX engine doesn't support accessing
           file modification dates and the attempt to use texosquery
           failed}%
           \cslet{@dtm@#1@year}{0}%
           \cslet{@dtm@#1@month}{0}%
           \cslet{@dtm@#1@day}{0}%
           \cslet{@dtm@#1@dow}{-1}%
           \cslet{@dtm@#1@hour}{0}%
           \cslet{@dtm@#1@minute}{0}%
           \cslet{@dtm@#1@second}{0}%
           \cslet{@dtm@#1@TZhour}{0}%
           \cslet{@dtm@#1@TZminute}{0}%
          }%
          {%
            \expandafter\@dtm@parsepdfdatetime\@dtm@tmp\@dtm@endparsepdfdatetime
            \cslet{@dtm@#1@year}{\@dtm@year}%
            \cslet{@dtm@#1@month}{\@dtm@month}%
            \cslet{@dtm@#1@day}{\@dtm@day}%
            \cslet{@dtm@#1@dow}{\@dtm@dow}%
            \cslet{@dtm@#1@hour}{\@dtm@hour}%
            \cslet{@dtm@#1@minute}{\@dtm@minute}%
            \cslet{@dtm@#1@second}{\@dtm@second}%
            \cslet{@dtm@#1@TZhour}{\@dtm@timezonehour}%
            \cslet{@dtm@#1@TZminute}{\@dtm@timezoneminute}%
          }%
        }%
      }
      {}
    }
  }%
}%
```

```
{%
```

`\pdfcreationdate` not defined. By a process of elimination, the TeX engine is either XeTeX or it's very old. (Or it may be a new version of LuaTeX without luatex85.) In this case, the seconds and time zone can't be obtained. The hour and minute need to be calculated from TeX's `\time` primitive.

```
  \count@=\time\relax
  \divide\count@ by 60\relax
  \edef\@dtm@currenthour{\number\count@}%
  \multiply\count@ by -60\relax
  \advance\count@ by \time\relax
  \edef\@dtm@currentminute{\number\count@}%
  \newcommand*{\@dtm@currentsecond}{00}%
  \newcommand\@dtm@currenttimezonehour{00}%
  \newcommand\@dtm@currenttimezoneminute{00}%
```

Get the day, month and year from TeX's primitives.

```
  \edef\@dtm@currentyear{\number\year}%
  \edef\@dtm@currentmonth{\number\month}%
  \edef\@dtm@currentday{\number\day}%
}
```

Make `\DTMsavefilemoddate` robust.

```
\robustify\DTMsavefilemoddate
```

Current day of week defaults to `-1` (that is, ignore it).

`@dtm@currentdow`

```
\newcommand*{\@dtm@currentdow}{-1}
```

Allow XeLaTeX users a way of manually setting the current time zone.

`Msetcurrentzone`

```
\newcommand*{\DTMsetcurrentzone}[2]{%
 \renewcommand\@dtm@currenttimezonehour{#1}%
 \renewcommand\@dtm@currenttimezoneminute{#2}%
}
```

`\DTMtoday` Provided in case of conflict to obtain datetime2's version of `\today`.

```
\newcommand*{\DTMtoday}{%
\DTMdisplaydate
 {\@dtm@currentyear}%
 {\@dtm@currentmonth}%
 {\@dtm@currentday}%
 {\@dtm@currentdow}%
}
```

`\today` Version 1.4 dropped `\renewcommand` in case `\today` hasn't already been defined.

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

The scrlttr2 class redefines `\today` at the start of the document, so check for this.

```
\@ifclassloaded{scrlttr2}{\AtBeginDocument{\let\today\DTMtoday}}{}
```

`\DTMToday` First letter upper case version. Added to v1.4 to provide datetime2's version in case of conflict.

```
\newcommand*{\DTMToday}{%
 \DTMDisplaydate
  {\@dtm@currentyear}%
  {\@dtm@currentmonth}%
  {\@dtm@currentday}%
  {\@dtm@currentdow}%
}
```

`\Today`

```
\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.

```
\newcommand*\DTMdisplaydate[4]{%
  \number#1\dtm@yearmonthsep\DTMtwodigits{#2}\dtm@monthdaysep\DTMtwodigits{#3}%
}%
```

`\DTMDisplaydate` First letter upper case version. Defaults to `\DTMdisplaydate`.

```
\newcommand*{\DTMDisplaydate}{\DTMdisplaydate}
```

`\DTMfinaldot` Some date styles require the date to end with a period (full stop). These styles should use `\DTLfinaldot` for the terminating period. This allows the starred version of `\DTMdate` to locally redefine this command to do nothing when the date is required at the end of a sentence.

```
\newcommand*{\DTMfinaldot}{.}
```

`\DTMdate` Display date where date is specified in the format *⟨yyyy⟩*-*⟨mm⟩*-*⟨dd⟩*. Use `\expandafter` in case argument is a control sequence containing the date. This command isn't expandable. This now has a starred version that locally redefines `\DTMfinaldot` to do nothing.

```
\newrobustcmd*{\DTMdate}{\@ifstar\@sDTMdate\@DTMdate}
```

`\@DTMdate`

```
\newcommand*{\@DTMdate}[1]{%
  \expandafter\@dtm@parsedate#1\@dtm@endparsedate
  \DTMdisplaydate{\@dtm@year}{\@dtm@month}{\@dtm@day}{\@dtm@dow}%
}
```

`\@sDTMdate`

```
\newcommand*{\@sDTMdate}[1]{%
  {%
    \let\DTMfinaldot\empty
    \@DTMdate{#1}%
  }%
}
```

`\DTMDate` Upper case version. This now has a starred version that locally redefines `\DTMfinaldot` to do nothing.

```
\newrobustcmd*{\DTMDate}{\@ifstar\@sDTMDate\@DTMDate}
```

`\@sDTMDate`

```
\newcommand*{\@DTMDate}[1]{%
  \expandafter\@dtm@parsedate#1\@dtm@endparsedate
  \DTMDisplaydate{\@dtm@year}{\@dtm@month}{\@dtm@day}{\@dtm@dow}%
}
```

`\@sDTMDate`

```
\newcommand*{\@sDTMDate}[1]{%
  {%
    \let\DTMfinaldot\empty
    \@DTMDate{#1}%
  }%
}
```

`\DTMcurrenttime` Display the current time.

```
\newcommand*{\DTMcurrenttime}{%
 \DTMdisplaytime
  {\@dtm@currenthour}%
  {\@dtm@currentminute}%
  {\@dtm@currentsecond}%
}
```

`\DTMdisplaytime`

```
\DTMdisplaytime{⟨hour⟩}{⟨minute⟩}{⟨sec⟩}
```

Display the given time.

```
\newcommand*\DTMdisplaytime[3]{%
  \DTMtwodigits{#1}\dtm@hourminsep\DTMtwodigits{#2}%
  \ifDTMshowseconds\dtm@minsecsep\DTMtwodigits{#3}\fi
}%
```

`\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.

```
\newrobustcmd*{\DTMtime}[1]{%
  \@dtm@parsetime#1\@dtm@endparsetime
  \DTMdisplaytime{\@dtm@hour}{\@dtm@minute}{\@dtm@second}%
}
```

\DTMcurrentzone Display the current time zone.

```
\newcommand*{\DTMcurrentzone}{%
 \DTMdisplayzone
  {\@dtm@currenttimezonehour}%
  {\@dtm@currenttimezoneminute}%
}
```

\DTMdisplayzone Display time zone.

```
\newcommand*{\DTMdisplayzone}[2]{%
 \ifboolexpe
 { bool{DTMshowisoZ}
   and test{\ifnumequal{#1}{0}}
   and test{\ifnumequal{#2}{0}}
 }%
 {%
   Z%
 }%
 {%
  \ifnum#1<0\else+\fi\DTMtwodigits{#1}%
  \ifDTMshowzoneminutes\dtm@hourminsep\DTMtwodigits{#2}\fi
 }%
}
```

\DTMnow Current date, time and time zone.

```
\newcommand*{\DTMnow}{%
 \DTMdisplay
 {\@dtm@currentyear}
 {\@dtm@currentmonth}
 {\@dtm@currentday}
 {\@dtm@currentdow}
 {\@dtm@currenthour}%
 {\@dtm@currentminute}%
 {\@dtm@currentsecond}%
 {\@dtm@currenttimezonehour}%
 {\@dtm@currenttimezoneminute}%
}
```

\DTMNow Current date, time and time zone.

```
\newcommand*{\DTMNow}{%
 \DTMDisplay
 {\@dtm@currentyear}
 {\@dtm@currentmonth}
 {\@dtm@currentday}
 {\@dtm@currentdow}
 {\@dtm@currenthour}%
 {\@dtm@currentminute}%
 {\@dtm@currentsecond}%
 {\@dtm@currenttimezonehour}%
 {\@dtm@currenttimezoneminute}%
}
```

\DTMdisplay

\DTMdisplay{⟨*YYYY*⟩}{⟨*MM*⟩}{⟨*DD*⟩}{⟨*DOW*⟩}{⟨*hh*⟩}{⟨*mm*⟩}{⟨*ss*⟩}
{⟨*TZh*⟩}{⟨*TZm*⟩}

Display the date and time.

```
\newcommand*{\DTMdisplay}[9]{%
 \ifDTMshowdate
   \DTMdisplaydate{#1}{#2}{#3}{#4}%
   \dtm@datetimesep
 \fi
 \DTMdisplaytime
  {#5}%
  {#6}%
  {#7}%
 \ifDTMshowzone
  \dtm@timezonesep
  \DTMdisplayzone
   {#8}%
   {#9}%
 \fi
}
```

\DTMDisplay

\DTMDisplay{⟨*YYYY*⟩}{⟨*MM*⟩}{⟨*DD*⟩}{⟨*DOW*⟩}{⟨*hh*⟩}{⟨*mm*⟩}{⟨*ss*⟩}
{⟨*TZh*⟩}{⟨*TZm*⟩}

First letter upper case version. Defaults to `\DTMdisplay`.

```
\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

```
\newcommand*{\DTMtwodigits}[1]{%
 \ifnum#1<0
```

```
  -\DTMtwodigits{-#1}%
 \else
   \ifnum#1<100
     \ifnum#1<10
       0\number#1
     \else
        \number#1
     \fi
   \else
```

`\numexpr` rounds rather than truncates integer division, which is a little awkward to get around in an expandable context.

```
     \ifnum\numexpr#1-(#1/100)*100<0
       \number\numexpr#1-((#1/100)-1)*100\relax
     \else
       \number\numexpr#1-(#1/100)*100\relax
     \fi
   \fi
 \fi
}
```

`\DTMcentury` Expands to the given number divided by 100 rounded upwards (in absolute terms). Provided in case the user just wants the century.

```
\newcommand*{\DTMcentury}[1]{%
  \ifnum#1<0
    -\DTMcentury{-#1}%
  \else
    \ifnum\numexpr#1-(#1/100)*100<1
       \number\numexpr#1/100\relax
    \else
       \number\numexpr(#1/100)+1\relax
    \fi
  \fi
}
```

`\DTMdivhundred` Expands to the given number divided by 100.

```
\newcommand*{\DTMdivhundred}[1]{%
  \ifnum#1<0
    -\DTMdivhundred{-#1}%
  \else
    \ifnum\numexpr#1-(#1/100)*100<0
       \number\numexpr(#1)/100-1\relax
    \else
       \number\numexpr((#1)/100)\relax
    \fi
  \fi
}
```

`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.

```
\newcommand*{\DTMtexorpdfstring}[2]{#1}
\AtBeginDocument{%
  \@ifpackageloaded{hyperref}%
  {%
    \renewcommand*{\DTMtexorpdfstring}{\texorpdfstring}%
  }%
  {}%
}
```

Access separator:

\DTMsep

```
\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`.

```
\newcommand*{\DTMnewdatestyle}[2]{%
  \ifcsdef{@dtm@datestyle@#1}%
  {%
    \PackageError{datetime2}{Date style '#1' already exists}{}%
  }%
  {%
     \csdef{@dtm@datestyle@#1}{#2}%
  }%
}
```

Mrenewdatestyle

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.

```
\newcommand*{\DTMrenewdatestyle}[2]{%
  \ifcsundef{@dtm@datestyle@#1}%
  {%
    \PackageError{datetime2}{Date style '#1' doesn't exist}{}%
  }%
  {%
     \csdef{@dtm@datestyle@#1}{#2}%
  }%
}
```

rovidedatestyle

Define a date style if it doesn't already exist.

```
\newcommand*{\DTMprovidedatestyle}[2]{%
  \ifcsdef{@dtm@datestyle@#1}%
  {%
  }%
  {%
```

```
     \csdef{@dtm@datestyle@#1}{#2}%
  }%
}
```

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`.

```
\newcommand*{\DTMnewtimestyle}[2]{%
  \ifcsdef{@dtm@timestyle@#1}%
  {%
    \PackageError{datetime2}{Time style ‘#1’ already exists}{}%
  }%
  {%
     \csdef{@dtm@timestyle@#1}{#2}%
  }%
}
```

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.

```
\newcommand*{\DTMrenewtimestyle}[2]{%
  \ifcsundef{@dtm@timestyle@#1}%
  {%
    \PackageError{datetime2}{Time style ‘#1’ doesn’t exist}{}%
  }%
  {%
     \csdef{@dtm@timestyle@#1}{#2}%
  }%
}
```

rovidetimestyle Define a time style if it doesn’t already exist.

```
\newcommand*{\DTMprovidetimestyle}[2]{%
  \ifcsdef{@dtm@timestyle@#1}%
  {%
  }%
  {%
     \csdef{@dtm@timestyle@#1}{#2}%
  }%
}
```

DTMnewzonestyle Define a new zone-only style. This should only redefine `\DTMdisplayzone`, which may or may not use the separator `\dtm@hourminsep`.

```
\newcommand*{\DTMnewzonestyle}[2]{%
  \ifcsdef{@dtm@zonestyle@#1}%
  {%
    \PackageError{datetime2}{Zone style ‘#1’ already exists}{}%
  }%
  {%
     \csdef{@dtm@zonestyle@#1}{#2}%
  }%
}
```

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.

```
\newcommand*{\DTMrenewzonestyle}[2]{%
  \ifcsundef{@dtm@zonestyle@#1}%
  {%
    \PackageError{datetime2}{Zone style ‘#1’ doesn’t exist}{}%
  }%
  {%
     \csdef{@dtm@zonestyle@#1}{#2}%
  }%
}
```

rovidezonestyle

Defines a new zone style if it doesn’t already exist.

```
\newcommand*{\DTMprovidezonestyle}[2]{%
  \ifcsdef{@dtm@zonestyle@#1}%
  {%
  }%
  {%
     \csdef{@dtm@zonestyle@#1}{#2}%
  }%
}
```

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 ⟨*TZh*⟩:⟨*TZm*⟩ is stored in `\@dtm@zonemap@`⟨*TZh*⟩`:`⟨*TZm*⟩.

\DTMdefzonemap

`\DTMdefzonemap{`⟨*TZh*⟩`}{`⟨*TZm*⟩`}{`⟨*map*⟩`}`

This will override any previous mapping for the given time zone.

```
\newcommand*{\DTMdefzonemap}[3]{%
  \csdef{@dtm@zonemap@\DTMtwodigits{#1}:\DTMtwodigits{#2}}{#3}%
}
```

onemapordefault

Expands to the mapping or the default if not defined.

```
\newcommand*{\DTMusezonemapordefault}[2]{%
  \ifcsundef{@dtm@zonemap@\DTMtwodigits{#1}:\DTMtwodigits{#2}}%
  {%
    \ifnum#1<0\else+\fi
    \DTMtwodigits{#1}%
    \ifDTMshowzoneminutes\DTMsep{hourmin}\DTMtwodigits{#2}\fi
  }%
  {\csname @dtm@zonemap@\DTMtwodigits{#1}:\DTMtwodigits{#2}\endcsname}%
}
```

\DTMusezonemap Expands to the mapping. (No check if defined.)

```
\newcommand*{\DTMusezonemap}[2]{%
  \csname @dtm@zonemap@\DTMtwodigits{#1}:\DTMtwodigits{#2}\endcsname
}
```

\DTMhaszonemap

```
\newcommand*{\DTMhaszonemap}[4]{%
  \ifcsundef{@dtm@zonemap@\DTMtwodigits{#1}:\DTMtwodigits{#2}}{#4}{#3}%
}
```

\DTMclearmap Undefines the given zone mapping. No check is made to determine if the map exists.

```
\newcommand*{\DTMclearmap}[2]{%
 \csundef{@dtm@zonemap@\DTMtwodigits{#1}:\DTMtwodigits{#2}}%
}
```

\DTMshowmap Debugging command.

```
\newcommand*{\DTMshowmap}[2]{%
 \csshow{@dtm@zonemap@\DTMtwodigits{#1}:\DTMtwodigits{#2}}%
}
```

\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.

```
\newcommand*{\DTMresetzones}{}
```

DTMNatoZoneMaps Provide a command to set the time zone abbreviations to the military/NATO style.

```
\newcommand*{\DTMNatoZoneMaps}{%
  \defzonemap{01}{00}{A}% Alpha time zone
  \defzonemap{02}{00}{B}% Bravo time zone
  \defzonemap{03}{00}{C}% Charlie time zone
  \defzonemap{04}{00}{D}% Delta time zone
  \defzonemap{05}{00}{E}% Echo time zone
  \defzonemap{06}{00}{F}% Foxtrot time zone
  \defzonemap{07}{00}{G}% Golf time zone
  \defzonemap{08}{00}{H}% Hotel time zone
  \defzonemap{09}{00}{I}% India time zone
  \defzonemap{10}{00}{K}% Kilo time zone
  \defzonemap{11}{00}{L}% Lima time zone
  \defzonemap{12}{00}{M}% Mike time zone
  \defzonemap{-01}{00}{N}% November time zone
  \defzonemap{-02}{00}{O}% Oscar time zone
  \defzonemap{-03}{00}{P}% Papa time zone
  \defzonemap{-04}{00}{Q}% Quebec time zone
  \defzonemap{-05}{00}{R}% Romeo time zone
  \defzonemap{-06}{00}{S}% Sierra time zone
  \defzonemap{-07}{00}{T}% Tango time zone
  \defzonemap{-08}{00}{U}% Uniform time zone
  \defzonemap{-09}{00}{V}% Victor time zone
  \defzonemap{-10}{00}{W}% Whiskey time zone
  \defzonemap{-11}{00}{X}% X-ray time zone
  \defzonemap{-12}{00}{Y}% Yankee time zone
  \defzonemap{00}{00}{Z}% Zulu time zone
}
```

\DTMifhasstyle

`\DTMifhasstyle{⟨label⟩}{⟨true⟩}{⟨false⟩}`

Test to determine if style exists.

```
\newcommand*{\DTMifhasstyle}[3]{%
  \ifcsdef{@dtm@style@#1}{#2}{#3}%
}
```

Mifhasdatestyle

`\DTMifhasdatestyle{⟨label⟩}{⟨true⟩}{⟨false⟩}`

Test to determine if partial date style exists.

```
\newcommand*{\DTMifhasdatestyle}[3]{%
  \ifcsdef{@dtm@datestyle@#1}{#2}{#3}%
}
```

Mifhastimestyle

`\DTMifhastimestyle{⟨label⟩}{⟨true⟩}{⟨false⟩}`

Test to determine if time style exists.

```
\newcommand*{\DTMifhastimestyle}[3]{%
  \ifcsdef{@dtm@timestyle@#1}{#2}{#3}%
}
```

Mifhaszonestyle

`\DTMifhaszonestyle{⟨label⟩}{⟨true⟩}{⟨false⟩}`

Test to determine if time zone style exists.

```
\newcommand*{\DTMifhaszonestyle}[3]{%
  \ifcsdef{@dtm@zonestyle@#1}{#2}{#3}%
}
```

\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`.

```
\newcommand*{\DTMnewstyle}[5]{%
  \DTMifhasstyle{#1}%
  {%
    \PackageError{datetime2}{Style ‘#1’ already exists}{}%
  }%
  {%
    \DTMnewdatestyle{#1}{#2}%
    \DTMnewtimestyle{#1}{#3}%
    \DTMnewzonestyle{#1}{#4}%
     \csdef{@dtm@style@#1}{%
        \csuse{@dtm@datestyle@#1}%
        \csuse{@dtm@timestyle@#1}%
        \csuse{@dtm@zonestyle@#1}%
        #5%
     }%
  }%
}
```

\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`.

```
\newcommand*{\DTMrenewstyle}[5]{%
  \DTMifhasstyle{#1}%
  {%
    \DTMrenewdatestyle{#1}{#2}%
    \DTMrenewtimestyle{#1}{#3}%
    \DTMrenewzonestyle{#1}{#4}%
     \csdef{@dtm@style@#1}{%
        \csuse{@dtm@datestyle@#1}%
        \csuse{@dtm@timestyle@#1}%
        \csuse{@dtm@zonestyle@#1}%
        #5%
     }%
  }%
  {%
    \PackageError{datetime2}{Style ‘#1’ doesn’t exist}{}%
  }%
}
```

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.

```
\newcommand*{\DTMprovidestyle}[5]{%
  \DTMifhasstyle{#1}%
  {%
  }%
  {%
    \DTMprovidedatestyle{#1}{#2}%
    \DTMprovidetimestyle{#1}{#3}%
    \DTMprovidezonestyle{#1}{#4}%
     \csdef{@dtm@style@#1}{%
        \csuse{@dtm@datestyle@#1}%
        \csuse{@dtm@timestyle@#1}%
        \csuse{@dtm@zonestyle@#1}%
        #5%
     }%
  }%
}
```

DTMsetdatestyle

```
\newrobustcmd*{\DTMsetdatestyle}[1]{%
  \ifcsdef{@dtm@datestyle@#1}%
  {\csuse{@dtm@datestyle@#1}}%
  {%
    \PackageError{datetime2}{Date style ‘#1’ not defined}{}%
  }%
}
```

DTMsettimestyle

```
\newrobustcmd*{\DTMsettimestyle}[1]{%
  \ifcsdef{@dtm@timestyle@#1}%
  {\csuse{@dtm@timestyle@#1}}%
  {%
    \PackageError{datetime2}{Time style ‘#1’ not defined}{}%
  }%
}
```

DTMsetzonestyle

```
\newrobustcmd*{\DTMsetzonestyle}[1]{%
  \ifcsdef{@dtm@zonestyle@#1}%
  {\csuse{@dtm@zonestyle@#1}}%
  {%
    \PackageError{datetime2}{Zone style ‘#1’ not defined}{}%
  }%
}
```

\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.

```
\newcommand*{\DTMtryregional}{%
  \@ifstar\s@dtm@tryregional\@dtm@tryregional
}
```

dtm@tryregional

```
\newcommand*{\@dtm@tryregional}[3][]{%
  \edef\@dtm@langcode{#2}%
  \edef\@dtm@countrycode{#3}%
  \s@dtm@tryregional[#1]{\@dtm@langcode}{\@dtm@countrycode}%
}
```

dtm@tryregional

```
\newcommand*{\s@dtm@tryregional}[3][]{%
 \def\@dtm@thisstyle{}%
 \edef\@dtm@root{#1}%
 \ifdefempty{#2}{}%
 {%
   \let\@dtm@thisstyle#2%
```

Determine the root language name if not already provided in the optional argument.

```
   \ifdefempty\@dtm@root
   {%
     \edef\@dtm@root{\TrackedLanguageFromIsoCode{639-1}{#2}}%
     \ifdefempty\@dtm@root
     {%
       \edef\@dtm@root{\TrackedLanguageFromIsoCode{639-2}{#2}}%
     }%
     {}%
   }%
   {}%
 }%
 \ifdefempty{#3}{}%
 {%
   \ifdefempty\@dtm@thisstyle
   {\let\@dtm@thisstyle#3}%
   {\eappto\@dtm@thisstyle{-#3}}%
 }%
 \ifdefempty\@dtm@thisstyle
 {}%
 {%
```

```
   \DTMifcaseregional
   {}%
   {%
     \DTMifhasstyle{\@dtm@thisstyle}%
     {%
       \csuse{@dtm@style@\@dtm@thisstyle}%
     }%
     {%
       \ifdefempty\@dtm@root
       {}%
       {%
         \DTMifhasstyle{\@dtm@root}%
         {%
           \csuse{@dtm@style@\@dtm@root}%
         }%
         {}%
       }%
     }%
   }%
   {%
     \DTMifhasstyle{\@dtm@thisstyle-numeric}%
     {%
       \csuse{@dtm@style@\@dtm@thisstyle-numeric}%
     }%
     {%
       \ifdefempty\@dtm@root
       {}%
       {%
         \DTMifhasstyle{\@dtm@root-numeric}%
         {%
           \csuse{@dtm@style@\@dtm@root-numeric}%
         }%
         {}%
       }%
     }%
   }%
 }%
}
```

`\DTMsetregional`

```
\newcommand*{\DTMsetregional}[1][text]{%
  \DTMsetup{useregional=#1}%
  \ifstrequal{#1}{false}%
  {%
    \DTMsetstyle{default}%
  }%
  {%
    \ifcsdef{date\languagename}%
    {%
```

```
      \csuse{date\languagename}%
    }%
    {%
```

Iterate through dialect list.

```
      \ForEachTrackedDialect{\@dtm@thisdialect}%
      {%
        \edef\@dtm@lang{\TrackedLanguageFromDialect\@dtm@thisdialect}%
        \edef\@dtm@langcode{\TrackedIsoCodeFromLanguage{639-1}{\@dtm@lang}}%
        \ifdefempty\@dtm@langcode
        {%
          \edef\@dtm@langcode{\TrackedIsoCodeFromLanguage{639-2}{\@dtm@lang}}%
        }%
        {}%
        \edef\@dtm@countrycode{%
          \TrackedIsoCodeFromLanguage{3166-1}{\@dtm@thisdialect}}%
        \s@dtm@tryregional[\@dtm@lang]{\@dtm@langcode}{\@dtm@countrycode}%
      }%
    }%
  }%
}
```

\DTMsetstyle

```
\newrobustcmd*{\DTMsetstyle}[1]{%
  \DTMifhasstyle{#1}%
  {\csuse{@dtm@style@#1}}%
  {%
     \let\dtm@unknownstyle\@dtm@unknownstyle
     \ifcsdef{@dtm@datestyle#1}%
       {\csuse{@dtm@datestyle@#1}\let\dtm@unknownstyle\@dtm@unknown@style}%
       {\@dtm@warning{No date style `#1' defined}}%
     \ifcsdef{@dtm@timestyle#1}%
       {\csuse{@dtm@timestyle@#1}\let\dtm@unknownstyle\@dtm@unknown@style}%
       {\@dtm@warning{No time style `#1' defined}}%
     \ifcsdef{@dtm@zonestyle#1}%
       {\csuse{@dtm@zonestyle@#1}\let\dtm@unknownstyle\@dtm@unknown@style}%
       {\@dtm@warning{No zone style `#1' defined}}%
     \dtm@unknownstyle{#1}%
  }%
}
```

tm@unknownstyle

```
\newcommand*{\@dtm@unknownstyle}[1]{%
  \PackageError{datetime2}{Unknown style `#1'}{}%
}
```

m@unknown@style

```
\newcommand*{\@dtm@unknown@style}[1]{%
  \@dtm@warning{No full style `#1' defined}{}%
}
```

Define `default` style:

```
\DTMnewstyle
 {default}%label
 {% date style
    \renewcommand*\DTMdisplaydate[4]{%
      \number##1\DTMsep{yearmonth}\DTMtwodigits{##2}%
      \DTMsep{monthday}\DTMtwodigits{##3}%
    }%
    \renewcommand*{\DTMDisplaydate}{\DTMdisplaydate}%
 }%
 {% time style
    \renewcommand*\DTMdisplaytime[3]{%
      \DTMtwodigits{##1}\DTMsep{hourmin}\DTMtwodigits{##2}%
      \ifDTMshowseconds\DTMsep{minsec}\DTMtwodigits{##3}\fi
    }%
 }%
 {% zone style
   \renewcommand*{\DTMdisplayzone}[2]{%
     \ifboolexpe
     { bool{DTMshowisoZ}
       and test{\ifnumequal{##1}{0}}
       and test{\ifnumequal{##2}{0}}
     }%
     {%
       Z%
     }%
     {%
```

A space is needed after the conditional to prevent an unwanted `\relax` from being inserted if the time zone is negative.

```
      \ifnum##1<0 \else+\fi\DTMtwodigits{##1}%
      \ifDTMshowzoneminutes\DTMsep{hourmin}\DTMtwodigits{##2}\fi
     }%
   }%
 }%
 {% full style
   \renewcommand*{\DTMdisplay}[9]{%
    \ifDTMshowdate
     \DTMdisplaydate{##1}{##2}{##3}{##4}%
     \DTMsep{datetime}%
    \fi
    \DTMdisplaytime
     {##5}%
     {##6}%
     {##7}%
    \ifDTMshowzone
     \DTMsep{timezone}%
     \DTMdisplayzone
      {##8}%
      {##9}%
    \fi
   }%
   \renewcommand*{\DTMDisplay}{\DTMdisplay}%
 }
```

Define `iso` style which ignores the separator settings:

```
\DTMnewstyle
{iso}%label
{% date style
   \renewcommand*\DTMdisplaydate[4]{%
     \number##1-\DTMtwodigits{##2}-\DTMtwodigits{##3}%
   }%
   \renewcommand*{\DTMDisplaydate}{\DTMdisplaydate}%
}%
{% time style
   \renewcommand*\DTMdisplaytime[3]{%
     \DTMtwodigits{##1}:\DTMtwodigits{##2}%
     \ifDTMshowseconds:\DTMtwodigits{##3}\fi
   }%
}%
{% zone style
  \renewcommand*{\DTMdisplayzone}[2]{%
    \ifboolexpe
    { bool{DTMshowisoZ}
      and test{\ifnumequal{##1}{0}}
      and test{\ifnumequal{##2}{0}}
    }%
    {%
      Z%
    }%
    {%
```

A space is needed after the conditional to prevent an unwanted `\relax` from being inserted if the time zone is negative.

```
       \ifnum##1<0 \else+\fi\DTMtwodigits{##1}%
       \ifDTMshowzoneminutes:\DTMtwodigits{##2}\fi
    }%
  }%
}%
{% full style
  \renewcommand*{\DTMdisplay}[9]{%
   \ifDTMshowdate
    \DTMdisplaydate{##1}{##2}{##3}{##4}%
    T%
   \fi
   \DTMdisplaytime
    {##5}%
    {##6}%
    {##7}%
   \ifDTMshowzone
    \DTMdisplayzone
       {##8}%
       {##9}%
   \fi
  }%
  \renewcommand*{\DTMDisplay}{\DTMdisplay}%
}
```

Define `pdf` style which converts into a format that can be used in `\pdfinfo`:

```
\DTMnewstyle
 {pdf}%label
 {% date style
    \renewcommand*\DTMdisplaydate[4]{%
      D:\number##1 % space intended
      \DTMtwodigits{##2}\DTMtwodigits{##3}%
    }%
    \renewcommand*{\DTMDisplaydate}{\DTMdisplaydate}%
 }%
 {% time style
    \renewcommand*\DTMdisplaytime[3]{%
      \DTMtwodigits{##1}\DTMtwodigits{##2}\DTMtwodigits{##3}%
    }%
 }%
 {% zone style
   \renewcommand*{\DTMdisplayzone}[2]{%
     \ifboolexpe
     { bool{DTMshowisoZ}
       and test{\ifnumequal{##1}{0}}
       and test{\ifnumequal{##2}{0}}
     }%
     {%
       Z%
     }%
     {%
```

A space is needed after the conditional to prevent an unwanted `\relax` from being inserted if the time zone is negative.

```
      \ifnum##1<0 \else+\fi\DTMtwodigits{##1}'\DTMtwodigits{##2}'%
     }%
   }%
 }%
 {% full style
   \renewcommand*{\DTMdisplay}[9]{%
    \DTMdisplaydate{##1}{##2}{##3}{##4}%
    \DTMdisplaytime{##5}{##6}{##7}%
    \DTMdisplayzone{##8}{##9}%
   }%
   \renewcommand*{\DTMDisplay}{\DTMdisplay}%
 }
```

Define yyyymd style:

```
\DTMnewstyle
 {yyyymd}%label
 {% date style
    \renewcommand*\DTMdisplaydate[4]{%
      \number##1
      \DTMsep{yearmonth}%
      \number##2
      \DTMsep{monthday}%
      \number##3
    }%
    \renewcommand*{\DTMDisplaydate}{\DTMdisplaydate}%
 }%
 {% time style
    \renewcommand*\DTMdisplaytime[3]{%
      \DTMtwodigits{##1}\DTMsep{hourmin}\DTMtwodigits{##2}%
      \ifDTMshowseconds\DTMsep{minsec}\DTMtwodigits{##3}\fi
    }%
 }%
 {% zone style
   \renewcommand*{\DTMdisplayzone}[2]{%
     \ifboolexpe
     { bool{DTMshowisoZ}
       and test{\ifnumequal{##1}{0}}
       and test{\ifnumequal{##2}{0}}
     }%
     {%
       Z%
     }%
     {%
```

A space is needed after the conditional to prevent an unwanted `\relax` from being inserted if the time zone is negative.

```
      \ifnum##1<0 \else+\fi\DTMtwodigits{##1}%
      \ifDTMshowzoneminutes\DTMsep{hourmin}\DTMtwodigits{##2}\fi
     }%
   }%
 }%
 {% full style
   \renewcommand*{\DTMdisplay}[9]{%
    \ifDTMshowdate
     \DTMdisplaydate{##1}{##2}{##3}{##4}%
     \DTMsep{datetime}%
    \fi
    \DTMdisplaytime
     {##5}%
     {##6}%
     {##7}%
    \ifDTMshowzone
```

```
     \DTMsep{timezone}%
     \DTMdisplayzone
      {##8}%
      {##9}%
    \fi
   }%
   \renewcommand*{\DTMDisplay}{\DTMdisplay}%
 }
```

Define ddmmyyyy style:

```
\DTMnewstyle
 {ddmmyyyy}%label
 {% date style
    \renewcommand*\DTMdisplaydate[4]{%
      \DTMtwodigits{##3}\DTMsep{monthday}%
      \DTMtwodigits{##2}\DTMsep{yearmonth}%
      \number##1
    }%
    \renewcommand*{\DTMDisplaydate}{\DTMdisplaydate}%
 }%
 {% time style
    \renewcommand*\DTMdisplaytime[3]{%
      \DTMtwodigits{##1}\DTMsep{hourmin}\DTMtwodigits{##2}%
      \ifDTMshowseconds\DTMsep{minsec}\DTMtwodigits{##3}\fi
    }%
 }%
 {% zone style
   \renewcommand*{\DTMdisplayzone}[2]{%
     \ifboolexpe
     { bool{DTMshowisoZ}
       and test{\ifnumequal{##1}{0}}
       and test{\ifnumequal{##2}{0}}
     }%
     {%
       Z%
     }%
     {%
```

A space is needed after the conditional to prevent an unwanted `\relax` from being inserted if the time zone is negative.

```
      \ifnum##1<0 \else+\fi\DTMtwodigits{##1}%
      \ifDTMshowzoneminutes\DTMsep{hourmin}\DTMtwodigits{##2}\fi
     }%
   }%
 }%
 {% full style
   \renewcommand*{\DTMdisplay}[9]{%
    \ifDTMshowdate
     \DTMdisplaydate{##1}{##2}{##3}{##4}%
     \DTMsep{datetime}%
    \fi
    \DTMdisplaytime
     {##5}%
     {##6}%
     {##7}%
    \ifDTMshowzone
     \DTMsep{timezone}%
     \DTMdisplayzone
      {##8}%
      {##9}%
    \fi
   }%
   \renewcommand*{\DTMDisplay}{\DTMdisplay}%
 }
```

Define dmyyyy style:

```
\DTMnewstyle
 {dmyyyy}%label
 {% date style
    \renewcommand*\DTMdisplaydate[4]{%
      \number##3
      \DTMsep{monthday}%
      \number##2
      \DTMsep{yearmonth}%
      \number##1
    }%
    \renewcommand*{\DTMDisplaydate}{\DTMdisplaydate}%
 }%
 {% time style
    \renewcommand*\DTMdisplaytime[3]{%
      \DTMtwodigits{##1}\DTMsep{hourmin}\DTMtwodigits{##2}%
      \ifDTMshowseconds\DTMsep{minsec}\DTMtwodigits{##3}\fi
    }%
 }%
 {% zone style
   \renewcommand*{\DTMdisplayzone}[2]{%
     \ifboolexpe
     { bool{DTMshowisoZ}
       and test{\ifnumequal{##1}{0}}
       and test{\ifnumequal{##2}{0}}
     }%
     {%
       Z%
     }%
     {%
```

A space is needed after the conditional to prevent an unwanted `\relax` from being inserted if the time zone is negative.

```
      \ifnum##1<0 \else+\fi\DTMtwodigits{##1}%
      \ifDTMshowzoneminutes\DTMsep{hourmin}\DTMtwodigits{##2}\fi
```

```
     }%
   }%
 }%
 {% full style
   \renewcommand*{\DTMdisplay}[9]{%
    \ifDTMshowdate
     \DTMdisplaydate{##1}{##2}{##3}{##4}%
     \DTMsep{datetime}%
    \fi
    \DTMdisplaytime
     {##5}%
     {##6}%
     {##7}%
    \ifDTMshowzone
     \DTMsep{timezone}%
     \DTMdisplayzone
      {##8}%
      {##9}%
    \fi
   }%
   \renewcommand*{\DTMDisplay}{\DTMdisplay}%
 }
```

Define dmyy style:

```
\DTMnewstyle
 {dmyy}%label
 {% date style
    \renewcommand*\DTMdisplaydate[4]{%
      \number##3 % space intended
      \DTMsep{monthday}%
      \number##2 % space intended
      \DTMsep{yearmonth}%
      \DTMtwodigits{##1}%
    }%
    \renewcommand*{\DTMDisplaydate}{\DTMdisplaydate}%
 }%
 {% time style
    \renewcommand*\DTMdisplaytime[3]{%
      \DTMtwodigits{##1}\DTMsep{hourmin}\DTMtwodigits{##2}%
      \ifDTMshowseconds\DTMsep{minsec}\DTMtwodigits{##3}\fi
    }%
 }%
 {% zone style
   \renewcommand*{\DTMdisplayzone}[2]{%
     \ifboolexpe
     { bool{DTMshowisoZ}
       and test{\ifnumequal{##1}{0}}
       and test{\ifnumequal{##2}{0}}
     }%
     {%
```

```
       Z%
     }%
     {%
```

A space is needed after the conditional to prevent an unwanted `\relax` from being inserted if the time zone is negative.

```
      \ifnum##1<0 \else+\fi\DTMtwodigits{##1}%
      \ifDTMshowzoneminutes\DTMsep{hourmin}\DTMtwodigits{##2}\fi
     }%
   }%
 }%
 {% full style
   \renewcommand*{\DTMdisplay}[9]{%
    \ifDTMshowdate
     \DTMdisplaydate{##1}{##2}{##3}{##4}%
     \DTMsep{datetime}%
    \fi
    \DTMdisplaytime
     {##5}%
     {##6}%
     {##7}%
    \ifDTMshowzone
     \DTMsep{timezone}%
     \DTMdisplayzone
      {##8}%
      {##9}%
    \fi
   }%
   \renewcommand*{\DTMDisplay}{\DTMdisplay}%
 }
```

Define ddmmyy style:

```
\DTMnewstyle
 {ddmmyy}%label
 {% date style
    \renewcommand*\DTMdisplaydate[4]{%
      \DTMtwodigits{##3}\DTMsep{monthday}%
      \DTMtwodigits{##2}\DTMsep{yearmonth}%
      \DTMtwodigits{##1}%
    }%
    \renewcommand*{\DTMDisplaydate}{\DTMdisplaydate}%
 }%
 {% time style
    \renewcommand*\DTMdisplaytime[3]{%
      \DTMtwodigits{##1}\DTMsep{hourmin}\DTMtwodigits{##2}%
      \ifDTMshowseconds\DTMsep{minsec}\DTMtwodigits{##3}\fi
    }%
 }%
 {% zone style
   \renewcommand*{\DTMdisplayzone}[2]{%
```

```
     \ifboolexpe
     { bool{DTMshowisoZ}
       and test{\ifnumequal{##1}{0}}
       and test{\ifnumequal{##2}{0}}
     }%
     {%
       Z%
     }%
     {%
```

A space is needed after the conditional to prevent an unwanted `\relax` from being inserted if the time zone is negative.

```
      \ifnum##1<0 \else+\fi\DTMtwodigits{##1}%
      \ifDTMshowzoneminutes\DTMsep{hourmin}\DTMtwodigits{##2}\fi
     }%
   }%
 }%
 {% full style
   \renewcommand*{\DTMdisplay}[9]{%
    \ifDTMshowdate
     \DTMdisplaydate{##1}{##2}{##3}{##4}%
     \DTMsep{datetime}%
    \fi
    \DTMdisplaytime
     {##5}%
     {##6}%
     {##7}%
    \ifDTMshowzone
     \DTMsep{timezone}%
     \DTMdisplayzone
      {##8}%
      {##9}%
    \fi
   }%
   \renewcommand*{\DTMDisplay}{\DTMdisplay}%
 }
```

Define `mmddyyyy` style:

```
\DTMnewstyle
 {mmddyyyy}%label
 {% date style
    \renewcommand*\DTMdisplaydate[4]{%
      \DTMtwodigits{##2}\DTMsep{monthday}%
      \DTMtwodigits{##3}\DTMsep{dayyear}%
      \number##1
    }%
    \renewcommand*{\DTMDisplaydate}{\DTMdisplaydate}%
 }%
 {% time style
    \renewcommand*\DTMdisplaytime[3]{%
```

```
     \DTMtwodigits{##1}\DTMsep{hourmin}\DTMtwodigits{##2}%
     \ifDTMshowseconds\DTMsep{minsec}\DTMtwodigits{##3}\fi
   }%
}%
{% zone style
  \renewcommand*{\DTMdisplayzone}[2]{%
    \ifboolexpe
    { bool{DTMshowisoZ}
      and test{\ifnumequal{##1}{0}}
      and test{\ifnumequal{##2}{0}}
    }%
    {%
      Z%
    }%
    {%
```

A space is needed after the conditional to prevent an unwanted `\relax` from being inserted if the time zone is negative.

```
     \ifnum##1<0 \else+\fi\DTMtwodigits{##1}%
     \ifDTMshowzoneminutes\DTMsep{hourmin}\DTMtwodigits{##2}\fi
    }%
  }%
}%
{% full style
  \renewcommand*{\DTMdisplay}[9]{%
   \ifDTMshowdate
    \DTMdisplaydate{##1}{##2}{##3}{##4}%
    \DTMsep{datetime}%
   \fi
   \DTMdisplaytime
    {##5}%
    {##6}%
    {##7}%
   \ifDTMshowzone
    \DTMsep{timezone}%
    \DTMdisplayzone
     {##8}%
     {##9}%
   \fi
  }%
  \renewcommand*{\DTMDisplay}{\DTMdisplay}%
}
```

Define `mdyyyy` style:

```
\DTMnewstyle
 {mdyyyy}%label
 {% date style
    \renewcommand*\DTMdisplaydate[4]{%
      \number##2 % space intended
      \DTMsep{monthday}%
```

```
     \number##3 % space intended
     \DTMsep{dayyear}%
     \number##1 % space intended
   }%
   \renewcommand*{\DTMDisplaydate}{\DTMdisplaydate}%
}%
{% time style
   \renewcommand*\DTMdisplaytime[3]{%
     \DTMtwodigits{##1}\DTMsep{hourmin}\DTMtwodigits{##2}%
     \ifDTMshowseconds\DTMsep{minsec}\DTMtwodigits{##3}\fi
   }%
}%
{% zone style
  \renewcommand*{\DTMdisplayzone}[2]{%
    \ifboolexpe
    { bool{DTMshowisoZ}
      and test{\ifnumequal{##1}{0}}
      and test{\ifnumequal{##2}{0}}
    }%
    {%
      Z%
    }%
    {%
```

A space is needed after the conditional to prevent an unwanted `\relax` from being inserted if the time zone is negative.

```
     \ifnum##1<0 \else+\fi\DTMtwodigits{##1}%
     \ifDTMshowzoneminutes\DTMsep{hourmin}\DTMtwodigits{##2}\fi
    }%
  }%
}%
{% full style
  \renewcommand*{\DTMdisplay}[9]{%
   \ifDTMshowdate
    \DTMdisplaydate{##1}{##2}{##3}{##4}%
    \DTMsep{datetime}%
   \fi
   \DTMdisplaytime
    {##5}%
    {##6}%
    {##7}%
   \ifDTMshowzone
    \DTMsep{timezone}%
    \DTMdisplayzone
     {##8}%
     {##9}%
   \fi
  }%
  \renewcommand*{\DTMDisplay}{\DTMdisplay}%
}
```

Define `mdyy` style:

```
\DTMnewstyle
 {mdyy}%label
 {% date style
    \renewcommand*\DTMdisplaydate[4]{%
      \number##2 % space intended
      \DTMsep{monthday}%
      \number##3 % space intended
      \DTMsep{dayyear}%
      \DTMtwodigits{##1}%
    }%
    \renewcommand*{\DTMDisplaydate}{\DTMdisplaydate}%
 }%
 {% time style
    \renewcommand*\DTMdisplaytime[3]{%
      \DTMtwodigits{##1}\DTMsep{hourmin}\DTMtwodigits{##2}%
      \ifDTMshowseconds\DTMsep{minsec}\DTMtwodigits{##3}\fi
    }%
 }%
 {% zone style
   \renewcommand*{\DTMdisplayzone}[2]{%
     \ifboolexpe
      { bool{DTMshowisoZ}
       and test{\ifnumequal{##1}{0}}
       and test{\ifnumequal{##2}{0}}
     }%
     {%
       Z%
     }%
     {%
```

A space is needed after the conditional to prevent an unwanted `\relax` from being inserted if the time zone is negative.

```
      \ifnum##1<0 \else+\fi\DTMtwodigits{##1}%
      \ifDTMshowzoneminutes\DTMsep{hourmin}\DTMtwodigits{##2}\fi
     }%
   }%
 }%
 {% full style
   \renewcommand*{\DTMdisplay}[9]{%
    \ifDTMshowdate
     \DTMdisplaydate{##1}{##2}{##3}{##4}%
     \DTMsep{datetime}%
    \fi
    \DTMdisplaytime
     {##5}%
     {##6}%
     {##7}%
    \ifDTMshowzone
```

```
     \DTMsep{timezone}%
     \DTMdisplayzone
      {##8}%
      {##9}%
    \fi
   }%
   \renewcommand*{\DTMDisplay}{\DTMdisplay}%
 }
```

Define `mmddyy` style:

```
\DTMnewstyle
 {mmddyy}%label
 {% date style
    \renewcommand*\DTMdisplaydate[4]{%
      \DTMtwodigits{##2}\DTMsep{monthday}%
      \DTMtwodigits{##3}\DTMsep{dayyear}%
      \DTMtwodigits{##1}%
    }%
    \renewcommand*{\DTMDisplaydate}{\DTMdisplaydate}%
 }%
 {% time style
    \renewcommand*\DTMdisplaytime[3]{%
      \DTMtwodigits{##1}\DTMsep{hourmin}\DTMtwodigits{##2}%
      \ifDTMshowseconds\DTMsep{minsec}\DTMtwodigits{##3}\fi
    }%
 }%
 {% zone style
   \renewcommand*{\DTMdisplayzone}[2]{%
     \ifboolexpe
     { bool{DTMshowisoZ}
       and test{\ifnumequal{##1}{0}}
       and test{\ifnumequal{##2}{0}}
     }%
     {%
       Z%
     }%
     {%
```

A space is needed after the conditional to prevent an unwanted `\relax` from being inserted if the time zone is negative.

```
      \ifnum##1<0 \else+\fi\DTMtwodigits{##1}%
      \ifDTMshowzoneminutes\DTMsep{hourmin}\DTMtwodigits{##2}\fi
     }%
   }%
 }%
 {% full style
   \renewcommand*{\DTMdisplay}[9]{%
    \ifDTMshowdate
     \DTMdisplaydate{##1}{##2}{##3}{##4}%
     \DTMsep{datetime}%
    \fi
    \DTMdisplaytime
     {##5}%
     {##6}%
     {##7}%
    \ifDTMshowzone
     \DTMsep{timezone}%
     \DTMdisplayzone
      {##8}%
      {##9}%
    \fi
   }%
   \renewcommand*{\DTMDisplay}{\DTMdisplay}%
 }
```

Define hmmss time style

```
\DTMnewtimestyle
 {hmmss}% label
 {%
    \renewcommand*\DTMdisplaytime[3]{%
      \number##1
      \DTMsep{hourmin}\DTMtwodigits{##2}%
      \ifDTMshowseconds\DTMsep{minsec}\DTMtwodigits{##3}\fi
    }%
 }%
```

Define map zone style

```
\DTMnewzonestyle
 {map}% label
 {%
    \renewcommand*\DTMdisplaytime[3]{%
      \DTMusezonemapordefault{##1}{##2}%
    }%
 }%
```

Define hhmm zone style

```
\DTMnewzonestyle
 {hhmm}% label
 {%
    \renewcommand*\DTMdisplaytime[3]{%
```

A space is needed after the conditional to prevent an unwanted `\relax` from being inserted if the time zone is negative.

```
      \ifnum##1<0 \else+\fi\DTMtwodigits{##1}%
      \ifDTMshowzoneminutes\DTMsep{hourmin}\DTMtwodigits{##2}\fi
    }%
 }
```

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.

```
\newrobustcmd*{\DTMsavedate}[2]{%
  \expandafter\@dtm@parsedate#2\@dtm@endparsedate
  \cslet{@dtm@#1@year}{\@dtm@year}%
  \cslet{@dtm@#1@month}{\@dtm@month}%
  \cslet{@dtm@#1@day}{\@dtm@day}%
  \cslet{@dtm@#1@dow}{\@dtm@dow}%
  \ifcsundef{@dtm@#1@hour}{\csdef{@dtm@#1@hour}{0}}{}%
  \ifcsundef{@dtm@#1@minute}{\csdef{@dtm@#1@minute}{0}}{}%
  \ifcsundef{@dtm@#1@second}{\csdef{@dtm@#1@second}{0}}{}%
  \ifcsundef{@dtm@#1@TZhour}{\csdef{@dtm@#1@TZhour}{0}}{}%
  \ifcsundef{@dtm@#1@TZminute}{\csdef{@dtm@#1@TZminute}{0}}{}%
}
```

`savenoparsedate` Save the date without parsing the ⟨*YYYY*⟩-⟨*MM*⟩-⟨*DD*⟩ format.

```
\newrobustcmd*{\DTMsavenoparsedate}[5]{%
  \csedef{@dtm@#1@year}{\number#2}%
  \csedef{@dtm@#1@month}{\number#3}%
  \csedef{@dtm@#1@day}{\number#4}%
  \csedef{@dtm@#1@dow}{\number#5}%
  \ifcsundef{@dtm@#1@hour}{\csdef{@dtm@#1@hour}{0}}{}%
  \ifcsundef{@dtm@#1@minute}{\csdef{@dtm@#1@minute}{0}}{}%
  \ifcsundef{@dtm@#1@second}{\csdef{@dtm@#1@second}{0}}{}%
  \ifcsundef{@dtm@#1@TZhour}{\csdef{@dtm@#1@TZhour}{0}}{}%
  \ifcsundef{@dtm@#1@TZminute}{\csdef{@dtm@#1@TZminute}{0}}{}%
}
```

`\DTMsavetime` Save the time specified in the format ⟨*hh*⟩:⟨*mm*⟩:⟨*ss*⟩. `\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.

```
\newrobustcmd*{\DTMsavetime}[2]{%
  \expandafter\@dtm@parsetime#2\@dtm@endparsetime
  \cslet{@dtm@#1@hour}{\@dtm@hour}%
  \cslet{@dtm@#1@minute}{\@dtm@minute}%
  \cslet{@dtm@#1@second}{\@dtm@second}%
  \ifcsundef{@dtm@#1@year}{\csdef{@dtm@#1@year}{0}}{}%
  \ifcsundef{@dtm@#1@month}{\csdef{@dtm@#1@month}{0}}{}%
  \ifcsundef{@dtm@#1@day}{\csdef{@dtm@#1@day}{0}}{}%
  \ifcsundef{@dtm@#1@dow}{\csdef{@dtm@#1@dow}{-1}}{}%
  \ifcsundef{@dtm@#1@TZhour}{\csdef{@dtm@#1@TZhour}{0}}{}%
  \ifcsundef{@dtm@#1@TZminute}{\csdef{@dtm@#1@TZminute}{0}}{}%
}
```

`\DTMsavetimezn` Save the time (including zone) specified in the format ⟨*hh*⟩:⟨*mm*⟩:⟨*ss*⟩ ⟨*tzh*⟩:⟨*tzm*⟩. `\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.

```
\newrobustcmd*{\DTMsavetimezn}[2]{%
  \expandafter\@dtm@parsetimezn#2\@dtm@endparsetimezn
  \cslet{@dtm@#1@hour}{\@dtm@hour}%
  \cslet{@dtm@#1@minute}{\@dtm@minute}%
  \cslet{@dtm@#1@second}{\@dtm@second}%
  \cslet{@dtm@#1@TZhour}{\@dtm@timezonehour}%
  \cslet{@dtm@#1@TZminute}{\@dtm@timezoneminute}%
  \ifcsundef{@dtm@#1@year}{\csdef{@dtm@#1@year}{0}}{}%
  \ifcsundef{@dtm@#1@month}{\csdef{@dtm@#1@month}{0}}{}%
  \ifcsundef{@dtm@#1@day}{\csdef{@dtm@#1@day}{0}}{}%
  \ifcsundef{@dtm@#1@dow}{\csdef{@dtm@#1@dow}{-1}}{}%
}
```

`TMsavetimestamp` Save the time (including zone) specified in the format ⟨*YYYY*⟩-⟨*MM*⟩-⟨*DD*⟩T⟨*hh*⟩:⟨*mm*⟩:⟨*ss*⟩⟨*time zone*⟩

The first argument is the label.

```
\newrobustcmd*{\DTMsavetimestamp}[2]{%
  \expandafter\@dtm@parsetimestamp#2\@dtm@endparsetimestamp
  \cslet{@dtm@#1@year}{\@dtm@year}%
  \cslet{@dtm@#1@month}{\@dtm@month}%
  \cslet{@dtm@#1@day}{\@dtm@day}%
  \cslet{@dtm@#1@dow}{\@dtm@dow}%
  \cslet{@dtm@#1@hour}{\@dtm@hour}%
  \cslet{@dtm@#1@minute}{\@dtm@minute}%
  \cslet{@dtm@#1@second}{\@dtm@second}%
  \cslet{@dtm@#1@TZhour}{\@dtm@timezonehour}%
  \cslet{@dtm@#1@TZminute}{\@dtm@timezoneminute}%
}
```

`\DTMsavenow` Save the date-time stamp in PDF format.

```
\newrobustcmd*{\DTMsavepdftimestamp}[2]{%
  \edef\@dtm@tmp{#2}%
  \ifx\@dtm@tmp\empty
    \cslet{@dtm@#1@year}{0}%
    \cslet{@dtm@#1@month}{0}%
    \cslet{@dtm@#1@day}{0}%
    \cslet{@dtm@#1@dow}{0}%
    \cslet{@dtm@#1@hour}{0}%
    \cslet{@dtm@#1@minute}{0}%
    \cslet{@dtm@#1@second}{0}%
    \cslet{@dtm@#1@TZhour}{0}%
    \cslet{@dtm@#1@TZminute}{0}%
  \else
  \expandafter\@dtm@parsepdfdatetime\@dtm@tmp\@dtm@endparsepdfdatetime
    \cslet{@dtm@#1@year}{\@dtm@year}%
    \cslet{@dtm@#1@month}{\@dtm@month}%
    \cslet{@dtm@#1@day}{\@dtm@day}%
    \cslet{@dtm@#1@dow}{\@dtm@dow}%
    \cslet{@dtm@#1@hour}{\@dtm@hour}%
    \cslet{@dtm@#1@minute}{\@dtm@minute}%
    \cslet{@dtm@#1@second}{\@dtm@second}%
    \cslet{@dtm@#1@TZhour}{\@dtm@timezonehour}%
    \cslet{@dtm@#1@TZminute}{\@dtm@timezoneminute}%
  \fi
}
```

`\DTMsavenow` Save the current time.

```
\newrobustcmd{\DTMsavenow}[1]{%
  \cslet{@dtm@#1@year}{\@dtm@currentyear}%
  \cslet{@dtm@#1@month}{\@dtm@currentmonth}%
  \cslet{@dtm@#1@day}{\@dtm@currentday}%
  \cslet{@dtm@#1@dow}{\@dtm@currentdow}%
  \cslet{@dtm@#1@hour}{\@dtm@currenthour}%
  \cslet{@dtm@#1@minute}{\@dtm@currentminute}%
  \cslet{@dtm@#1@second}{\@dtm@currentsecond}%
  \cslet{@dtm@#1@TZhour}{\@dtm@currenttimezonehour}%
  \cslet{@dtm@#1@TZminute}{\@dtm@currenttimezoneminute}%
}
```

`\DTMmakeglobal` Globally set the stored information.

```
\newrobustcmd{\DTMmakeglobal}[1]{%
  \global\csletcs{@dtm@#1@year}{@dtm@#1@year}%
  \global\csletcs{@dtm@#1@month}{@dtm@#1@month}%
  \global\csletcs{@dtm@#1@day}{@dtm@#1@day}%
  \global\csletcs{@dtm@#1@dow}{@dtm@#1@dow}%
  \global\csletcs{@dtm@#1@hour}{@dtm@#1@hour}%
  \global\csletcs{@dtm@#1@minute}{@dtm@#1@minute}%
  \global\csletcs{@dtm@#1@second}{@dtm@#1@second}%
  \global\csletcs{@dtm@#1@TZhour}{@dtm@#1@TZhour}%
  \global\csletcs{@dtm@#1@TZminute}{@dtm@#1@TZminute}%
}
```

Expandable ways of fetching saved data. (No check for existence performed.) The argument is the label.

`\DTMfetchyear`

```
\newcommand*{\DTMfetchyear}[1]{\csname @dtm@#1@year\endcsname}
```

`\DTMfetchmonth`

```
\newcommand*{\DTMfetchmonth}[1]{\csname @dtm@#1@month\endcsname}
```

\DTMfetchday

```
\newcommand*{\DTMfetchday}[1]{\csname @dtm@#1@day\endcsname}
```

\DTMfetchdow

```
\newcommand*{\DTMfetchdow}[1]{\csname @dtm@#1@dow\endcsname}
```

\DTMfetchhour

```
\newcommand*{\DTMfetchhour}[1]{\csname @dtm@#1@hour\endcsname}
```

\DTMfetchminute

```
\newcommand*{\DTMfetchminute}[1]{\csname @dtm@#1@minute\endcsname}
```

\DTMfetchsecond

```
\newcommand*{\DTMfetchsecond}[1]{\csname @dtm@#1@second\endcsname}
```

\DTMfetchTZhour

```
\newcommand*{\DTMfetchTZhour}[1]{\csname @dtm@#1@TZhour\endcsname}
```

TMfetchTZminute

```
\newcommand*{\DTMfetchTZminute}[1]{\csname @dtm@#1@TZminute\endcsname}
```

\DTMusedate

`\DTMusedate{⟨label⟩}`

Displays the previously saved date using `\DTMdisplaydate`.

```
\newcommand*\DTMusedate[1]{%
  \ifcsundef{@dtm@#1@year}%
  {%
     \PackageError{datetime2}{Undefined date ‘#1’}{}%
  }%
  {%
     \DTMdisplaydate
      {\csname @dtm@#1@year\endcsname}%
      {\csname @dtm@#1@month\endcsname}%
      {\csname @dtm@#1@day\endcsname}%
      {\csname @dtm@#1@dow\endcsname}%
  }%
}%
```

\DTMUsedate

`\DTMUsedate{⟨label⟩}`

Displays the previously saved date using `\DTMDisplaydate`.

```
\newcommand*\DTMUsedate[1]{%
  \ifcsundef{@dtm@#1@year}%
  {%
     \PackageError{datetime2}{Undefined date ‘#1’}{}%
  }%
  {%
     \DTMDisplaydate
      {\csname @dtm@#1@year\endcsname}%
      {\csname @dtm@#1@month\endcsname}%
      {\csname @dtm@#1@day\endcsname}%
      {\csname @dtm@#1@dow\endcsname}%
  }%
}%
```

`\DTMusetime`

`\DTMusetime{⟨label⟩}`

Displays the previously saved time using `\DTMdisplaytime`.

```
\newcommand*\DTMusetime[1]{%
  \ifcsundef{@dtm@#1@hour}%
  {%
     \PackageError{datetime2}{Undefined time ‘#1’}{}%
  }%
  {%
     \DTMdisplaytime
      {\csname @dtm@#1@hour\endcsname}%
      {\csname @dtm@#1@minute\endcsname}%
      {\csname @dtm@#1@second\endcsname}%
  }%
}%
```

`\DTMusezone`

`\DTMusezone{⟨label⟩}`

Displays the previously saved date using `\DTMdisplayzone`.

```
\newcommand*\DTMusezone[1]{%
  \ifcsundef{@dtm@#1@TZhour}%
  {%
     \PackageError{datetime2}{Undefined time ‘#1’}{}%
  }%
  {%
     \DTMdisplayzone
      {\csname @dtm@#1@TZhour\endcsname}%
      {\csname @dtm@#1@TZminute\endcsname}%
  }%
}%
```

\DTMuse

`\DTMuse{⟨label⟩}`

Displays the previously saved date and time.

```
\newcommand*\DTMuse[1]{%
  \ifcsundef{@dtm@#1@year}%
  {%
     \PackageError{datetime2}{Undefined date-time ‘#1’}{}%
  }%
  {%
     \DTMdisplay
      {\csname @dtm@#1@year\endcsname}%
      {\csname @dtm@#1@month\endcsname}%
      {\csname @dtm@#1@day\endcsname}%
      {\csname @dtm@#1@dow\endcsname}%
      {\csname @dtm@#1@hour\endcsname}%
      {\csname @dtm@#1@minute\endcsname}%
      {\csname @dtm@#1@second\endcsname}%
      {\csname @dtm@#1@TZhour\endcsname}%
      {\csname @dtm@#1@TZminute\endcsname}%
  }%
}%
```

\DTMUse

`\DTMUse{⟨label⟩}`

Displays the previously saved date and time.

```
\newcommand*\DTMUse[1]{%
  \ifcsundef{@dtm@#1@year}%
  {%
     \PackageError{datetime2}{Undefined date-time ‘#1’}{}%
  }%
  {%
     \DTMDisplay
      {\csname @dtm@#1@year\endcsname}%
      {\csname @dtm@#1@month\endcsname}%
      {\csname @dtm@#1@day\endcsname}%
      {\csname @dtm@#1@dow\endcsname}%
      {\csname @dtm@#1@hour\endcsname}%
      {\csname @dtm@#1@minute\endcsname}%
      {\csname @dtm@#1@second\endcsname}%
      {\csname @dtm@#1@TZhour\endcsname}%
      {\csname @dtm@#1@TZminute\endcsname}%
  }%
}%
```

\DTMifsaveddate Determine if the given label has been assigned to a date, time and zone.

```
\newcommand{\DTMifsaveddate}[3]{%
  \ifcsundef{@dtm@#1@year}{#3}{#2}%
}
```

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.

```
\newcommand*{\@dtm@requiremodule}[1]{%
  \IfTrackedLanguageFileExists{#1}%
  {datetime2-}% prefix
  {.ldf}% suffix
  {%
    \RequireDateTimeModule{\CurrentTrackedTag}%
  }%
  {%
    \@dtm@warning{Date-Time Language Module '#1' not installed}%
  }%
}
```

m@loadedregions List of loaded datetime2 language modules.

```
\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.

```
\newcommand*{\RequireDateTimeModule}[1]{%
 \ifundef\CurrentTrackedDialect
 {%
   \PackageError{datetime2}%
   {\string\RequireDateTimeModule\space not permitted here}%
   {This command is only permitted inside datetime2 language
    modules.}%
 }%
 {%
   \ifcsundef{ver@datetime2-#1.ldf}%
   {%
     \input{datetime2-#1.ldf}%
     \ifdefempty\@dtm@loadedregions
     {%
```

```
       \edef\@dtm@loadedregions{#1}%
     }%
     {%
       \edef\@dtm@loadedregions{\@dtm@loadedregions,#1}%
     }%
```

In case a synonym is also used, add a mapping from the module name to the current tracked dialect.

```
     \csedef{@dtm@moddialectmap@#1}{\CurrentTrackedDialect}%
   }%
   {%
```

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.

```
     \ifcsdef{date\CurrentTrackedDialect}
     {%
       \letcs{\@dtm@otherdialect}{@dtm@moddialectmap@#1}%
       \edef\@dtm@thisdialect{\CurrentTrackedDialect}%
       \ifdefequal\@dtm@thisdialect\@dtm@otherdialect
       {}%
       {%
         \ifcsdef{date\@dtm@otherdialect}%
         {%
           \csletcs{date\@dtm@thisdialect}{date\@dtm@otherdialect}%
         }%
         {}%
       }%
     }%
     {}%
   }%
```

In case it's needed, create a mapping between the dialect name and the module name.

```
   \csedef{@dtm@dialectmodmap@\CurrentTrackedDialect}{#1}%
 }%
}
```

lecttomodulemap

`\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.

```
\newcommand*{\DTMdialecttomodulemap}[1]{%
 \ifcsdef{ver@datetime2-#1.ldf}%
 {#1}%
 {\csname @dtm@dialectmodmap@#1\endcsname}%
}
```

`sDateTimeModule` For use in language module to identify itself.

```
\newcommand*{\ProvidesDateTimeModule}[1]{%
  \ProvidesFile{datetime2-#1.ldf}%
}
```

`\DTMusemodule` Provided for packages or documents that need to load a module. This shouldn't be used inside the `.ldf` files.

```
\newcommand*{\DTMusemodule}[2]{%
 \ifcsdef{@tracklang@add@#1}%
 {%
   \TrackPredefinedDialect{#1}%
 }%
 {}%
 \let\@dtm@org@dialect\CurrentTrackedDialect
 \def\CurrentTrackedDialect{#1}%
 \RequireDateTimeModule{#2}%
 \let\CurrentTrackedDialect\@dtm@org@dialect
}
```

`\DTMdefkey`

```
\DTMdefkey{⟨region⟩}{⟨key⟩}[⟨default⟩]{⟨func⟩}
```

Used by language modules to define a key.

```
\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.

```
\newcommand*{\DTMdefchoicekey}[1]{\define@choicekey[dtm]{#1}}
```

`\DTMdefboolkey`

```
\DTMdefboolkey{⟨region⟩}[⟨mp⟩]{⟨key⟩}[⟨default⟩]{⟨func⟩}
```

Used by language modules to define a boolean key.

```
\newcommand*{\DTMdefboolkey}[1]{\define@boolkey[dtm]{#1}}
```

`\DTMifbool`

`\DTMifbool{⟨region⟩}{⟨key⟩}{⟨true⟩}{⟨false⟩}`

Test boolean key that was defined using `\DTMdefboolkey`

```
\newcommand*{\DTMifbool}[4]{\ifbool{dtm@#1@#2}{#3}{#4}}
```

`\DTMsetbool`

`\DTMsetbool{⟨region⟩}{⟨key⟩}{⟨value⟩}`

Set boolean key that was defined using `\DTMdefboolkey`

```
\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.

```
\newcommand*{\DTMlangsetup}{%
  \@ifstar\s@DTMlangsetup\@DTMlangsetup}
```

Unstarred version:

```
\newcommand*{\@DTMlangsetup}[2][\@dtm@loadedregions]{%
 \@for\@dtm@region:=#1\do{%
   \setkeys*+[dtm]{\@dtm@region}{#2}%
   \ifdefempty\XKV@rm{}%
   {%
     \@dtm@warning{Region `\@dtm@region' has ignored
      \MessageBreak the following settings:\MessageBreak
      \XKV@rm
      ^^J}%
   }%
 }%
}
```

Starred version:

```
\newcommand*{\s@DTMlangsetup}[2][\@dtm@loadedregions]{%
 \@for\@dtm@region:=#1\do{%
   \setkeys*+[dtm]{\@dtm@region}{#2}%
 }%
}
```

Now load all the required modules (if installed) using the tracklang interface. (Language packages, such as babel or polyglossia must be loaded before this.)

```
\AnyTrackedLanguages
{%
  \ForEachTrackedDialect{\this@dialect}%
  {%
    \@dtm@requiremodule\this@dialect
  }%
}
{%
```

No tracked languages. The default is already set up, so nothing to do here.

```
}
```

Load datetime2-calc if required.

```
\@dtm@usecalc
```

Use the style package option, if set.

```
\ifdefempty\@dtm@initialstyle{}{\DTMsetstyle{\@dtm@initialstyle}}
```

11.2 datetime2-calc.sty code

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{datetime2-calc}[2021/03/21 v1.5.7 (NLCT)]
```

Load other required packages

```
\RequirePackage{pgfkeys}
\RequirePackage{pgfcalendar}
```

11.2.1 Conversions and Calculations

`\@dtm@julianday` Register for storing Julian day number.

```
\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.

```
\def\@dtm@parsedate#1-#2-#3\@dtm@endparsedate{%
  \pgfcalendardatetojulian{#1-#2-#3}{\@dtm@julianday}%
  \pgfcalendarjuliantodate{\@dtm@julianday}{\@dtm@year}{\@dtm@month}{\@dtm@day}%
  \pgfcalendarjuliantoweekday{\@dtm@julianday}{\count@}%
  \edef\@dtm@dow{\number\count@}%
}
```

Set the current day of week

`@dtm@currentdow`

```
\pgfcalendardatetojulian
 {\@dtm@currentyear-\@dtm@currentmonth-\@dtm@currentday}%
 {\@dtm@julianday}%
\pgfcalendarjuliantoweekday{\@dtm@julianday}{\count@}%
\edef\@dtm@currentdow{\number\count@}%
```

`TMsavejulianday` Save the date obtained from the Julian day number.

```
\newrobustcmd*{\DTMsavejulianday}[2]{%
  \pgfcalendarjuliantodate{#2}{\@dtm@year}{\@dtm@month}{\@dtm@day}%
  \pgfcalendarjuliantoweekday{#2}{\count@}%
  \csedef{@dtm@#1@dow}{\number\count@}%
  \cslet{@dtm@#1@year}{\@dtm@year}%
  \cslet{@dtm@#1@month}{\@dtm@month}%
  \cslet{@dtm@#1@day}{\@dtm@day}%
  \ifcsundef{@dtm@#1@hour}{\csdef{@dtm@#1@hour}{0}}{}%
  \ifcsundef{@dtm@#1@minute}{\csdef{@dtm@#1@minute}{0}}{}%
  \ifcsundef{@dtm@#1@second}{\csdef{@dtm@#1@second}{0}}{}%
  \ifcsundef{@dtm@#1@TZhour}{\csdef{@dtm@#1@TZhour}{0}}{}%
  \ifcsundef{@dtm@#1@TZminute}{\csdef{@dtm@#1@TZminute}{0}}{}%
}
```

`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.

```
\newrobustcmd*{\DTMsaveddatetojulianday}[2]{%
  \ifcsundef{@dtm@#1@year}%
  {%
     \PackageError{datetime2-calc}{Unknown date ‘#1’}{}%
  }%
  {%
    \pgfcalendardatetojulian
     {\csname @dtm@#1@year\endcsname
     -\csname @dtm@#1@month\endcsname
     -\csname @dtm@#1@day\endcsname}%
     {#2}%
  }%
}
```

`fsettojulianday` 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.

```
\newrobustcmd*{\DTMsaveddateoffsettojulianday}[3]{%
  \ifcsundef{@dtm@#1@year}%
  {%
     \PackageError{datetime2-calc}{Unknown date ‘#1’}{}%
  }%
  {%
    \pgfcalendardatetojulian
     {\csname @dtm@#1@year\endcsname
     -\csname @dtm@#1@month\endcsname
     -\csname @dtm@#1@day\endcsname
     +#2}%
     {#3}%
  }%
}
```

`\DTMifdate` Test a saved date using `\pgfcalendarifdate`

```
\newrobustcmd*{\DTMifdate}[4]{%
  \ifcsundef{@dtm@#1@year}%
  {%
     \PackageError{datetime2-calc}{Unknown date ‘#1’}{}%
  }%
  {%
    \pgfcalendarifdate
     {\csname @dtm@#1@year\endcsname
     -\csname @dtm@#1@month\endcsname
     -\csname @dtm@#1@day\endcsname}%
     {#2}{#3}{#4}%
  }%
}
```

TMsaveddatediff Computes the difference between two saved dates. The result is stored in the third argument, which should be a count register.

```
\newrobustcmd*{\DTMsaveddatediff}[3]{%
  \ifcsundef{@dtm@#1@year}%
  {%
     \PackageError{datetime2-calc}{Unknown date ‘#1’}{}%
  }%
  {%
     \ifcsundef{@dtm@#2@year}%
     {%
        \PackageError{datetime2-calc}{Unknown date ‘#1’}{}%
     }%
     {%
       \pgfcalendardatetojulian
        {\csname @dtm@#1@year\endcsname
        -\csname @dtm@#1@month\endcsname
        -\csname @dtm@#1@day\endcsname}%
        {#3}%
       \pgfcalendardatetojulian
        {\csname @dtm@#2@year\endcsname
        -\csname @dtm@#2@month\endcsname
        -\csname @dtm@#2@day\endcsname}%
        {\@dtm@julianday}%
       \advance#3 by -\@dtm@julianday\relax
    }%
  }%
}
```

\DTMtozulu Converts the datetime data referenced by the first argument into Zulu time and saves it to data referenced by the second argument.

```
\newrobustcmd*{\DTMtozulu}[2]{%
  \ifcsundef{@dtm@#1@year}%
  {%
     \PackageError{datetime2-calc}{Unknown date ‘#1’}{}%
  }%
  {%
    \DTMsaveaszulutime{#2}%
    {\DTMfetchyear{#1}}%
    {\DTMfetchmonth{#1}}%
    {\DTMfetchday{#1}}%
    {\DTMfetchhour{#1}}%
    {\DTMfetchminute{#1}}%
    {\DTMfetchsecond{#1}}%
    {\DTMfetchTZhour{#1}}%
    {\DTMfetchTZminute{#1}}%
  }%
}
```

`Msaveaszulutime` Converts the given datetime into Zulu (+00:00) and saves the result.

```
\DTMsavetozulutime{⟨name⟩}{⟨year⟩}{⟨month⟩}{⟨day⟩}{⟨hour⟩}
{⟨minute⟩}{⟨second⟩}{⟨tzh⟩}{⟨tzm⟩}
```

```
\newrobustcmd*{\DTMsaveaszulutime}[9]{%
  \edef\@dtm@year{\number#2}%
  \edef\@dtm@month{\number#3}%
  \edef\@dtm@day{\number#4}%
  \edef\@dtm@hour{\number#5}%
  \edef\@dtm@minute{\number#6}%
  \edef\@dtm@second{\number#7}%
  \edef\@dtm@TZhour{\number#8}%
  \edef\@dtm@TZminute{\number#9}%
  \pgfcalendardatetojulian{\@dtm@year-\@dtm@month-\@dtm@day}{\@dtm@julianday}%
```

First adjust the minute offset if non-zero

```
  \ifnum\@dtm@TZminute=0\relax
  \else
    \count@=\@dtm@minute\relax
```

Add or subtract the offset minute

```
    \ifnum\@dtm@TZhour<0\relax
      \advance\count@ by \@dtm@TZminute\relax
    \else
      \advance\count@ by -\@dtm@TZminute\relax
    \fi
    \edef\@dtm@minute{\number\count@}%
```

Does the hour need adjusting?

```
    \ifnum\count@<0\relax
      \advance\count@ by 60\relax
      \edef\@dtm@minute{\number\count@}%
```

Need to subtract 1 from the hour but does the day need adjusting?

```
      \ifnum\@dtm@hour=0\relax
       \def\@dtm@hour{23}%
```

Day needs adjusting.

```
       \advance\@dtm@julianday by -1\relax
      \else
```

Subtract 1 from the hour

```
      \count@ = \@dtm@hour\relax
      \advance\count@ by -1\relax
      \edef\@dtm@hour{\number\count@}%
    \fi
  \else
```

Minute isn’t negative. Is it ≥ 60?

```
    \ifnum\count@>59\relax
      \advance\count@ by -60\relax
      \edef\@dtm@minute{\number\count@}%
```

Add 1 to the hour

```
      \count@ = \@dtm@hour\relax
      \advance\count@ by 1\relax
      \edef\@dtm@hour{\number\count@}%
```

Does the day need adjusting?

```
      \ifnum\@dtm@hour=24\relax
        \def\@dtm@hour{00}%
        \advance\@dtm@julianday by 1\relax
      \fi
    \fi
  \fi
\fi
```

Now adjust the hour offset if non-zero

```
\ifnum\@dtm@TZhour=0\relax
\else
  \count@=\@dtm@hour\relax
  \advance\count@ by -\@dtm@TZhour\relax
```

Does the day need adjusting?

```
  \ifnum\count@<0\relax
    \advance\count@ by 24\relax
    \edef\@dtm@hour{\number\count@}%
    \advance\@dtm@julianday by -1\relax
  \else
    \ifnum\count@>23\relax
    \advance\count@ by -24\relax
    \edef\@dtm@hour{\number\count@}%
    \advance\@dtm@julianday by 1\relax
    \else
      \edef\@dtm@hour{\number\count@}%
    \fi
  \fi
\fi
\pgfcalendarjuliantodate{\@dtm@julianday}{\@dtm@year}{\@dtm@month}{\@dtm@day}%
\pgfcalendarjuliantoweekday{\@dtm@julianday}{\count@}%
```

Save the results.

```
\csedef{@dtm@#1@dow}{\number\count@}%
```

```
  \cslet{@dtm@#1@year}{\@dtm@year}%
  \cslet{@dtm@#1@month}{\@dtm@month}%
  \cslet{@dtm@#1@day}{\@dtm@day}%
  \cslet{@dtm@#1@hour}{\@dtm@hour}%
  \cslet{@dtm@#1@minute}{\@dtm@minute}%
  \cslet{@dtm@#1@second}{\@dtm@second}%
  \csdef{@dtm@#1@TZhour}{0}%
  \csdef{@dtm@#1@TZminute}{0}%
}
```

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`

```
\newcommand*{\dtm@ifdianameexists}[3]{%
 \IfTrackedDialect{\languagename}%
 {%
   \ifcsdef{DTM\TrackedLanguageFromDialect{\languagename}#1}%
   {#2}%
   {#3}%
 }%
 {#3}%
}
```

`\DTMmonthname`

```
\newrobustcmd{\DTMmonthname}[1]{%
```

First check if `\DTM`⟨*language*⟩`monthname` exists.

```
  \ifcsdef{DTM\languagename monthname}%
  {%
```

It exists, so use it.

```
    \csuse{DTM\languagename monthname}{#1}%
  }%
  {%
```

Try obtaining the language name from the dialect using tracklang's interface.

```
    \dtm@ifdianameexists{monthname}%
    {%
```

It exists, so use it.

```
       \csuse{DTM\TrackedLanguageFromDialect{\languagename}monthname}{#1}%
    }%
    {%
```

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.

```
      \dtmnamewarning{\DTMmonthname}%
      \pgfcalendarmonthname{#1}%
    }%
  }%
}
```

\DTMMonthname

```
\newrobustcmd{\DTMMonthname}[1]{%
```

First check if \DTM⟨*language*⟩Monthname exists.

```
  \ifcsdef{DTM\languagename Monthname}%
  {%
```

It exists, so use it.

```
    \csuse{DTM\languagename Monthname}{#1}%
  }%
  {%
```

Try obtaining the language name from the dialect using tracklang's interface.

```
    \dtm@ifdianameexists{Monthname}%
    {%
```

It exists, so use it.

```
       \csuse{DTM\TrackedLanguageFromDialect{\languagename}Monthname}{#1}%
    }%
    {%
```

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).

```
      \ifcsdef{DTM\languagename monthname}%
      {%
        \csuse{DTM\languagename monthname}{#1}%
      }%
      {%
        \dtm@ifdianameexists{monthname}%
        {%
          \csuse{DTM\TrackedLanguageFromDialect{\languagename}monthname}{#1}%
        }%
        {%
```

Can't find no-case change version either, so use pgfcalendar command instead (which will need a case-change applied).

```
          \dtmnamewarning{\DTMMonthname}%
```

If mfirstuc has been loaded, use it.

```
          \ifdef\emakefirstuc
          {%
           \emakefirstuc{\pgfcalendarmonthname{#1}}%
          }%
          {%
```

Hasn't been loaded, so just expand and apply `\MakeUppercase`:

```
            \protected@edef\dtm@tmp@monthname{\pgfcalendarmonthname{#1}}%
            \expandafter\MakeUppercase\dtm@tmp@monthname
          }%
        }%
      }%
    }%
  }%
}
```

Mshortmonthname

```
\newrobustcmd{\DTMshortmonthname}[1]{%
```

First check if `\DTM`⟨*language*⟩`shortmonthname` exists.

```
  \ifcsdef{DTM\languagename shortmonthname}%
  {%
```

It exists, so use it.

```
    \csuse{DTM\languagename shortmonthname}{#1}%
  }%
  {%
```

Try obtaining the language name from the dialect using tracklang's interface.

```
    \dtm@ifdianameexists{shortmonthname}%
    {%
```

It exists, so use it.

```
      \csuse{DTM\TrackedLanguageFromDialect{\languagename}shortmonthname}{#1}%
    }%
    {%
```

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.

```
      \dtmnamewarning{\DTMshortmonthname}%
      \pgfcalendarmonthshortname{#1}%
    }%
  }%
}
```

MshortMonthname

```
\newrobustcmd{\DTMshortMonthname}[1]{%
```

First check if `\DTM`⟨*language*⟩`shortMonthname` exists.

```
  \ifcsdef{DTM\languagename shortMonthname}%
  {%
```

It exists, so use it.

```
    \csuse{DTM\languagename shortMonthname}{#1}%
  }%
  {%
```

Try obtaining the language name from the dialect using tracklang's interface.

```
    \dtm@ifdianameexists{shortMonthname}%
    {%
```

It exists, so use it.

```
       \csuse{DTM\TrackedLanguageFromDialect{\languagename}shortMonthname}{#1}%
    }%
    {%
```

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).

```
      \ifcsdef{DTM\languagename shortmonthname}%
      {%
        \csuse{DTM\languagename shortmonthname}{#1}%
      }%
      {%
        \dtm@ifdianameexists{shortmonthname}%
        {%
           \csuse{DTM\TrackedLanguageFromDialect{\languagename}shortmonthname}%
           {#1}%
        }%
        {%
```

Can't find no-case change version either, so use pgfcalendar command instead (which will need a case-change applied).

```
          \dtmnamewarning{\DTMshortMonthname}%
```

If mfirstuc has been loaded, use it.

```
          \ifdef\emakefirstuc
          {%
           \emakefirstuc{\pgfcalendarmonthshortname{#1}}%
          }%
          {%
```

Hasn't been loaded, so just expand and apply `\MakeUppercase`:

```
            \protected@edef\dtm@tmp@monthname{\pgfcalendarmonthshortname{#1}}%
            \expandafter\MakeUppercase\dtm@tmp@monthname
          }%
        }%
      }%
    }%
  }%
}
```

edayofweekindex

`\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.

```
\newrobustcmd*{\DTMcomputedayofweekindex}[2]{%
  \pgfcalendardatetojulian{#1}{\@dtm@julianday}%
  \pgfcalendarjuliantoweekday{\@dtm@julianday}{\count@}%
  \edef#2{\number\count@}%
}
```

These are shared with the locale package, so they may already be defined.

`\dtmMondayIndex`

```
\def\dtmMondayIndex{0}
```

`dtmTuesdayIndex`

```
\def\dtmTuesdayIndex{1}
```

`mWednesdayIndex`

```
\def\dtmWednesdayIndex{2}
```

`tmThursdayIndex`

```
\def\dtmThursdayIndex{3}
```

`\dtmFridayIndex`

```
\def\dtmFridayIndex{4}
```

`tmSaturdayIndex`

```
\def\dtmSaturdayIndex{5}
```

`\dtmSundayIndex`

```
\def\dtmSundayIndex{6}
```

`\DTMweekdayname`

```
\newrobustcmd{\DTMweekdayname}[1]{%
```

First check if `\DTM⟨language⟩weekdayname` exists.

```
  \ifcsdef{DTM\languagename weekdayname}%
  {%
```

It exists, so use it.

```
    \csuse{DTM\languagename weekdayname}{#1}%
  }%
  {%
```

Try obtaining the language name from the dialect using tracklang's interface.

```
    \dtm@ifdianameexists{weekdayname}%
    {%
```

It exists, so use it.

```
       \csuse{DTM\TrackedLanguageFromDialect{\languagename}weekdayname}{#1}%
    }%
    {%
```

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.

```
      \dtmnamewarning{\DTMweekdayname}%
      \pgfcalendarweekdayname{#1}%
    }%
  }%
}
```

\DTMWeekdayname

```
\newrobustcmd{\DTMWeekdayname}[1]{%
```

First check if `\DTM`⟨*language*⟩`Weekdayname` exists.

```
  \ifcsdef{DTM\languagename Weekdayname}%
  {%
```

It exists, so use it.

```
    \csuse{DTM\languagename Weekdayname}{#1}%
  }%
  {%
```

Try obtaining the language name from the dialect using tracklang's interface.

```
    \dtm@ifdianameexists{Weekdayname}%
    {%
```

It exists, so use it.

```
       \csuse{DTM\TrackedLanguageFromDialect{\languagename}Weekdayname}{#1}%
    }%
    {%
```

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).

```
      \ifcsdef{DTM\languagename weekdayname}%
      {%
        \csuse{DTM\languagename weekdayname}{#1}%
      }%
      {%
        \dtm@ifdianameexists{weekdayname}%
        {%
           \csuse{DTM\TrackedLanguageFromDialect{\languagename}weekdayname}{#1}%
        }%
        {%
```

Can't find no-case change version either, so use pgfcalendar command instead (which will need a case-change applied).

```
          \dtmnamewarning{\DTMWeekdayname}%
```

If mfirstuc has been loaded, use it.

```
          \ifdef\emakefirstuc
          {%
           \emakefirstuc{\pgfcalendarweekdayname{#1}}%
          }%
          {%
```

Hasn't been loaded, so just expand and apply `\MakeUppercase`:

```
            \protected@edef\dtm@tmp@weekdayname{\pgfcalendarweekdayname{#1}}%
            \expandafter\MakeUppercase\dtm@tmp@weekdayname
          }%
        }%
      }%
    }%
  }%
}
```

hortweekdayname

```
\newrobustcmd{\DTMshortweekdayname}[1]{%
```

First check if `\DTM`⟨*language*⟩`shortweekdayname` exists.

```
  \ifcsdef{DTM\languagename shortweekdayname}%
  {%
```

It exists, so use it.

```
    \csuse{DTM\languagename shortweekdayname}{#1}%
  }%
  {%
```

Try obtaining the language name from the dialect using tracklang's interface.

```
    \dtm@ifdianameexists{shortweekdayname}%
    {%
```

It exists, so use it.

```
       \csuse{DTM\TrackedLanguageFromDialect{\languagename}shortweekdayname}{#1}%
    }%
    {%
```

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.

```
      \dtmnamewarning{\DTMshortweekdayname}%
      \pgfcalendarweekdayshortname{#1}%
    }%
  }%
}
```

hortWeekdayname

```
\newrobustcmd{\DTMshortWeekdayname}[1]{%
```

First check if `\DTM`⟨*language*⟩`shortWeekdayname` exists.

```
\ifcsdef{DTM\languagename shortWeekdayname}%
{%
```

It exists, so use it.

```
  \csuse{DTM\languagename shortWeekdayname}{#1}%
}%
{%
```

Try obtaining the language name from the dialect using tracklang's interface.

```
  \dtm@ifdianameexists{shortWeekdayname}%
  {%
```

It exists, so use it.

```
     \csuse{DTM\TrackedLanguageFromDialect{\languagename}shortWeekdayname}{#1}%
  }%
  {%
```

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).

```
    \ifcsdef{DTM\languagename shortweekdayname}%
    {%
      \csuse{DTM\languagename shortweekdayname}{#1}%
    }%
    {%
      \dtm@ifdianameexists{shortweekdayname}%
      {%
         \csuse{DTM\TrackedLanguageFromDialect{\languagename}shortweekdayname}%
         {#1}%
      }%
      {%
```

Can't find no-case change version either, so use pgfcalendar command instead (which will need a case-change applied).

```
        \dtmnamewarning{\DTMshortWeekdayname}%
```

If mfirstuc has been loaded, use it.

```
       \ifdef\emakefirstuc
       {%
         \emakefirstuc{\pgfcalendarweekdayshortname{#1}}%
       }%
       {%
```

Hasn't been loaded, so just expand and apply `\MakeUppercase`:

```
          \protected@edef\dtm@tmp@weekdayname{%
            \pgfcalendarweekdayshortname{#1}}%
          \expandafter\MakeUppercase\dtm@tmp@weekdayname
        }%
      }%
    }%
  }%
}%
```

```
}
```

\DTMordinal

```
\newrobustcmd{\DTMordinal}[1]{%
```

First check if `\DTM⟨language⟩ordinal` exists.

```
  \ifcsdef{DTM\languagename ordinal}%
  {%
```

It exists, so use it.

```
    \csuse{DTM\languagename ordinal}{#1}%
  }%
  {%
```

Try obtaining the language name from the dialect using tracklang's interface.

```
    \dtm@ifdianameexists{ordinal}%
    {%
```

It exists, so use it.

```
      \csuse{DTM\TrackedLanguageFromDialect{\languagename}ordinal}{#1}%
    }%
    {%
```

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.

```
      \number#1
    }%
  }%
}
```

\dtmnamewarning Issue warning unless warnings have been suppressed.

```
\newcommand*{\dtmnamewarning}[1]{%
 \if@dtm@warn
  \PackageWarning{datetime2-calc}%
  {Can't find underlying language macro for \MessageBreak
   \string#1\space(language: \languagename); \MessageBreak
    using pgfcalendar macro instead}%
 \fi
}
```

Change History

- 1.0 (2015-03-24)
  - General: Initial release 109
- 1.03 (2016-01-20)
  - `\DTMusemodule`: new 159
- 1.1 (2015-09-15)
  - `\DTMsaveaszulutime`: fixed bug in incorrect conversion 164
    - fixed bug in misnamed `\@dtm@hour` . 165
- 1.2 (2015-11-10)
  - `\DTMprovidedatestyle`: new 127
  - `\DTMprovidestyle`: new 133
  - `\DTMprovidetimestyle`: new 128
  - `\DTMprovidezonestyle`: new 129
  - `\DTMrenewdatestyle`: new 127
  - `\DTMrenewstyle`: new 132
  - `\DTMrenewtimestyle`: new 128
  - `\DTMrenewzonestyle`: new 129
- 1.3 (2016-01-22)
  - General: added ddmmyy style 144
    - added mmddyy style 149
    - added predefined dialect test 114
  - `\dtm@ifdianameexists`: new 166
  - `\DTMdialecttomodulemap`: new 158
  - `\DTMlangsetup`: added starred version 160
  - `\DTMMonthname`: new 167
  - `\DTMmonthname`: new 166
  - `\dtmnamewarning`: new 174
  - `\DTMordinal`: new 174
  - `\DTMshortMonthname`: new 168
  - `\DTMshortmonthname`: new 168
  - `\DTMshortWeekdayname`: new 172
  - `\DTMshortweekdayname`: new 172
  - `\DTMWeekdayname`: new 171
  - `\DTMweekdayname`: new 170
  - `\RequireDateTimeModule`: added dialect to module map 158
    - added module to dialect map 158
- 1.4 (2016-02-11)
  - `\DTMcomputedayofweekindex`: new . . 170
  - `\DTMToday`: new 122
  - `\DTMtoday`: new 121
  - `\today`: no longer using `\renewcommand` 121
- 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 131
  - `\DTMifhasstyle`: new 131
  - `\DTMifhastimestyle`: new 131
  - `\DTMifhaszonestyle`: new 131
  - `\DTMsavefrompdfdata`: new 116
  - `\DTMsetregional`: new 135
  - `\DTMtryregional`: new 134
- 1.5.3 (2018-07-20)
  - General: added space after test 137–142, 144–150
  - `\dtmFridayIndex`: new 170
  - `\dtmMondayIndex`: new 170
  - `\dtmSaturdayIndex`: new 170
  - `\DTMsavenow`: new 152
  - `\dtmSundayIndex`: new 170
  - `\dtmThursdayIndex`: new 170
  - `\dtmTuesdayIndex`: new 170
  - `\dtmWednesdayIndex`: new 170
- 1.5.4 (2019-10-21)
  - `useregional`: renamed scratch variables 112
- 1.5.5 (2019-11-11)
  - `\@DTMdate`: new 122
  - `\@sDTMDate`: new 123
  - `\@sDTMdate`: new 123
  - General: add check for `\TrackIfKnownLanguage` 114
  - `\DTMDate`: added starred version 123
  - `\DTMdate`: added starred version 122
  - `\DTMfinaldot`: new 122

1.5.6 (2020-03-02)
    `\DTMsetregional`: removed spurious space 135

1.5.7 (2021-03-21)
    `\DTMsaveddatediff`: removed spurious space 163

Index