tex-locale v1.0: setup document locale

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

2018-08-26

Abstract

The generic TEX `tex-locale.tex` code uses both tracklang (at least v1.3.4) and texosquery (at least v1.2, but newest version recommended) to look up the locale information from the operating system and provide commands that can access locale-dependent information, such as the currency symbol and numeric separators. This works best with the shell escape when building the document. (Some adjustments are needed to work without the shell escape.) TEX Live 2017 now includes `texosquery-jre8` on the restricted list, but the texosquery installation needs to be set up to enable this. Set up instructions are in the file `texosquery.cfg`, which is distributed with the texosquery package. If the application isn't on the restricted list, then you'll need to enable the unrestricted mode, but take care as this mode is insecure.

Example plain XETEX document:

```
\font\nimbus="NimbusRoman-Regular" at 10pt
\nimbus

\input locale

Currency: \CurrentLocaleCurrency.
Date: \CurrentLocaleDate.
Time: \CurrentLocaleTime.
\bye
```

The LATEX package `tex-locale.sty` can additionally load babel or polyglossia with the locale's language setting, as well as various other packages such as fontspec (XELATEX or LUALATEX) or fontenc and inputenc. Packages that provide currency symbols can also be loaded automatically (textcomp by default). Example LATEX document:

```
\documentclass{article}

\usepackage{tex-locale}

\begin{document}
Currency: \CurrentLocaleCurrency.
Date: \CurrentLocaleDate.
Time: \CurrentLocaleTime.
\end{document}
```

TEX's restricted mode prohibits the use of quotes in the shell escape for security reasons. This means that if you want the file modification date and you have spaces in your filename you must use the unrestricted mode to allow the filename to be delimited by quotes. In general it's best to avoid spaces in file names.

Contents

1 Introduction **5**
- 1.1 Using the package without the shell escape . . . 6
- 1.2 Encoding . . . 8

2 Generic Use **11**
- 2.1 General Information . . . 14
- 2.2 Attributes . . . 16
  - 2.2.1 Attribute Lists . . . 18
  - 2.2.2 General Attributes . . . 20
  - 2.2.3 Dialect Attributes . . . 22
    - 2.2.3.1 General Dialect Attributes . . . 22
    - 2.2.3.2 Dates and Times Dialect Attributes . . . 23
    - 2.2.3.3 Time Zones . . . 29
    - 2.2.3.4 Numeric . . . 29
  - 2.2.4 Region Attributes . . . 30
  - 2.2.5 Currency Attributes . . . 31
- 2.3 Patterns . . . 32
  - 2.3.1 Numeric Patterns . . . 32
  - 2.3.2 Date-Time Patterns . . . 38
- 2.4 Locale Information . . . 42
- 2.5 Dates and Times . . . 45
  - 2.5.1 Dates . . . 46
    - 2.5.1.1 Week Days . . . 47
    - 2.5.1.2 Month Names . . . 51
  - 2.5.2 Times . . . 52
- 2.6 Numbers . . . 54
  - 2.6.1 Numeric Symbols . . . 54
  - 2.6.2 Currencies . . . 55
- 2.7 Current Locale . . . 58
  - 2.7.1 Dates and Times . . . 60
  - 2.7.2 Numeric Symbols . . . 64
  - 2.7.3 Current Locale Patterns . . . 65

3 LaTeX Use **66**

4 The Code **70**
- 4.1 LaTeX Code (`tex-locale.sty`) . . . 70

4.2 Generic Code (tex-locale.tex) . . . 89
4.2.1 Setting Command Line Switches . . . 92
4.2.2 System Call . . . 94
4.2.3 Attributes . . . 95
4.2.4 Parser Commands . . . 113
4.2.5 Locale User Commands . . . 126
4.2.6 Patterns . . . 143
4.2.7 Conditionals . . . 146
4.2.8 Post-Parsing . . . 147
4.2.9 Initialising Current Locale Commands . . . 147
4.2.10 Switching Locale . . . 148
4.3 Scripts to fontenc mappings (tex-locale-scripts-enc.def) . . . 154
4.4 texosquery to inputenc mappings (tex-locale-encodings.def) . . . 156
4.5 Language Support Setup . . . 156

Main Index **164**

Code Index **168**

Change History **174**

1 Introduction

The tex-locale package is designed to set up the document's locale-sensitive information by querying the relevant information from the operating system using `texosquery`. Section 2 describes generic commands that can be used in LaTeX or plain TeX documents. Section 3 describes the LaTeX-specific package `tex-locale.sty`, which does more than simply input `tex-locale.tex`.

The generic code `tex-locale.tex` requires: `tracklang.tex` and `texosquery.tex`. The LaTeX package `tex-locale.sty` additionally requires: `tracklang.sty`, `texosquery.sty`, etoolbox, xfor, ifxetex, ifluatex, xkeyval and optionally:

- datetime2 (and associated language modules);
- textcomp or fontawesome;
- fontspec or fontenc&inputenc;
- polyglossia or babel;
- CJK, CJKutf8 or xeCJK;
- tracklang-scripts (provided with tracklang).

The texosquery package (distributed separately) comes with generic TeX code `texosquery.tex`, a simple LaTeX package wrapper `texosquery.sty` and a Java application that comes in three variants:

- `texosquery-jre8.jar`: the full application, requires at least Java 8.
- `texosquery.jar`: a slightly cut-down version of the application with less locale support, requires at least Java 7.
- `texosquery-jre5.jar`: a significantly reduced version of the application with poor locale support, requires at least Java 5. *Less secure than the other variants.*

The Java 8 variant (`texosquery-jre8`) is on TeX Live 2017's restricted list, so it's possible to use it with the restricted shell escape. The other variants should not be added to the restricted list as old versions of Java are deprecated and considered security risks. In particular, the Java 5 variant's file listing actions are less secure as they allow file listings outside of the current directory path, which the Java 7 and 8 variants prohibit. If you have Java 8 installed, I recommend that you make `texosquery-jre8` the default. This can be done by editing the configuration file `texosquery.cfg`. See the texosquery documentation for further details. (This document assumes `texosquery-jre8` in the examples. Substitute the appropriate command if you have a different set up.)

MiKTeX users will need to enable piped shell escape with `--enable-pipes` (unless the non-shell escape method described in Section 1.1 is used).

The information returned by `texosquery` has special markup that's converted by `\TeXOSQuery` or `\TeXOSQueryFromFile`. For example, `\fhyn` is used to represent a literal hyphen (category code 12) whereas `\thyn` is used for a textual hyphen. For example, if a file name containing a hyphen is returned, the hyphen will be marked as `\fhyn`, whereas a date containing a hyphen will use `\thyn`.

1.1 Using the package without the shell escape

The tex-locale package is designed for use with the piped shell escape (preferably the restricted mode for greater security), but it's still possible to use `tex-locale.tex` when the shell escape is disabled, although it's less convenient. (You will need at least texosquery v1.4 for this method.)

First compile your document with the shell escape disabled. The dry run mode will automatically be on, and the texosquery command `\TeXOSQuery` will simply write the command it would've tried to the transcript. This will be prefixed by `TeXOSQuery:`

For example, suppose the file `test.tex` contains:

```
\def\LocaleMain{en-GB}
\def\LocaleOther{de-CH-1996,fr-BE}

\input locale

Currency: \CurrentLocaleCurrency.

\bye
```

Here the main locale has been explicitly set to `en-GB` and the other locales have been set to `de-CH-1996` and `fr-BE`, so the transcript file `test.log` will include:

```
TeXOSQuery: texosquery-jre8 -o -r -a -n -N -C -d 'test.tex' -D en-GB
-D de-CH-1996 -D fr-BE
```

Copy and paste this command into a command prompt or terminal and redirect the output to a file. (Don't include the initial `TeXOSQuery:` and don't include any line breaks.) For example (omit line break):

```
texosquery-jre8 -o -r -a -n -N -C -d 'test.tex' -D en-GB
-D de-CH-1996 -D fr-BE > localesettings.tex
```

Then define

\LocaleQueryFile

```
\LocaleQueryFile
```

to the file name before loading tex-locale:

```
\def\LocaleQueryFile{localesettings}
\def\LocaleMain{en-GB}
\def\LocaleOther{de-CH-1996,fr-BE}

\input locale

Currency: \CurrentLocaleCurrency.

\bye
```

If you change any of the document settings you'll need to re-run `texosquery` to update the query result file.

If you want the current date or time, the result files will need to be updated before every document build.

For example, if the file `test.tex` is simply:

```
\input locale

Today: \CurrentLocaleShortDate.
Currency: \CurrentLocaleCurrency.

\bye
```

Then the document build process would be:

```
texosquery-jre8 -o -r -a -n -N -C -d 'test.tex' -D > testsettings.tex
etex '\def\LocaleQueryFile{testsettings}\input test'
```

(Replace `etex` with `pdftex` or `xetex`, as required.)

If `\LocaleQueryFile` is defined and non-empty, the non-shell escape method will automatically be implemented even if the shell escape is actually enabled.

If you're using LaTeX (`tex-locale.sty`), there may be an initial query with `-b` or `-C` (or both) before `tex-locale.tex` is input. The result of this will also need to be captured in a file. For example:

```
texosquery-jre8 -b -C > localestysettings.tex
```

This file name should be provided in the command

`\LocaleStyQueryFile`

`\LocaleStyQueryFile`

In this case you only need to update the file if you change the document locales, encoding or engine. (For example, if you change the value of the package options main, other, support, inputenc or fontenc.)

LaTeX example:

```
\documentclass{article}
\newcommand{\LocaleStyQueryFile}{localestysettings}
\newcommand{\LocaleQueryFile}{localesettings}
\usepackage[main=en-GB,other={de-CH-1996,fr-BE}]{tex-locale}
\begin{document}
Currency: \CurrentLocaleCurrency.
\end{document}
```

Even if the shell escape is on, if `\LocaleQueryFile` or `\LocaleStyQueryFile` have been set (to a non-empty value), they'll be used instead.

It's possible to define one but not the other command. For example:

```
\documentclass{article}
\newcommand{\LocaleStyQueryFile}{localestysettings}
\usepackage[main=en-GB,other={de-CH-1996,fr-BE}]{tex-locale}
\begin{document}
Currency: \CurrentLocaleCurrency.
\end{document}
```

This requires the shell escape for the main locale information (used in `tex-locale.tex`) but omits the LaTeX-specific shell escape used in `tex-locale.sty`.

1.2 Encoding

The default encoding used by `texosquery` when writing the results to STDOUT (which are piped in through the shell escape) can be obtained with `--codeset` (from v1.6). For example:

```
texosquery-jre8 --codeset
```

The short form of this switch is `-cs`. So the above is equivalent to

```
texosquery-jre8 -cs
```

For example, if the encoding is UTF-8, then this returns

```
UTF\fhyn 8
```

There's a similar switch `--codeset-lcs` (from v1.2) that converts the encoding name to lower case and strips hyphens. For example

```
texosquery-jre8 --codeset-lcs
```

The short form is `-C`

```
texosquery-jre8 -C
```

For example, if the encoding is UTF-8, then this returns

```
utf8
```

(The switches are case-sensitive. The lower case `-c` has a different meaning.)

The document encoding must match the encoding used by `texosquery` when the query is made (through `\TeXOSQuery` or `\TeXOSQueryFromFile`). This is why the LaTeX package `tex-locale.sty` makes an initial query with `-C` so that it can set up the appropriate input encoding when used with the inputenc=auto package option. If the document doesn't use this interface, then the encoding needs to be correctly set before the query is made.

You have a choice of either changing the document encoding to match the encoding used by `texosquery` or by changing `texosquery`'s encoding to match all your documents. For example, if your Java runtime environment is set up so that the default file encoding is ISO 8859-1 (Latin-1) but you always use UTF-8 in your source code, then you can instruct `texosquery` to use UTF-8 with the `--encoding` option (from v1.6). For example:

```
texosquery-jre8 --encoding UTF-8 -N
```

will ensure that the currency symbol is written as a UTF-8 character. The short form is `-enc`. For example:

```
texosquery-jre8 -enc UTF-8 -N
```

You can append this option to the definition of `\TeXOSInvokerName`. For example:

```
\def\TeXOSInvokerName{texosquery-jre8 -enc UTF-8}
```

There are two hooks which, if defined, are used immediately before or immediately after the main query is made in `tex-locale.tex`. The hook used before the query is

\localeprequery

```
\localeprequery
```

and the hook used after the query is

\localepostquery

```
\localepostquery
```

These may start and end a grouping, if required, as the result from the query is given a global assignment, so these may be used to temporarily switch the input encoding just for the query. This is done by `tex-locale.sty` if it detects that the document requires CJKutf8. In this case it defines the hooks to locally switch the encoding while the CJK characters are read from the query result.

In addition to using markup commands like `\fhyn` and `\thyn`, `texosquery` also wraps non-ASCII characters in the argument of `\fwrp` (which expands to `\texosquerynonasciidetokwrap`) and

`\twrp` (which expands to `\texosquerynonasciiwrap`). The expanded versions of these commands may be changed to deal with non-ASCII characters, but in general it's simpler to either use the same encoding as the document or use the hooks to temporarily switch encoding while the information is read.

2 Generic Use

The commands available for generic use are defined in `tex-locale.tex`. Plain TEX users can load this using `\input`:

```
\input locale
```

LATEX users are also able to do this, but are better off loading `tex-locale.sty` instead:

```
\usepackage{tex-locale}
```

This does more than simply inputting `tex-locale.tex` as it also loads other packages as well, such as babel or polyglossia. See Section 3 for LATEX package options.

By default `tex-locale.tex` assumes that the document locale should match your own locale as identified by your operating system. This means that the same document code will produce different results when compiled in different locations. If you want information for a specific locale, then you need to identify the document's main locale and optionally other locales. For plain TEX users, this means defining `\LocaleMain` and `\LocaleOther` *before* loading `tex-locale.tex`. For LATEX users, this is done through the package options provided by `tex-locale.sty`.

`\LocaleMain`

```
\LocaleMain
```

If this is defined before `tex-locale.tex` is input, then this may be defined as the keyword `locale`:

```
\def\LocaleMain{locale}
```

or the relevant language tag. For example:

```
\def\LocaleMain{de-CH-1996}
```

The `locale` keyword indicates that the main locale should be found by querying the operating system (using `texosquery`). After `tex-locale.tex` has been input, this command will be set to the language tag for the main locale. If `\LocaleMain` is undefined before `tex-locale.tex` is input, then `locale` is assumed.

`\LocaleOther`

```
\LocaleOther
```

If this is defined before `tex-locale.tex` is input, then this should be a comma-separated list of language tags for the other locales. If `\LocaleMain` has been defined to a language tag, the list may also include the keyword `locale` to indicate the operating system's locale.

Note that the generic `tex-locale.tex` doesn't load babel or polyglossia, but it does track each language using tracklang's interface. You can switch to any of these locales using

\selectlocale

```
\selectlocale{⟨locale⟩}
```

where ⟨*locale*⟩ is either a language tag or tracklang dialect label. If the locale is unrecognised, an error will occur unless texosquery's dry run mode is on, in which case it will just be a warning.

This uses tracklang's `\SetCurrentTrackedDialect{`⟨*dialect*⟩`}` but additionally, if `\selectlanguage` has been defined, it will try to determine the correct label to pass to `\selectlanguage`.

This means that if you use LaTeX with `tex-locale.sty`, which loads babel or polyglossia, then you don't need to remember, for example, that `en-GB` has the babel dialect label `british`, and can simply do

```
\selectlocale{en-GB}
```

If `\selectlanguage` isn't defined, then this command doesn't switch the document language (that is, it doesn't change the hyphenation patterns or `\languagename`) but it does allow the `\CurrentLocale...` commands (described in Section 2.7) to reflect the change in locale.

Here's a plain XeTeX document that sets up three locales for the document. The first (main one) is obtained from the operating system (because `\LocaleMain` isn't defined) and the other two are explicitly set by defining `\LocaleOther` before `tex-locale.tex` is input:

```
\def\LocaleOther{fr-BE,de-CH-1996}

\input locale

Currency: \CurrentLocaleCurrency.

\selectlocale{fr-BE}

Currency: \CurrentLocaleCurrency.

\selectlocale{de-CH-1996}

Currency: \CurrentLocaleCurrency.
\bye
```

If this document is in a file called `test.tex`, then it can be compiled using:

```
pdftex --shell-escape test
```

or

```
etex --shell-escape test
```

or

```
xetex --shell-escape test
```

or

```
luatex --shell-escape test
```

(Omit `--shell-escape` if the application given by `\TeXOSInvokerName` is on the restricted list.)

The equivalent LaTeX document is:

```
\documentclass{article}

\usepackage[other={fr-BE,de-CH-1996}]{tex-locale}

\begin{document}
Currency: \CurrentLocaleCurrency.

\selectlocale{fr-BE}

Currency: \CurrentLocaleCurrency.

\selectlocale{de-CH-1996}

Currency: \CurrentLocaleCurrency.

\end{document}
```

If this document is in a file called `test.tex`, then it can be compiled using:

```
pdflatex --shell-escape test
```

or

```
latex --shell-escape test
```

or

```
xelatex --shell-escape test
```

or

```
lualatex --shell-escape test
```

(Again, omit `--shell-escape` if the application given by `\TeXOSInvokerName` is on the restricted list.)

In the first two cases (`pdflatex` or `latex`), babel will be loaded (and also fontenc and inputenc and various other packages). Whereas in the last two cases (`xelatex` or `lualatex`), polyglossia will be loaded (and also fontspec and various other packages). This means that `\selectlocale` will also

use \selectlanguage to switch the document language. In the plain TEX version, \languagename remains undefined.

The other difference between the plain TEX and the LATEX examples, is the default output produced by \CurrentLocaleCurrency. The plain TEX version uses the currency codes (for example, EUR) whereas the LATEX version uses currency symbols. These differences are described in more detail below.

2.1 General Information

Each invocation of texosquery requires starting up the Java Virtual Machine, so tex-locale.tex minimises this overhead by only using a single system call. The tex-locale.sty LATEX package may need additional information that also requires a texosquery call before tex-locale.tex is input. So LATEX users will have at most two system calls to texosquery when loading tex-locale.sty whereas plain TEX users will only have a single system call when loading just tex-locale.tex.

Since it's necessary to run texosquery to obtain all the locale information, tex-locale.tex uses the opportunity to also fetch more general information that may be of use.

\LocaleOSname

```
\LocaleOSname
```

This expands to the operating system name. For example, Linux.

\LocaleOSversion

```
\LocaleOSversion
```

This expands to the operating system version. For example, 4.1.13-100.fc21.x86_64. (Note that the underscore in this case has been detokenized.)

\LocaleOSarch

```
\LocaleOSarch
```

This expands to the architecture. For example, amd64.

\LocaleOScodeset

```
\LocaleOScodeset
```

This expands to the operating system's default file encoding (or the value of the Java setting file.encoding). This uses texosquery's -C switch, which converts the encoding name to lower case and strips the hyphens. (For example, UTF-8 is converted to utf8.) If you prefer to use the -cs (or --codeset) switch, you can define

\LocaleQueryCodesetParam

```
\LocaleQueryCodesetParam
```

to the required switch. For example:

```
\def\LocaleQueryCodesetParam{-cs}
\input locale
Encoding: \LocaleOScodeset.
\bye
```

(Note that `\codeset` was added to `texosquery` v1.6.)

\LocaleOStag

```
\LocaleOStag
```

This expands to the operating system's language tag, but note that this doesn't mean that this locale has been tracked (either as the main locale or one of the other document locales). It's for informational purposes only.

\LocaleNowStamp

```
\LocaleNowStamp
```

This expands to the current date-time in PDF format. This is like the PDF primitive `\pdfcreationdate` (also available with LuaTEX through `\pdffeedback`). Since `\pdfcreationdate` isn't provided by XETEX, `\LocaleNowStamp` can be used instead if required.

You can also obtain the file modification date of the document source file. The file is assumed to be `\jobname.tex`, but you can choose another instead by first defining

\LocaleMainFile

```
\LocaleMainFile
```

before loading `tex-locale.tex`. The value should be the file name. For example:

```
\def\LocaleMainFile{mydoc.tex}
\input locale
```

You can define this command to empty if you don't want the file modification PDF date-stamp. For example:

```
\def\LocaleMainFile{}
\input locale
```

If the file name contains spaces, you can't use the restricted shell escape and will have to use the less secure unrestricted mode instead.

If `\LocaleMainFile` is not empty, the modification date can be accessed using

\LocaleFileMod

```
\LocaleFileMod
```

For example:

```
\input locale

\pdfinfo{
  /CreationDate (\LocaleNowStamp)
  /ModDate (\LocaleFileMod)
}

Now: \LocaleNowStamp.
Mod: \LocaleFileMod.

\bye
```

This command will be empty if the modification date wasn't found.

2.2 Attributes

The `tex-locale.tex` package stores and references information through the use of attributes. These are set using:

\LocaleSetAttribute

```
\LocaleSetAttribute{⟨label⟩}{⟨attribute⟩}{⟨value⟩}
```

and accessed using

\LocaleGetAttribute

```
\LocaleGetAttribute{⟨label⟩}{⟨attribute⟩}
```

The ⟨*label*⟩ identifies the type of information, ⟨*attribute*⟩ is the attribute's label and ⟨*value*⟩ is the attribute's value.

There are three particular types of identifiers: locales, regions and currencies. For convenience, there are some shortcut commands so you don't need to use the special prefix part of the label.

For locales, where ⟨*dialect*⟩ is the tracklang dialect label associated with that locale:

\LocaleSetDialectAttribute

```
\LocaleSetDialectAttribute{⟨dialect⟩}{⟨attribute⟩}{⟨value⟩}
```

\LocaleGetDialectAttribute

```
\LocaleGetDialectAttribute{⟨dialect⟩}{⟨attribute⟩}
```

Region attributes are set using:

\LocaleSetRegionAttribute

```
\LocaleSetRegionAttribute{⟨region code⟩}{⟨attribute⟩}{⟨value⟩}
```

where ⟨*region code*⟩ is the ISO territory code (for example, GB or CH). You can fetch a region's attribute value using:

\LocaleGetRegionAttribute

```
\LocaleGetRegionAttribute{⟨region code⟩}{⟨attribute⟩}
```

Currency attributes are set using:

\LocaleSetCurrencyAttribute

```
\LocaleSetCurrencyAttribute{⟨currency code⟩}{⟨attribute⟩}{⟨value⟩}
```

where ⟨*currency code*⟩ is the ISO currency code (for example, GBP or EUR). You can fetch a currency's attribute value using:

\LocaleGetCurrencyAttribute

```
\LocaleGetCurrencyAttribute{⟨currency code⟩}{⟨attribute⟩}
```

There's a debugging command available that uses \show to show an attribute's value (or an error if undefined):

\localeshowattribute

```
\localeshowattribute{⟨label⟩}{⟨attribute⟩}
```

Again there are shortcuts for the dialect, region and currency attribute groups:

\localeshowdialectattribute

```
\localeshowdialectattribute{⟨dialect⟩}{⟨attribute⟩}
```

\localeshowregionattribute

```
\localeshowregionattribute{⟨region code⟩}{⟨attribute⟩}
```

\localeshowcurrencyattribute

```
\localeshowcurrencyattribute{⟨currency code⟩}{⟨attribute⟩}
```

Further details of these and other attribute commands can be found in Section 4.2.3.

For the most part, the attributes are stored and accessed internally by the various commands described in this chapter, so you won't usually need to worry about them. However, you can modify the attributes. For example, the de-CH-1996 locale uses an apostrophe ’ as the numeric group separator. This is actually returned by texosquery as the 0x27 straight apostrophe from the Basic Latin

block. This isn't a problem for plain TEX using the default Computer Modern font, as illustrated by the following document:

```
1'234.5
\bye
```

This uses a curly quote in the resulting 1'234.5. However, changing the font can result in a straight apostrophe instead, as illustrated in this plain XETEX document:

```
\font\nimbus="NimbusRoman-Regular" at 10pt
\nimbus

1'234.5
\bye
```

This isn't a problem with XELATEX if the fontspec package is used:

```
\documentclass{article}
\usepackage{fontspec}
\setmainfont{NimbusRoman-Regular}

\begin{document}
1'234.5
\end{document}
```

So a plain XETEX user might prefer to change the numeric group separator to the right single quote character ' (0x2019). The numeric group separator is provided by the `groupsep` dialect attribute. The tracklang dialect label corresponding to the `de-CH-1996` locale is `nswissgerman`. So the group separator for this dialect can be changed by setting the attribute to the new value using:

```
\LocaleSetDialectAttribute{nswissgerman}{groupsep}{'}
```

(See Section 2.2.2 to convert a language tag to dialect label.)

2.2.1 Attribute Lists

Some attribute values may be comma-separated lists. You can add a unique item to a list using:

\LocaleAddToAttributeList

```
\LocaleAddToAttributeList{⟨label⟩}{⟨attribute⟩}{⟨item⟩}
```

This will do nothing if the item is already in the list, otherwise it will append ⟨*item*⟩ to the list. No expansion is performed on ⟨*item*⟩. The item may be empty.

\LocaleXpAddToAttributeList

```
\LocaleXpAddToAttributeList{⟨label⟩}{⟨attribute⟩}{⟨item⟩}
```

As above but the first token of ⟨*item*⟩ is expanded before being added to the list.

You can test if an item is in one of those lists using:

\LocaleIfInAttributeList

```
\LocaleIfInAttributeList{⟨label⟩}{⟨attribute⟩}{⟨item⟩}{⟨true⟩}{⟨false⟩}
```

where ⟨*item*⟩ is the item. This does ⟨*true*⟩ if ⟨*item*⟩ is found in the ⟨*attribute*⟩ list for ⟨*label*⟩, otherwise it does ⟨*false*⟩. If the attribute hasn't been defined, ⟨*false*⟩ is done.

There's a similar command that expands the first token of ⟨*item*⟩ before performing the test:

\LocaleIfInAttributeList

```
\LocaleIfXpInAttributeList{⟨label⟩}{⟨attribute⟩}{⟨item⟩}{⟨true⟩}{⟨false⟩}
```

You can iterate over the list using:

\LocaleForEachInAttributeList

```
\LocaleForEachInAttributeList{⟨label⟩}{⟨attribute⟩}{⟨cs⟩}{⟨body⟩}
```

This iterates over each element of the list, setting the control sequence ⟨*cs*⟩ to the current item and performing ⟨*body*⟩. Note that this doesn't skip empty items.

There are shortcut commands for the dialect, region and currency attributes:

\LocaleAddToDialectAttributeList

```
\LocaleAddToDialectAttributeList{⟨label⟩}{⟨attribute⟩}{⟨item⟩}
```

\LocaleXpAddToDialectAttributeList

```
\LocaleXpAddToDialectAttributeList{⟨label⟩}{⟨attribute⟩}{⟨item⟩}
```

\LocaleIfInDialectAttributeList

```
\LocaleIfInDialectAttributeList{⟨label⟩}{⟨attribute⟩}{⟨item⟩}{⟨true⟩}{⟨false⟩}
```

\LocaleIfInDialectAttributeList

```
\LocaleIfXpInDialectAttributeList{⟨label⟩}{⟨attribute⟩}{⟨item⟩}{⟨true⟩}
{⟨false⟩}
```

\LocaleForEachInDialectAttributeList

```
\LocaleForEachInDialectAttributeList{⟨label⟩}{⟨attribute⟩}{⟨cs⟩}{⟨body⟩}
```

\LocaleAddToRegionAttributeList

```
\LocaleAddToRegionAttributeList{⟨label⟩}{⟨attribute⟩}{⟨item⟩}
```

\LocaleXpAddToRegionAttributeList

```
\LocaleXpAddToRegionAttributeList{⟨label⟩}{⟨attribute⟩}{⟨item⟩}
```

\LocaleIfInRegionAttributeList

```
\LocaleIfInRegionAttributeList{⟨label⟩}{⟨attribute⟩}{⟨item⟩}{⟨true⟩}{⟨false⟩}
```

\LocaleIfInRegionAttributeList

```
\LocaleIfXpInRegionAttributeList{⟨label⟩}{⟨attribute⟩}{⟨item⟩}{⟨true⟩}{⟨false⟩}
```

\LocaleForEachInRegionAttributeList

```
\LocaleForEachInRegionAttributeList{⟨label⟩}{⟨attribute⟩}{⟨cs⟩}{⟨body⟩}
```

\LocaleAddToCurrencyAttributeList

```
\LocaleAddToCurrencyAttributeList{⟨label⟩}{⟨attribute⟩}{⟨item⟩}
```

\LocaleXpAddToCurrencyAttributeList

```
\LocaleXpAddToCurrencyAttributeList{⟨label⟩}{⟨attribute⟩}{⟨item⟩}
```

\LocaleIfInCurrencyAttributeList

```
\LocaleIfInCurrencyAttributeList{⟨label⟩}{⟨attribute⟩}{⟨item⟩}{⟨true⟩}{⟨false⟩}
```

\LocaleIfInCurrencyAttributeList

```
\LocaleIfXpInCurrencyAttributeList{⟨label⟩}{⟨attribute⟩}{⟨item⟩}{⟨true⟩}
{⟨false⟩}
```

\LocaleForEachInCurrencyAttributeList

```
\LocaleForEachInCurrencyAttributeList{⟨label⟩}{⟨attribute⟩}{⟨cs⟩}{⟨body⟩}
```

2.2.2 General Attributes

The language tag attribute type `tagtodialect` is set using

`\LocaleSetAttribute{⟨label⟩}{tagtodialect}{⟨value⟩}`

and can be accessed using

`\LocaleGetAttribute{⟨label⟩}{tagtodialect}`

The ⟨*label*⟩ is a language tag (such as `de-CH-1996`) and the ⟨*value*⟩ is the associated tracklang dialect label (for example, `nswissgerman`). This attribute is used by `\selectlocale` to convert the language tag to the corresponding dialect label.

If the time zone information was fetched with `texosquery`'s `-Z` switch, then the time zone IDs will be stored in the time zone `id` attribute list that can be fetched with

```
\LocaleGetAttribute{timezone}{id}
```

or iterated over using `\LocaleForEachInAttributeList`. For example:

```
\def\LocaleIfDateTimePatternsSupported#1#2{#1}

\input locale

Time zone IDs:

\LocaleForEachInAttributeList{timezone}{id}{\ThisId}{\ThisId\endgraf}

\bye
```

This simply lists all the known time zone identifiers. (Since `\LocaleForEachInAttributeList` is a short command, `\endgraf` is used to create a paragraph break in the above example.)

There are two general currency attribute lists, where the ⟨*label*⟩ is `currencies` and the attribute values are `official` for the list of official currency codes and `regional` for the regional currency codes. For example:

```
\def\LocaleMain{en-GB}
\def\LocaleOther{en-IM,en-GG,fr-BE,nl-BE,de-DE-1996}

\input locale

Official: \LocaleGetAttribute{currencies}{official}.
Regional: \LocaleGetAttribute{currencies}{regional}.

\bye
```

produces:

> Official: GBP,EUR. Regional: GBP,IMP,GGP,EUR.

You can iterate over these lists using `\LocaleForEachInAttributeList`. For example:

```
% arara: xetex: {shell: on}
\font\nimbus="NimbusRoman-Regular" at 10pt
\nimbus

\def\LocaleMain{en-GB}
```

```
\def\LocaleOther{en-IM,en-GG,fr-BE,nl-BE,de-DE-1996}

\input locale

\LocaleForEachInAttributeList{currencies}{regional}{\thiscode}%
{\thiscode\ (\LocaleGetCurrencyAttribute{\thiscode}{sym}) }

\bye
```

which produces:

GBP (£) IMP (M£) GGP (£) EUR (€)

2.2.3 Dialect Attributes

The following attributes are associated with dialects and can be set using

`\LocaleSetDialectAttribute{`⟨*dialect*⟩`}{`⟨*attribute*⟩`}{`⟨*value*⟩`}`

or fetched with

`\LocaleGetDialectAttribute{`⟨*dialect*⟩`}{`⟨*attribute*⟩`}`

where ⟨*dialect*⟩ is the tracklang dialect label. Known values of ⟨*attribute*⟩ are listed below.

2.2.3.1 General Dialect Attributes

- `langtag`: the ⟨*value*⟩ is the language tag associated with the dialect label ⟨*dialect*⟩.
- `langname`: the ⟨*value*⟩ is the language name associated with the dialect label ⟨*dialect*⟩.
- `nativelangname`: the ⟨*value*⟩ is the native language name associated with the dialect label ⟨*dialect*⟩.
- `regionname`: the ⟨*value*⟩ is the region name associated with the dialect label ⟨*dialect*⟩.
- `nativeregionname`: the ⟨*value*⟩ is the native region name associated with the dialect label ⟨*dialect*⟩.
- `variantname`: the ⟨*value*⟩ is the variant name associated with the dialect label ⟨*dialect*⟩.
- `nativevariantname`: the ⟨*value*⟩ is the native variant name associated with the dialect label ⟨*dialect*⟩.

2.2.3.2 Dates and Times Dialect Attributes

The commands described in Section 2.5 use these attributes.

- `fulldate`: the ⟨*value*⟩ is the full date associated with the dialect label ⟨*dialect*⟩. (Used by `\LocaleFullDate`.)
- `longdate`: the ⟨*value*⟩ is the long date associated with the dialect label ⟨*dialect*⟩. (Used by `\LocaleLongDate`.)
- `meddate`: the ⟨*value*⟩ is the medium date associated with the dialect label ⟨*dialect*⟩. (Used by `\LocaleMediumDate`.)
- `shortdate`: the ⟨*value*⟩ is the short date associated with the dialect label ⟨*dialect*⟩. (Used by `\LocaleShortDate`.)
- `firstday`: the ⟨*value*⟩ is the index of the first day of the week associated with the dialect label ⟨*dialect*⟩. (Used by `\LocaleFirstDayIndex`.)
- `fulltime`: the ⟨*value*⟩ is the full time associated with the dialect label ⟨*dialect*⟩. (Used by `\LocaleFullTime`.)
- `longtime`: the ⟨*value*⟩ is the long time associated with the dialect label ⟨*dialect*⟩. (Used by `\LocaleLongTime`.)
- `medtime`: the ⟨*value*⟩ is the medium time associated with the dialect label ⟨*dialect*⟩. (Used by `\LocaleMediumTime`.)
- `shorttime`: the ⟨*value*⟩ is the short time associated with the dialect label ⟨*dialect*⟩. (Used by `\LocaleShortTime`.)
- `fulldatetime`: the ⟨*value*⟩ is the full date and time associated with the dialect label ⟨*dialect*⟩. (Used by `\LocaleFullDateTime`.)
- `longdatetime`: the ⟨*value*⟩ is the long date and time associated with the dialect label ⟨*dialect*⟩. (Used by `\LocaleLongDateTime`.)
- `meddatetime`: the ⟨*value*⟩ is the medium date and time associated with the dialect label ⟨*dialect*⟩. (Used by `\LocaleMediumDateTime`.)
- `shortdatetime`: the ⟨*value*⟩ is the short date and time associated with the dialect label ⟨*dialect*⟩. (Used by `\LocaleShortDateTime`.)

Note that the combined date and time attributes (such as `fulldatetime`) aren't used by `\CurrentLocaleDat`

Patterns The following attributes store date or time patterns (see Section 2.3.2).

- `fulldatefmt`: the ⟨*value*⟩ is the full date format associated with the dialect label ⟨*dialect*⟩.
- `longdatefmt`: the ⟨*value*⟩ is the long date format associated with the dialect label ⟨*dialect*⟩.
- `meddatefmt`: the ⟨*value*⟩ is the medium date format associated with the dialect label ⟨*dialect*⟩.
- `shortdatefmt`: the ⟨*value*⟩ is the short date format associated with the dialect label ⟨*dialect*⟩.
- `fulltimefmt`: the ⟨*value*⟩ is the full time format associated with the dialect label ⟨*dialect*⟩.
- `longtimefmt`: the ⟨*value*⟩ is the long time format associated with the dialect label ⟨*dialect*⟩.
- `medtimefmt`: the ⟨*value*⟩ is the medium time format associated with the dialect label ⟨*dialect*⟩.
- `shorttimefmt`: the ⟨*value*⟩ is the short time format associated with the dialect label ⟨*dialect*⟩.
- `fulldatetimefmt`: the ⟨*value*⟩ is the full date time format associated with the dialect label ⟨*dialect*⟩.
- `longdatetimefmt`: the ⟨*value*⟩ is the long date time format associated with the dialect label ⟨*dialect*⟩.
- `meddatetimefmt`: the ⟨*value*⟩ is the medium date time format associated with the dialect label ⟨*dialect*⟩.
- `shortdatetimefmt`: the ⟨*value*⟩ is the short date time format associated with the dialect label ⟨*dialect*⟩.

Day Names The commands `\LocaleDayName`, `\LocaleShortDayName`, `\LocaleStandaloneDayName` and `\LocaleStandaloneShortDayName` use these attributes.

- `day.0`: the ⟨*value*⟩ is the name for day 0 (Monday) associated with the dialect label ⟨*dialect*⟩.
- `day.1`: the ⟨*value*⟩ is the name for day 1 (Tuesday) associated with the dialect label ⟨*dialect*⟩.
- `day.2`: the ⟨*value*⟩ is the name for day 2 (Wednesday) associated with the dialect label ⟨*dialect*⟩.
- `day.3`: the ⟨*value*⟩ is the name for day 3 (Thursday) associated with the dialect label ⟨*dialect*⟩.
- `day.4`: the ⟨*value*⟩ is the name for day 4 (Friday) associated with the dialect label ⟨*dialect*⟩.
- `day.5`: the ⟨*value*⟩ is the name for day 5 (Saturday) associated with the dialect label ⟨*dialect*⟩.
- `day.6`: the ⟨*value*⟩ is the name for day 6 (Sunday) associated with the dialect label ⟨*dialect*⟩.
- `shortday.0`: the ⟨*value*⟩ is the short name for day 0 (Monday) associated with the dialect label ⟨*dialect*⟩.

- `shortday.1`: the ⟨*value*⟩ is the short name for day 1 (Tuesday) associated with the dialect label ⟨*dialect*⟩.
- `shortday.2`: the ⟨*value*⟩ is the short name for day 2 (Wednesday) associated with the dialect label ⟨*dialect*⟩.
- `shortday.3`: the ⟨*value*⟩ is the short name for day 3 (Thursday) associated with the dialect label ⟨*dialect*⟩.
- `shortday.4`: the ⟨*value*⟩ is the short name for day 4 (Friday) associated with the dialect label ⟨*dialect*⟩.
- `shortday.5`: the ⟨*value*⟩ is the short name for day 5 (Saturday) associated with the dialect label ⟨*dialect*⟩.
- `shortday.6`: the ⟨*value*⟩ is the short name for day 6 (Sunday) associated with the dialect label ⟨*dialect*⟩.
- `standalone.day.0`: the ⟨*value*⟩ is the name for day 0 (Monday) associated with the dialect label ⟨*dialect*⟩.
- `standalone.day.1`: the ⟨*value*⟩ is the standalone name for day 1 (Tuesday) associated with the dialect label ⟨*dialect*⟩.
- `standalone.day.2`: the ⟨*value*⟩ is the standalone name for day 2 (Wednesday) associated with the dialect label ⟨*dialect*⟩.
- `standalone.day.3`: the ⟨*value*⟩ is the standalone name for day 3 (Thursday) associated with the dialect label ⟨*dialect*⟩.
- `standalone.day.4`: the ⟨*value*⟩ is the standalone name for day 4 (Friday) associated with the dialect label ⟨*dialect*⟩.
- `standalone.day.5`: the ⟨*value*⟩ is the standalone name for day 5 (Saturday) associated with the dialect label ⟨*dialect*⟩.
- `standalone.day.6`: the ⟨*value*⟩ is the standalone name for day 6 (Sunday) associated with the dialect label ⟨*dialect*⟩.
- `standalone.shortday.0`: the ⟨*value*⟩ is the standalone short name for day 0 (Monday) associated with the dialect label ⟨*dialect*⟩.
- `standalone.shortday.1`: the ⟨*value*⟩ is the standalone short name for day 1 (Tuesday) associated with the dialect label ⟨*dialect*⟩.
- `standalone.shortday.2`: the ⟨*value*⟩ is the standalone short name for day 2 (Wednesday) associated with the dialect label ⟨*dialect*⟩.
- `standalone.shortday.3`: the ⟨*value*⟩ is the standalone short name for day 3 (Thursday) associated with the dialect label ⟨*dialect*⟩.

- `standalone.shortday.4`: the ⟨*value*⟩ is the standalone short name for day 4 (Friday) associated with the dialect label ⟨*dialect*⟩.
- `standalone.shortday.5`: the ⟨*value*⟩ is the standalone short name for day 5 (Saturday) associated with the dialect label ⟨*dialect*⟩.
- `standalone.shortday.6`: the ⟨*value*⟩ is the standalone short name for day 6 (Sunday) associated with the dialect label ⟨*dialect*⟩.

Month Names The commands `\LocaleMonthName`, `\LocaleShortMonthName`, `\LocaleStandaloneMont` and `\LocaleStandaloneShortMonthName` use these attributes.

- `month.1`: the ⟨*value*⟩ is the name for month 1 (January) associated with the dialect label ⟨*dialect*⟩.
- `month.2`: the ⟨*value*⟩ is the name for month 2 (February) associated with the dialect label ⟨*dialect*⟩.
- `month.3`: the ⟨*value*⟩ is the name for month 3 (March) associated with the dialect label ⟨*dialect*⟩.
- `month.4`: the ⟨*value*⟩ is the name for month 4 (April) associated with the dialect label ⟨*dialect*⟩.
- `month.5`: the ⟨*value*⟩ is the name for month 5 (May) associated with the dialect label ⟨*dialect*⟩.
- `month.6`: the ⟨*value*⟩ is the name for month 6 (June) associated with the dialect label ⟨*dialect*⟩.
- `month.7`: the ⟨*value*⟩ is the name for month 7 (July) associated with the dialect label ⟨*dialect*⟩.
- `month.8`: the ⟨*value*⟩ is the name for month 8 (August) associated with the dialect label ⟨*dialect*⟩.
- `month.9`: the ⟨*value*⟩ is the name for month 9 (September) associated with the dialect label ⟨*dialect*⟩.
- `month.10`: the ⟨*value*⟩ is the name for month 10 (October) associated with the dialect label ⟨*dialect*⟩.
- `month.11`: the ⟨*value*⟩ is the name for month 11 (November) associated with the dialect label ⟨*dialect*⟩.
- `month.12`: the ⟨*value*⟩ is the name for month 12 (December) associated with the dialect label ⟨*dialect*⟩.
- `shortmonth.1`: the ⟨*value*⟩ is the short name for month 1 (January) associated with the dialect label ⟨*dialect*⟩.
- `shortmonth.2`: the ⟨*value*⟩ is the short name for month 2 (February) associated with the dialect label ⟨*dialect*⟩.

- `shortmonth.3`: the ⟨*value*⟩ is the short name for month 3 (March) associated with the dialect label ⟨*dialect*⟩.
- `shortmonth.4`: the ⟨*value*⟩ is the short name for month 4 (April) associated with the dialect label ⟨*dialect*⟩.
- `shortmonth.5`: the ⟨*value*⟩ is the short name for month 5 (May) associated with the dialect label ⟨*dialect*⟩.
- `shortmonth.6`: the ⟨*value*⟩ is the short name for month 6 (June) associated with the dialect label ⟨*dialect*⟩.
- `shortmonth.7`: the ⟨*value*⟩ is the short name for month 7 (July) associated with the dialect label ⟨*dialect*⟩.
- `shortmonth.8`: the ⟨*value*⟩ is the short name for month 8 (August) associated with the dialect label ⟨*dialect*⟩.
- `shortmonth.9`: the ⟨*value*⟩ is the short name for month 9 (September) associated with the dialect label ⟨*dialect*⟩.
- `shortmonth.10`: the ⟨*value*⟩ is the short name for month 10 (October) associated with the dialect label ⟨*dialect*⟩.
- `shortmonth.11`: the ⟨*value*⟩ is the short name for month 11 (November) associated with the dialect label ⟨*dialect*⟩.
- `shortmonth.12`: the ⟨*value*⟩ is the short name for month 12 (December) associated with the dialect label ⟨*dialect*⟩.
- `standalone.month.1`: the ⟨*value*⟩ is the standalone name for month 1 (January) associated with the dialect label ⟨*dialect*⟩.
- `standalone.month.2`: the ⟨*value*⟩ is the standalone name for month 2 (February) associated with the dialect label ⟨*dialect*⟩.
- `standalone.month.3`: the ⟨*value*⟩ is the standalone name for month 3 (March) associated with the dialect label ⟨*dialect*⟩.
- `standalone.month.4`: the ⟨*value*⟩ is the standalone name for month 4 (April) associated with the dialect label ⟨*dialect*⟩.
- `standalone.month.5`: the ⟨*value*⟩ is the standalone name for month 5 (May) associated with the dialect label ⟨*dialect*⟩.
- `standalone.month.6`: the ⟨*value*⟩ is the standalone name for month 6 (June) associated with the dialect label ⟨*dialect*⟩.
- `standalone.month.7`: the ⟨*value*⟩ is the standalone name for month 7 (July) associated with the dialect label ⟨*dialect*⟩.

- `standalone.month.8`: the ⟨*value*⟩ is the standalone name for month 8 (August) associated with the dialect label ⟨*dialect*⟩.
- `standalone.month.9`: the ⟨*value*⟩ is the standalone name for month 9 (September) associated with the dialect label ⟨*dialect*⟩.
- `standalone.month.10`: the ⟨*value*⟩ is the standalone name for month 10 (October) associated with the dialect label ⟨*dialect*⟩.
- `standalone.month.11`: the ⟨*value*⟩ is the standalone name for month 11 (November) associated with the dialect label ⟨*dialect*⟩.
- `standalone.month.12`: the ⟨*value*⟩ is the standalone name for month 12 (December) associated with the dialect label ⟨*dialect*⟩.
- `standalone.shortmonth.1`: the ⟨*value*⟩ is the standalone short name for month 1 (January) associated with the dialect label ⟨*dialect*⟩.
- `standalone.shortmonth.2`: the ⟨*value*⟩ is the standalone short name for month 2 (February) associated with the dialect label ⟨*dialect*⟩.
- `standalone.shortmonth.3`: the ⟨*value*⟩ is the standalone short name for month 3 (March) associated with the dialect label ⟨*dialect*⟩.
- `standalone.shortmonth.4`: the ⟨*value*⟩ is the standalone short name for month 4 (April) associated with the dialect label ⟨*dialect*⟩.
- `standalone.shortmonth.5`: the ⟨*value*⟩ is the standalone short name for month 5 (May) associated with the dialect label ⟨*dialect*⟩.
- `standalone.shortmonth.6`: the ⟨*value*⟩ is the standalone short name for month 6 (June) associated with the dialect label ⟨*dialect*⟩.
- `standalone.shortmonth.7`: the ⟨*value*⟩ is the standalone short name for month 7 (July) associated with the dialect label ⟨*dialect*⟩.
- `standalone.shortmonth.8`: the ⟨*value*⟩ is the standalone short name for month 8 (August) associated with the dialect label ⟨*dialect*⟩.
- `standalone.shortmonth.9`: the ⟨*value*⟩ is the standalone short name for month 9 (September) associated with the dialect label ⟨*dialect*⟩.
- `standalone.shortmonth.10`: the ⟨*value*⟩ is the standalone short name for month 10 (October) associated with the dialect label ⟨*dialect*⟩.
- `standalone.shortmonth.11`: the ⟨*value*⟩ is the standalone short name for month 11 (November) associated with the dialect label ⟨*dialect*⟩.
- `standalone.shortmonth.12`: the ⟨*value*⟩ is the standalone short name for month 12 (December) associated with the dialect label ⟨*dialect*⟩.

2.2.3.3 Time Zones

These attributes won't be set if the time zone information wasn't fetched (with `texosquery`'s `-Z` switch). These attributes include the time zone identifier in the attribute label. (The list of known identifiers is stored in the `timezone` `id` attribute.) For example, `Europe/London` is the identifier for the UK time zone. So the attribute `timezone.Europe/London.short` is for the short name for that zone (`GMT`) and `timezone.Europe/London.shortdst` is for the short daylight saving name for that zone (`BST`).

- `timezone.`⟨*zone*⟩`.short`: the ⟨*value*⟩ is the short name for the zone identified by ⟨*zone*⟩ associated with the dialect label ⟨*dialect*⟩.
- `timezone.`⟨*zone*⟩`.long`: the ⟨*value*⟩ is the long name for the zone identified by ⟨*zone*⟩ associated with the dialect label ⟨*dialect*⟩.
- `timezone.`⟨*zone*⟩`.shortdst`: the ⟨*value*⟩ is the short daylight saving name for the zone identified by ⟨*zone*⟩ associated with the dialect label ⟨*dialect*⟩.
- `timezone.`⟨*zone*⟩`.longdst`: the ⟨*value*⟩ is the long daylight saving name for the zone identified by ⟨*zone*⟩ associated with the dialect label ⟨*dialect*⟩.

2.2.3.4 Numeric

- `groupsep`: the ⟨*value*⟩ is the numeric group separator associated with the dialect label ⟨*dialect*⟩. (Used by `\LocaleNumericGroupSep`.)
- `decsep`: the ⟨*value*⟩ is the numeric decimal separator associated with the dialect label ⟨*dialect*⟩. (Used by `\LocaleNumericDecimalSep`.)
- `currencysep`: the ⟨*value*⟩ is the numeric currency separator associated with the dialect label ⟨*dialect*⟩. (Used by `\LocaleNumericMonetarySep`.)
- `exp`: the ⟨*value*⟩ is the exponent symbol associated with the dialect label ⟨*dialect*⟩. (Used by `\LocaleNumericExponent`.)
- `usesgroup`: the ⟨*value*⟩ (either 1 or 0) indicates if the number group separator should be used with the dialect label ⟨*dialect*⟩. (Used by `\LocaleIfNumericUsesGroup`.)
- `percent`: the ⟨*value*⟩ is the percentage symbol associated with the dialect label ⟨*dialect*⟩. (Used by `\LocaleNumericPercent`.)
- `permill`: the ⟨*value*⟩ is the per mill symbol associated with the dialect label ⟨*dialect*⟩. (Used by `\LocaleNumericPermill`.)
- `decfmt`: the ⟨*value*⟩ is the decimal format associated with the dialect label ⟨*dialect*⟩. (Used by `\CurrentLocaleDecimalPattern`.)
- `intfmt`: the ⟨*value*⟩ is the integer format associated with the dialect label ⟨*dialect*⟩. (Used by `\CurrentLocaleIntegerPattern`.)

- `curfmt`: the ⟨*value*⟩ is the currency format associated with the dialect label ⟨*dialect*⟩. (Used by `\CurrentLocaleCurrencyPattern`.)
- `perfmt`: the ⟨*value*⟩ is the percent format associated with the dialect label ⟨*dialect*⟩. (Used by `\CurrentLocalePercentPattern`.)
- `currency`: the ⟨*value*⟩ is the currency code associated with the dialect label ⟨*dialect*⟩. (Used by `\LocaleCurrencyLabel`.)
- `regionalcurrency`: the ⟨*value*⟩ is the regional currency code associated with the dialect label ⟨*dialect*⟩. (Used by `\LocaleCurrencyRegionalLabel`.)
- `currencysym`: the ⟨*value*⟩ is the currency symbol associated with the dialect label ⟨*dialect*⟩. (Used by `\LocaleCurrencySymbol`, if the `currency` attribute isn't `XXX`.)
- `currencytex`: the ⟨*value*⟩ is the TEX code representing the currency symbol associated with the dialect label ⟨*dialect*⟩. (Used by `\LocaleCurrencyTeXSymbol`, if the `currency` attribute isn't `XXX`.)

Note that the currency information is also available through the currency attributes `official`, `sym` and `tex`. The dialect attribute allows for a different symbol to be used within the context of a specific dialect. For example, if a country has multiple scripts (such as Latin and Cyrillic) then the dialect currency symbol can reflect a specific script, whereas the currency `sym` attribute might be in the default script.

2.2.4 Region Attributes

The following attributes are associated with region and can be set using

`\LocaleSetRegionAttribute{`⟨*region code*⟩`}{`⟨*attribute*⟩`}{`⟨*value*⟩`}`

or fetched with

`\LocaleGetRegionAttribute{`⟨*region code*⟩`}{`⟨*attribute*⟩`}`

- `currency`: the regional currency code (for example, `IMP`).
- `dialect`: a comma-separated list of the dialects associated with this region. For example:

```
\def\LocaleMain{en-GB}
\def\LocaleOther{fr-BE,nl-BE}
\input locale

GB: \LocaleGetRegionAttribute{GB}{dialect}.
BE: \LocaleGetRegionAttribute{BE}{dialect}.

\bye
```

produces: GB: british. BE: belgique,flemish.

2.2.5 Currency Attributes

The following attributes are associated with currency and can be set using

`\LocaleSetCurrencyAttribute{`⟨*code*⟩`}{`⟨*attribute*⟩`}{`⟨*value*⟩`}`

or fetched with

`\LocaleGetCurrencyAttribute{`⟨*code*⟩`}{`⟨*attribute*⟩`}`

where ⟨*code*⟩ is the ISO currency code or regional currency code.

- `region`: a comma-separated list of regions that use this currency. If the regional currency code is different from the official ISO currency code, then both codes will have this attribute set. For example:

  ```
  \def\LocaleMain{en-GB}
  \def\LocaleOther{en-IM}

  \input locale

  GBP: \LocaleGetCurrencyAttribute{GBP}{region}.
  IMP: \LocaleGetCurrencyAttribute{IMP}{region}.

  \bye
  ```

  produces: GBP: GB,IM. IMP: IM.

- `official`: the official ISO currency code for ⟨*code*⟩. In most cases this will be the same as ⟨*code*⟩. For example:

  ```
  \def\LocaleMain{en-GB}
  \def\LocaleOther{en-IM}

  \input locale

  GBP: \LocaleGetCurrencyAttribute{GBP}{official}.
  IMP: \LocaleGetCurrencyAttribute{IMP}{official}.

  \bye
  ```

  produces: GBP: GBP. IMP: GBP.

- `sym`: the currency symbol (using characters, not TEX commands, except for `\$`) for regional currency ⟨*code*⟩.

- `tex`: the currency symbol using TEX code (such as `\texosquerycurrencypound`) for regional currency ⟨*code*⟩.

2.3 Patterns

The texosquery package provides for two types of patterns: date-time and numeric. The date-time patterns may just display a date, or just a time or both. They may or may not include a time zone. The numeric patterns can be used to format integers, decimals, scientific notation, percentages or currency. New patterns can be defined by commands provided in `texosquery.tex`. The tex-locale package can also fetch the patterns for the document locales so that they can be applied to data in the document to match the current locale.

The numeric patterns are fairly easy to use but are tricky to define. The date-time patterns are fairly easy to define but are tricky to use.

2.3.1 Numeric Patterns

The texosquery numeric pattern symbol commands are redefined by tex-locale to use the appropriate symbol from the current locale. For example, `\texosquerypatfmtgroupsep` is redefined to use `\CurrentLocaleNumericGroupSep`. (See the texosquery documentation for further details of these pattern commands.)

You can access a numeric pattern for a particular locale by querying one of the dialect attributes listed in Section 2.2.3.4: `intfmt` (integer pattern), `decfmt` (decimal pattern), `curfmt` (currency pattern) or `perfmt` (percent pattern).

These can be accessed explicitly using

```
\LocaleGetDialectAttribute{⟨dialect⟩}{⟨attribute⟩}
```

For example:

```
\LocaleGetDialectAttribute{british}{intfmt}
```

However it's simpler to use the shortcut commands described in Section 2.7.3. For example:

```
\CurrentLocaleIntegerPattern
```

A pattern can be used in two ways. If it's simply used in the document, then it just reproduces the pattern specification. For example:

```
\def\LocaleMain{en-GB}
\input locale

{\tt \CurrentLocaleDecimalPattern}
\bye
```

This simply produces:

```
#,###,###,##0.##########
```

This can be used for debugging purposes to check the pattern.

The second use is to apply the pattern to a number. This is done with texosquery's `\texosqueryfmtnumber` command:

```
\texosqueryfmtnumber{⟨pattern⟩}{⟨int part⟩}{⟨frac part⟩}{⟨mantissa⟩}
```

where ⟨*pattern*⟩ is the pattern (defined using `\texosquerydefpattern` or fetched with the `texosquery` application), ⟨*int part*⟩ is the integer part, ⟨*frac part*⟩ is the fractional part and ⟨*mantissa*⟩ is the exponent part.

For example:

```
% arara: xetex: {shell: on}

\def\LocaleMain{en-GB}
\input locale

\texosqueryfmtnumber{\CurrentLocaleIntegerPattern}{12}{34567}{4}

\texosqueryfmtnumber{\CurrentLocaleCurrencyPattern}{12345}{67}{0}

\texosqueryfmtnumber{\CurrentLocaleDecimalPattern}{123}{4567}{2}

\texosqueryfmtnumber{\CurrentLocalePercentPattern}{123}{4567}{2}
\bye
```

This produces:

123,456

GBP12,345.67

12,345.67

1,234,567%

There are some shortcut commands provided, to allow for more convenient code.

A numeric pattern can be applied to a numeric value using

`\localenumfmt`

```
\localenumfmt{⟨pattern⟩}{⟨decimal⟩}
```

where ⟨*pattern*⟩ is the pattern (as for `\texosqueryfmtnumber`) and ⟨*decimal*⟩ is the numeric value in the form ⟨*int*⟩`.`⟨*frac*⟩`E`⟨*mantissa*⟩ (where `.` represents the decimal separator and `E` represents the exponent separator, regardless of locale). The `.`⟨*frac*⟩ and `E`⟨*mantissa*⟩ parts are optional.

The `\localenumfmt` command splits up ⟨*decimal*⟩ into the ⟨*int part*⟩, ⟨*frac part*⟩ and ⟨*mantissa*⟩ required by `\texosqueryfmtnumber`. It then does

⟨*fmt cs*⟩{\texosqueryfmtnumber{⟨*pattern*⟩}{⟨*int part*⟩}{⟨*frac part*⟩}{⟨*mantissa*⟩}

where ⟨*fmt cs*⟩ is one of:

\localenumfmtneg

```
\localenumfmtneg{⟨text⟩}
```

if the ⟨*int part*⟩ is negative,

\localenumfmtzero

```
\localenumfmtzero{⟨text⟩}
```

if the ⟨*int part*⟩, ⟨*frac part*⟩ and ⟨*mantissa*⟩ are all zero, otherwise

\localenumfmtpos

```
\localenumfmtpos{⟨text⟩}
```

These three commands all simply default to just doing ⟨*text*⟩.

\localeint

```
\localeint{⟨value⟩}
```

This is equivalent to

```
\localenumfmt{\CurrentLocaleIntegerPattern}{⟨value⟩}
```

\localedec

```
\localedec{⟨decimal⟩}
```

As above but uses \CurrentLocaleDecimalPattern.

\localecur

```
\localecur{⟨decimal⟩}
```

As above but uses \CurrentLocaleCurrencyPattern.

\localeper

```
\localeper{⟨decimal⟩}
```

As above but uses \CurrentLocalePercentPattern.

The above example can be simplified to:

```
\def\LocaleMain{en-GB}
\input locale
```

```
\localeint{12.34567E4}

\localecur{12345.67}

\localedec{123.4567E2}

\localeper{123.4567E2}
\bye
```

This produces the same result as before.

As described in Section 2.7.2, the currency unit can be switched to a symbol by redefining `\localecurrchoice`. For example:

```
\font\nimbus="NimbusRoman-Regular" at 10pt
\nimbus

\def\LocaleMain{en-GB}
\def\LocaleOther{pt-BR,de-CH-1996}
\input locale

\def\localecurrchoice#1#2#3#4{#3}

en-GB: \localecur{12345.67}

\selectlocale{pt-BR}
pt-BR: \localecur{12345.67}

\selectlocale{de-CH-1996}
de-CH-1996: \localecur{12345.67}

\bye
```

This produces:

en-GB: £12,345.67

pt-BR: R$ 12.345,67

de-CH-1996: SFr. 12’345.67

You can define your own pattern using the commands provided by texosquery. For example:

```
\texosquerydefpattern{\sinumpattern}{%
  \sinumfmt
  {\decfmt{\#\,\#\#\#\,\#\#\#\,\#\#\0}{\0\#\#\#\#\#\#\#\#\#}}%
  {\-\#\#\#\#\#\#\#\#\0\0}%
}
```

This can then be used with \localenumfmt. The texosquery documentation includes other examples, including a currency pattern that shifts the sign before the currency symbol for negative amounts:

```
\texosquerydefpattern{\curpattern}{%
\pmnumfmt
{\pcur{\decfmt{\#\,\#\#\#\,\#\#\#\,\#\#0}{\0\0\#\#\#\#\#\#\#\#}}{}}%
{\pcur{\decfmt{\#\,\#\#\#\,\#\#\#\,\#\#0}{\0\0\#\#\#\#\#\#\#\#}}{\-}}}
```

(No sign is used for positive amounts.)

Here's a complete document:

```
\font\nimbus="NimbusRoman-Regular" at 10pt
\nimbus

\def\LocaleMain{en-GB}
\def\LocaleOther{pt-BR,de-CH-1996}
\input locale

\def\localecurrchoice#1#2#3#4{#3}

\texosquerydefpattern{\sinumpattern}{%
  \sinumfmt
  {\decfmt{\#\,\#\#\#\,\#\#\#\,\#\#0}{\0\#\#\#\#\#\#\#\#}}%
  {\-\#\#\#\#\#\#\#\#0\0}%
}

\texosquerydefpattern{\curpattern}{%
\pmnumfmt
{\pcur{\decfmt{\#\,\#\#\#\,\#\#\#\,\#\#0}{\0\0\#\#\#\#\#\#\#\#}}{}}%
{\pcur{\decfmt{\#\,\#\#\#\,\#\#\#\,\#\#0}{\0\0\#\#\#\#\#\#\#\#}}{\-}}}

en-GB:
\localenumfmt{\sinumpattern}{1.2345E-3}.
\localenumfmt{\curpattern}{12345.6}.
\localenumfmt{\curpattern}{-12345.6}.
\localenumfmt{\curpattern}{0}.

\selectlocale{pt-BR}
pt-BR:
\localenumfmt{\sinumpattern}{1.2345E-3}.
\localenumfmt{\curpattern}{12345.6}.
\localenumfmt{\curpattern}{-12345.6}.
\localenumfmt{\curpattern}{0}.
```

```
\selectlocale{de-CH-1996}
de-CH-1996:
\localenumfmt{\sinumpattern}{1.2345E-3}.
\localenumfmt{\curpattern}{12345.6}.
\localenumfmt{\curpattern}{-12345.6}.
\localenumfmt{\curpattern}{0}.

\bye
```

This produces:

en-GB: 1.2345E−03. £12,345.60. −£12,345.60. £0.00.

pt-BR: 1,2345E−03. R$12.345,60. −R$12.345,60. R$0,00.

de-CH-1996: 1.2345E−03. SFr.12’345.60. −SFr.12’345.60. SFr.0.00.

Here’s a LaTeX alternative that redefines the formatting commands used by `\localenumfmt`:

```
\documentclass{article}

\usepackage{color}
\usepackage[main={en-GB},other={pt-BR,de-CH-1996}]{tex-locale}
\setmainfont{NimbusRoman-Regular}

\texosquerydefpattern{\sinumpattern}{%
  \sinumfmt
  {\decfmt{\#\,\#\#\#\,\#\#\#\,\#\#\0}{\0\#\#\#\#\#\#\#\#\#}}%
  {\-\#\#\#\#\#\#\#\#\0\0}%
}

\texosquerydefpattern{\curpattern}{%
\pmnumfmt
{\pcur{\decfmt{\#\,\#\#\#\,\#\#\#\,\#\#\0}{\0\0\#\#\#\#\#\#\#\#}}{}}%
{\pcur{\decfmt{\#\,\#\#\#\,\#\#\#\,\#\#\0}{\0\0\#\#\#\#\#\#\#\#}}{\-}}}

\renewcommand*{\localenumfmtpos}[1]{\textcolor{green}{#1}}
\renewcommand*{\localenumfmtneg}[1]{\textcolor{red}{#1}}
\renewcommand*{\localenumfmtzero}[1]{\textbf{#1}}

\begin{document}

en-GB:
\localenumfmt{\sinumpattern}{1.2345E-3}.
\localenumfmt{\curpattern}{12345.6}.
\localenumfmt{\curpattern}{-12345.6}.
```

```
\localenumfmt{\curpattern}{0}.

\selectlocale{pt-BR}
pt-BR:
\localenumfmt{\sinumpattern}{1.2345E-3}.
\localenumfmt{\curpattern}{12345.6}.
\localenumfmt{\curpattern}{-12345.6}.
\localenumfmt{\curpattern}{0}.

\selectlocale{de-CH-1996}
de-CH-1996:
\localenumfmt{\sinumpattern}{1.2345E-3}.
\localenumfmt{\curpattern}{12345.6}.
\localenumfmt{\curpattern}{-12345.6}.
\localenumfmt{\curpattern}{0}.
\end{document}
```

This produces:

en-GB: 1.2345E−03. £12,345.60. −£12,345.60. **£0.00**.

pt-BR: 1,2345E−03. R$12.345,60. −R$12.345,60. **R$0,00**.

de-CH-1996: 1.2345E−03. SFr.12’345.60. −SFr.12’345.60. **SFr.0.00**.

2.3.2 Date-Time Patterns

In addition to providing locale-sensitive strings for the current date-time, the `texosquery` application can also provide the locale’s pattern that’s used to describe how the dates and times should be formatted. Additionally, it can provide time-zone mappings from identifying labels to locale-sensitive names. Both these things require extra overhead and so these functions aren’t on by default.

You can enable them by defining

`\LocaleIfDateTimePatternsSupported`

```
\LocaleIfDateTimePatternsSupported{⟨true⟩}{⟨false⟩}
```

before you input `tex-locale.tex`. LaTeX users can use the more convenient timedata package option:

```
\usepackage[timedata]{tex-locale}
```

Plain TeX users need to define `\LocaleIfDateTimePatternsSupported` so that it does the ⟨*true*⟩ argument and ignores the ⟨*false*⟩ argument:

```
\def\LocaleIfDateTimePatternsSupported#1#2{#1}
\input locale
```

If enabled, tex-locale will add the `-M` and `-Z` switches to the `texosquery` command invocation. This will additionally redefine texosquery's pattern format commands (such as `\texosqueryfmtpatMMM`) to use the appropriate commands for the current locale. For example, `\texosqueryfmtpatMMM{⟨n⟩}` is redefined to use `\CurrentLocaleShortMonthName{⟨n⟩}`. (See the texosquery documentation for further details on these commands.)

If date-time patterns are supported then the current date-time data will be stored in

\LocaleDateTimeInfo

```
\LocaleDateTimeInfo
```

This will be empty if date-time patterns aren't supported.

The value of this command contains all the information for the current date-time provided in a format that can easily be parsed by date or time patterns. (This is done through texosquery's `\texosqueryfmtdatetime` command, which is internally used by `\LocaleApplyDateTimePattern`, described below.) For example, `\LocaleDateTimeInfo` might be set to

```
{1}{2017}{2017}{3}{12}{4}{85}{26}{4}{7}{1}{19}{19}{7}{7}{16}{39}{136}
{{1}{0}{Europe/London}{1}}
```

(Line break inserted above for clarity.) See the texosquery documentation for details of this syntax.

Alternatively you can use texosquery's pattern generator to create your own date-time pattern and explicitly use `\texosqueryfmtdatetime`. In which case `\LocaleDateTimeInfo` provides a convenient way of applying the pattern to the current date-time. For example:

```
\def\LocaleIfDateTimePatternsSupported#1#2{#1}
\def\LocaleMain{en-GB}
\input locale

\texosquerydefpattern{\pattern}{\%3E \%1d \%3M \%4y \%2H:\%2m:\%2s \%2z}

\ifx\LocaleDateTimeInfo\empty
\else
  \expandafter\texosqueryfmtdatetime\expandafter\pattern\LocaleDateTimeInfo
\fi
\bye
```

This produces:

> Sun 26 Mar 2017 21:38:06 BST

(assuming that was the date and time of the document build). The textual elements ("Sun", "Mar" and "BST") are obtained from the current locale.

If the date-time patterns are supported, you can apply a pattern to a specific date or time using

\LocaleApplyDateTimePattern

```
\LocaleApplyDateTimePattern{⟨dialect⟩}{⟨attribute⟩}{⟨data⟩}
```

This uses texosquery's \texosqueryfmtdatetime command to format a date or time according to a date-time pattern (identified by ⟨*attribute*⟩) for the locale identified by ⟨*dialect*⟩. The texosquery manual provides further details on this command, but essentially the first argument of \texosqueryfmtdatetime should be a control sequence containing the special pattern markup and the remaining arguments are the date-time data, which are supplied in the ⟨*data*⟩ argument of \LocaleApplyDateTimePattern.

The \LocaleDateTimeInfo command will either be empty or set to the correct data for the current date and time. This can be used in the ⟨*data*⟩ argument. Since \LocaleDateTimeInfo might be empty, \LocaleApplyDateTimePattern will first test for this and not try applying the pattern in that case. (Remember that you can also define your own custom date-time patterns, as mentioned in Section 2.1.)

Note that the ⟨*dialect*⟩ is used to fetch the pattern data, but the month and day names will be obtained using the \CurrentLocale... commands described in Section 2.7.1, which may not match ⟨*dialect*⟩. In general it's easiest to use the shortcut \CurrentLocaleApplyDateTimePattern (Section 2.7.3) instead to ensure the pattern matches the current locale.

The attributes associated with date-time patterns are listed below (see Section 2.2.3.2).

- fulldatefmt: the full date format pattern (as used in \LocaleFullDate).
- longdatefmt: the long date format pattern (as used in \LocaleLongDate).
- meddatefmt: the medium date format pattern (as used in \LocaleMediumDate).
- shortdatefmt: the short date format pattern (as used in \LocaleShortDate).
- fulltimefmt: the full time format pattern (as used in \LocaleFullTime).
- longtimefmt: the long time format pattern (as used in \LocaleLongTime).
- medtimefmt: the medium time format pattern (as used in \LocaleMediumTime).
- shorttimefmt: the short time format pattern (as used in \LocaleShortTime).
- fulldatetimefmt: the full date and time format pattern.
- longdatetimefmt: the long date and time format pattern.
- meddatetimefmt: the medium date and time format pattern.
- shortdatetimefmt: the short date and time format pattern.

For example:

```
\def\LocaleIfDateTimePatternsSupported#1#2{#1}
\def\LocaleMain{en-GB}

\input locale
```

```
Now:
\CurrentLocaleApplyDateTimePattern{fulldatetimefmt}{\LocaleDateTimeInfo}.
\bye
```

This displays:

> Now: Sunday, 26 March 2017 21:12:10 British Summer Time.

However, it's simpler to just use:

```
\def\LocaleIfDateTimePatternsSupported#1#2{#1}
\def\LocaleMain{en-GB}

\input locale

\def\localedatechoice#1#2#3#4{#1}
\def\localetimechoice#1#2#3#4{#1}

Now: \CurrentLocaleDateTime.
\bye
```

or with LaTeX:

```
\documentclass{article}

\usepackage[timedata,date=full,time=full,main={en-GB}]{tex-locale}

\begin{document}
Now: \CurrentLocaleDateTime.
\end{document}
```

The date-time patterns are therefore more useful when applied to something other that the current date-time. However, it's more complicated to work out the data.

For example:

```
\def\LocaleIfDateTimePatternsSupported#1#2{#1}
\def\LocaleMain{en-GB}

\input locale

\CurrentLocaleApplyDateTimePattern{fulldatetimefmt}%
{{1}{2017}{2017}{3}{12}{4}{85}{26}{4}{7}{1}{21}{21}{9}{9}{23}{36}{140}%
{{1}{0}{Europe/London}{1}}}
\bye
```

This produces:

> Sunday, 26 March 2017 21:23:36 British Summer Time.

2.4 Locale Information

Information about each locale is fetched for each tracked dialect (the main locale, identified by `\LocaleMain`, and the other locales, identified by `\LocaleOther`). This information can be accessed using the commands described below, where ⟨*dialect*⟩ is a tracklang dialect label that identifies the locale. There are convenient commands (`\CurrentLocale...`) that select the appropriate command with the label provided by the currently selected dialect. (See Section 2.7.)

Most of the commands in this package are intended to be used in an expandable context, and so will expand to nothing if the dialect isn't recognised.

`\LocaleMainDialect`

`\LocaleMainDialect`

This expands to the main locale's dialect label. For example with:

```
\def\LocaleMain{en-GB}
\input locale
```

the main dialect is `british`.

`\LocaleMainRegion`

`\LocaleMainRegion`

This expands to the main locale's region. For example, if the main locale is `en-GB`, then the main region is `GB`.

`\LocaleLanguageTag`

`\LocaleLanguageTag{`⟨*dialect*⟩`}`

Expands to the language tag for the given dialect. For example, if the document locales are set using:

```
\def\LocaleMain{en-GB}
\def\LocaleOther{pt-BR,fr-BE,de-CH-1996}
\input locale
```

then

```
\LocaleLanguageTag{british}
```

will expand to `en-GB` and

```
\LocaleLanguageTag{nswissgerman}
```

will expand to `de-CH-1996`.

`\LocaleLanguageName`

`\LocaleLanguageName{`⟨*dialect*⟩`}`

The name of the language associated with ⟨*dialect*⟩ provided in the Java virtual machine's default language.

\LocaleLanguageNativeName

```
\LocaleLanguageNativeName{⟨dialect⟩}
```

The name of the language associated with ⟨*dialect*⟩ provided in that language.

For example:

```
\font\nimbus="NimbusRoman-Regular" at 10pt
\nimbus

\def\LocaleMain{en-GB}
\def\LocaleOther{pt-BR,fr-BE}

\input locale

en-GB: \LocaleLanguageName{british}.
pt-BR: \LocaleLanguageName{brazilian}.
fr-BE: \LocaleLanguageName{belgique}.
\bye
```

For me this produces:

> en-GB: English. pt-BR: Portuguese. fr-BE: French.

because my operating system's default language is English. If I edit the `texosquery-jre8` bash script so that it includes

```
-Duser.language=fr
```

in the `java` arguments, then the same document will produce:

> en-GB: anglais. pt-BR: portugais. fr-BE: français.

So this isn't necessarily in the same language as the main locale (or any of the other locales tracked in the document). It's in the Java virtual machine's default language. Compare this document with:

```
\font\nimbus="NimbusRoman-Regular" at 10pt
\nimbus

\def\LocaleMain{en-GB}
\def\LocaleOther{pt-BR,fr-BE}

\input locale

en-GB: \LocaleLanguageNativeName{british}.
```

```
pt-BR: \LocaleLanguageNativeName{brazilian}.
fr-BE: \LocaleLanguageNativeName{belgique}.
\bye
```

This produces:

> en-GB: English. pt-BR: português. fr-BE: français.

So in this case each name is displayed according to its own language.

There are similar commands for the region and variant.

\LocaleRegionName

```
\LocaleRegionName{⟨dialect⟩}
```

The name of the region associated with ⟨*dialect*⟩ provided in the Java virtual machine's default language. This will be an empty string if the region wasn't supplied.

\LocaleRegionNativeName

```
\LocaleRegionNativeName{⟨dialect⟩}
```

The name of the region associated with ⟨*dialect*⟩ provided in that language. This will be an empty string if the region wasn't supplied.

\LocaleVariantName

```
\LocaleVariantName{⟨dialect⟩}
```

The name of the variant associated with ⟨*dialect*⟩ provided in the Java virtual machine's default language. This will be an empty string if a variant wasn't supplied.

\LocaleVariantNativeName

```
\LocaleVariantNativeName{⟨dialect⟩}
```

The name of the variant associated with ⟨*dialect*⟩ provided in that language. This will be an empty string if a variant wasn't supplied. This may well be the same as `\LocaleVariantName` as the variants are often identifiers (such as `1996` in the case of `de-CH-1996`) that remain the same in different languages.

You can find out if the language name, region name or variant have been set using the commands listed below. In each case, ⟨*dialect*⟩ is again the dialect label, ⟨*true*⟩ is the code to do if the condition is true and ⟨*false*⟩ is the code to do if the condition is false.

\LocaleIfHasLanguageName

```
\LocaleIfHasLanguageName{⟨dialect⟩}{⟨true⟩}{⟨false⟩}
```

This tests if the dialect has an associated language name. This will typically be true.

\LocaleIfHasRegionName

\LocaleIfHasRegionName{⟨*dialect*⟩}{⟨*true*⟩}{⟨*false*⟩}

This tests if the dialect has an associated region name. Some language tags don't have regions supplied. For example:

```
\def\LocaleMain{en}
\input locale
```

In this case, there's no region associated with the main locale.

\LocaleIfHasVariantName

\LocaleIfHasVariantName{⟨*dialect*⟩}{⟨*true*⟩}{⟨*false*⟩}

This tests if the dialect has an associated variant name. For example, there's no variant in `de-DE` but there is a variant in `de-DE-1996`.

2.5 Dates and Times

Date and time information is fetched for each tracked dialect. The `tex-locale.tex` code doesn't modify `\today` (although the `tex-locale.sty` LaTeX package can load datetime2, which does). Instead, the dates or times for a specific locale can be obtained using the commands listed below. The ⟨*dialect*⟩ argument indicates the tracklang dialect label. See Section 2.7 for shortcut commands that select the appropriate command with the label of the currently selected locale.

Dates and times are wrapped in a formatting command:

\localedatetimefmt

\localedatetimefmt{⟨*date-time text*⟩}

By default this simply does its argument. For example:

```
\def\LocaleMain{en-GB}

\input locale

\def\localedatetimefmt#1{{\it #1}}

Date: \LocaleFullDate{british}.

\bye
```

This displays the date in italic. The LaTeX equivalent is:

```
\documentclass{article}
\usepackage[main=en-GB]{tex-locale}
```

```
\renewcommand*{\localedatetimefmt}[1]{\textit{#1}}

\begin{document}
Date: \LocaleFullDate{british}.
\end{document}
```

2.5.1 Dates

\LocaleFullDate

```
\LocaleFullDate{⟨dialect⟩}
```

This displays the full date for the locale identified by ⟨*dialect*⟩ as provided by `texosquery`.

\LocaleLongDate

```
\LocaleLongDate{⟨dialect⟩}
```

This displays the long date for the locale identified by ⟨*dialect*⟩ as provided by `texosquery`.

\LocaleMediumDate

```
\LocaleMediumDate{⟨dialect⟩}
```

This displays the medium date for the locale identified by ⟨*dialect*⟩ as provided by `texosquery`.

\LocaleShortDate

```
\LocaleShortDate{⟨dialect⟩}
```

This displays the short date for the locale identified by ⟨*dialect*⟩ as provided by `texosquery`.

For example:

```
\def\LocaleMain{en-GB}

\input locale

Full: \LocaleFullDate{british}.
Long: \LocaleLongDate{british}.
Medium: \LocaleMediumDate{british}.
Short: \LocaleShortDate{british}.
\bye
```

In the above, the main locale is explicitly set by defining `\LocaleMain` before loading `tex-locale.tex`. The tracklang package identifies the `en-GB` locale with the label `british`.

Different locales have different concepts of full, long, medium and short dates. In most cases the short form is numeric and the full form is textual. The medium form may be numeric or it may use an abbreviated month name. The locale provided used by Java may also influence the result. For

example, if you are using texosquery-jre8 with java.locale.providers set to CLDR,JRE then the above document will display (assuming today is 2017-03-26):

> Full: Sunday, 26 March 2017. Long: 26 March 2017. Medium: 26 Mar 2017. Short: 26/03/2017.

However, if I instead use the Java 7 version (texosquery.jar) which doesn't support the CLDR, then the above document will display:

> Full: Sunday, 26 March 2017. Long: 26 March 2017. Medium: 26-Mar-2017. Short: 26/03/17.

The CLDR provides more extensive locale support than the JRE. In the case of en-GB, the difference is minor (as shown above), but in some cases a language may be supported in the CLDR but not in the JRE. For example, if in the above document I change the main locale to cy-GB (which maps to the tracklang dialect label GBwelsh):

```
\def\LocaleMain{cy-GB}

\input locale

Full: \LocaleFullDate{GBwelsh}.
Long: \LocaleLongDate{GBwelsh}.
Medium: \LocaleMediumDate{GBwelsh}.
Short: \LocaleShortDate{GBwelsh}.
\bye
```

then the Java 7 version produces US English dates:

> Full: Sunday, March 26, 2017. Long: March 26, 2017. Medium: Mar 26, 2017. Short: 3/26/17.

This is because Welsh isn't supported by the JRE, but it is supported by the CLDR, so the Java 8 version (texosquery-jre8) does work (provided java.locale.providers is set to CLDR,JRE):

> Full: Dydd Sul, 26 Mawrth 2017. Long: 26 Mawrth 2017. Medium: 26 Mawrth 2017. Short: 26/03/2017.

2.5.1.1 Week Days

In order to be compatible with pgfcalendar (and hence datetime2), the tex-locale package uses a zero-based indexing where 0 represents Monday, 1 represents Tuesday, etc. If you can't remember the index for a particular day, you can use the following commands:

\dtmMondayIndex

```
\dtmMondayIndex
```

This expands to 0, the index for Monday.

\dtmTuesdayIndex

```
\dtmTuesdayIndex
```

This expands to 1, the index for Tuesday.

\dtmWednesdayIndex

```
\dtmWednesdayIndex
```

This expands to 2, the index for Wednesday.

\dtmThursdayIndex

```
\dtmThursdayIndex
```

This expands to 3, the index for Thursday.

\dtmFridayIndex

```
\dtmFridayIndex
```

This expands to 4, the index for Friday.

\dtmSaturdayIndex

```
\dtmSaturdayIndex
```

This expands to 5, the index for Saturday.

\dtmSundayIndex

```
\dtmSundayIndex
```

This expands to 6, the index for Sunday.

You can get the day of week name identified by the Monday=0 based index for a particular dialect using

\LocaleDayName

```
\LocaleDayName{⟨dialect⟩}{⟨index⟩}
```

For example:

```
\def\LocaleMain{en-GB}
\def\LocaleOther{pt-BR}

\input locale

en-GB: \LocaleDayName{british}{0}.
pt-BR: \LocaleDayName{brazilian}{0}.
\bye
```

This produces:

en-GB: Monday. pt-BR: Segunda-feira.

The abbreviated day of week name can be obtained with:

\LocaleShortDayName

```
\LocaleShortDayName{⟨dialect⟩}{⟨index⟩}
```

This has the same syntax as the previous command. As with the date commands described in the previous section, the support for the given locale depends on the locale provider used by `texosquery`.

Some languages have a different form for day of week names when used in a standalone context, such as in a column header. These can be obtained using the commands below, *but only with `texosquery-jre8`*. If you are using Java 7 or earlier, these commands will produce identical results to the analogous command above.

\LocaleStandaloneDayName

```
\LocaleStandaloneDayName{⟨dialect⟩}{⟨index⟩}
```

The same syntax as above, this produces the standalone day of week name. The abbreviated name is produced with:

\LocaleStandaloneShortDayName

```
\LocaleStandaloneShortDayName{⟨dialect⟩}{⟨index⟩}
```

The first day of the week varies according to region. In some locales, Monday is considered the first day of the week (for example, en-GB), but in other locales, Sunday is the first day (for example, pt-BR). You can find out which day of the week is considered the first day using:

\LocaleFirstDayIndex

```
\LocaleFirstDayIndex{⟨dialect⟩}
```

This expands to an integer index identifying the day.

For example, Monday (0) is the first day of the week in `en-GB`, but Sunday (6) is the first day of the week in `pt-BR`.

```
\def\LocaleMain{en-GB}
\def\LocaleOther{pt-BR}

\input locale

First day (en-GB): \LocaleFirstDayIndex{british}.
First day (pt-BR): \LocaleFirstDayIndex{brazilian}.

\bye
```

So the above produces:

First day (en-GB): 0. First day (pt-BR): 6.

Since it's possible that you may want to use a different indexing system, there are commands provided to convert between them:

\LocaleDayIndexFromZeroMonToOneSun

```
\LocaleDayIndexFromZeroMonToOneSun{⟨index⟩}
```

This converts a day index from the Monday=0 based system to the Sunday=1 based system. For example, in Monday=0 indexing then the index 3 represents Thursday. In the Sunday=1 based system, the index 5 represents Thursday, so

```
\LocaleDayIndexFromZeroMonToOneSun{3}
```

expands to 5.

\LocaleDayIndexFromZeroMonToOneMon

```
\LocaleDayIndexFromZeroMonToOneMon{⟨index⟩}
```

This converts a day index from the Monday=0 (Sunday=6) based system to the ISO-8601 Monday=1 (Sunday=7) based system. For example

```
\LocaleDayIndexFromZeroMonToOneMon{3}
```

expands to 4.

\LocaleDayIndexFromOneSunToZeroMon

```
\LocaleDayIndexFromOneSunToZeroMon{⟨index⟩}
```

This converts a day index from the Sunday=1 based system to the Monday=0 based system. For example

```
\LocaleDayIndexFromOneSunToZeroMon{5}
```

expands to 3. (That is, it performs the reverse of `\LocaleDayIndexFromZeroMonToOneSun`.)

\LocaleDayIndexFromOneMonToZeroMon

```
\LocaleDayIndexFromOneMonToZeroMon{⟨index⟩}
```

This converts a day index from the ISO-8601 Monday=1 (Sunday=7) based system to the Monday=0 (Sunday=6) based system. For example

```
\LocaleDayIndexFromOneMonToZeroMon{4}
```

expands to 3. (That is, it performs the reverse of \LocaleDayIndexFromZeroMonToOneMon.)

\LocaleDayIndexFromRegion

\LocaleDayIndexFromRegion{⟨*dialect*⟩}{⟨*index*⟩}

This converts the region's day of the week (starting from 1 for the region's first day) to Monday=0 based indexing. The region is identified by ⟨*dialect*⟩. This first uses \LocaleFirstDayIndex {⟨*dialect*⟩} to find the 1-based first day of week index for the region and then converts it to the Monday=0 based system. An invalid ⟨*index*⟩ results in -1.

For example, the en-GB locale has Monday as the first day of the week. So if the ⟨*index*⟩ argument is 1

```
\LocaleDayIndexFromRegion{british}{1}
```

that indicates Monday. This is then converted to 0 (using \LocaleDayIndexFromOneMonToZeroMon). If the ⟨*index*⟩ argument is 7

```
\LocaleDayIndexFromRegion{british}{7}
```

that indicates Sunday, so this would be converted to 6.

The pt-BR locale has Sunday as the first day of the week. So if the ⟨*index*⟩ argument is 2

```
\LocaleDayIndexFromRegion{brazilian}{2}
```

that indicates Monday. This is converted to 0 (using \LocaleDayIndexFromOneSunToZeroMon). If the ⟨*index*⟩ argument is 1

```
\LocaleDayIndexFromRegion{brazilian}{1}
```

that indicates Sunday, so this would be converted to 6.

\LocaleDayIndexToRegion

\LocaleDayIndexToRegion{⟨*dialect*⟩}{⟨*index*⟩}

This performs the reverse operation. In this case the ⟨*index*⟩ argument is Monday=0 based and the result is the locale's day index (starting with 1 for the first day of the week).

2.5.1.2 Month Names

The month name for a given locale can be obtained using

\LocaleMonthName

\LocaleMonthName{⟨*dialect*⟩}{⟨*index*⟩}

where ⟨*dialect*⟩ is the tracklang dialect label that identifies the locale and ⟨*index*⟩ is an integer from 1 (January) to 12 (December) indicating the month.

For example:

```
\def\LocaleMain{en-GB}
\def\LocaleOther{pt-BR}

\input locale

en-GB: \LocaleMonthName{british}{1}.
pt-BR: \LocaleMonthName{brazilian}{1}.

\bye
```

produces:

en-GB: January. pt-BR: Janeiro.

The abbreviated month name can be obtained using

\LocaleShortMonthName

```
\LocaleShortMonthName{⟨dialect⟩}{⟨index⟩}
```

which has the same syntax as the previous command.

Some languages have a different form for month names when used in a standalone context, such as in a column header. These can be obtained using the commands below, *but only with texosquery-jre8*. If you are using Java 7 or earlier, these commands will produce identical results to the analogous command above.

\LocaleStandaloneMonthName

```
\LocaleStandaloneMonthName{⟨dialect⟩}{⟨index⟩}
```

The same syntax as above, this produces the standalone month name. The abbreviated name is produced with:

\LocaleStandaloneShortMonthName

```
\LocaleStandaloneShortMonthName{⟨dialect⟩}{⟨index⟩}
```

2.5.2 Times

\LocaleFullTime

```
\LocaleFullTime{⟨dialect⟩}
```

This displays the full time for the locale identified by ⟨*dialect*⟩ as provided by `texosquery`.

\LocaleLongTime

```
\LocaleLongTime{⟨dialect⟩}
```

This displays the long time for the locale identified by ⟨*dialect*⟩ as provided by `texosquery`.

\LocaleMediumTime

```
\LocaleMediumTime{⟨dialect⟩}
```

This displays the medium time for the locale identified by ⟨*dialect*⟩ as provided by `texosquery`.

\LocaleShortTime

```
\LocaleShortTime{⟨dialect⟩}
```

This displays the short time for the locale identified by ⟨*dialect*⟩ as provided by `texosquery`.

As with dates, the results are determined by the locale provider used by `texosquery`. For example, the Java 8 variant (`texosquery-jre8`) with `java.locale.providers` set to `CLDR,JRE` may produce a different result to a variant of `texosquery` that only uses the JRE provider.

For example:

```
\def\LocaleMain{en-GB}

\input locale

Full: \LocaleLongTime{british}.
Long: \LocaleLongTime{british}.
Medium: \LocaleMediumTime{british}.
Short: \LocaleShortTime{british}.

\bye
```

This produces:

> Full: 13:43:53 BST. Long: 13:43:53 BST. Medium: 13:43:53. Short: 13:43.

In this particular case there's no difference between using the CLDR and the JRE locale providers, but note that there's no difference in the full and long forms.

There are also commands for the combined date and time:

\LocaleFullDateTime

```
\LocaleFullDateTime{⟨dialect⟩}
```

This displays the full date and time for the locale identified by ⟨*dialect*⟩ as provided by `texosquery`.

\LocaleLongDateTime

```
\LocaleLongDateTime{⟨dialect⟩}
```

This displays the long date and time for the locale identified by ⟨*dialect*⟩ as provided by `texosquery`.

\LocaleMediumDateTime

```
\LocaleMediumDateTime{⟨dialect⟩}
```

This displays the medium date and time for the locale identified by ⟨*dialect*⟩ as provided by `texosquery`.

\LocaleShortDateTime

```
\LocaleShortDateTime{⟨dialect⟩}
```

This displays the short date and time for the locale identified by ⟨*dialect*⟩ as provided by `texosquery`.

2.6 Numbers

Different locales use different symbols to denote the decimal mark or the number group separator when displaying numbers. There are other numeric-related symbols that may also vary according to region, such as the currency symbol, exponent, percent or per mill signs. Since most of these are outside of the Basic Latin set, the examples here use XeTeX with a font that supports those symbols. If you don't have that font installed, you will need to adapt the examples accordingly.

As with date and times, the commands describe here need the tracklang dialect label that identifies the required locale. See Section 2.7 for shortcut commands that automatically select the current locale's dialect label.

In general, most of these commands won't need to be used explicitly as it's easier to use a numeric pattern instead (see Section 2.3.1).

2.6.1 Numeric Symbols

The number group separator for a particular locale can be obtained using:

\LocaleNumericGroupSep

```
\LocaleNumericGroupSep{⟨dialect⟩}
```

where ⟨*dialect*⟩ is the tracklang dialect label representing the locale.

You can determine whether or not the locale uses a number group separator using:

\LocaleIfNumericUsesGroup

```
\LocaleIfNumericUsesGroup{⟨dialect⟩}{⟨true code⟩}{⟨false code⟩}
```

This does ⟨*true code*⟩ if the locale identified by ⟨*dialect*⟩ uses a number group separator otherwise it does ⟨*false code*⟩.

The decimal mark is obtained using:

\LocaleNumericDecimalSep

```
\LocaleNumericDecimalSep{⟨dialect⟩}
```

The monetary separator is often the same as the decimal separator, but isn't always, so there's a separate command for it:

\LocaleNumericMonetarySep

```
\LocaleNumericMonetarySep{⟨dialect⟩}
```

The exponent symbol is obtained using:

\LocaleNumericExponent

```
\LocaleNumericExponent{⟨dialect⟩}
```

The percent symbol is obtained using:

\LocaleNumericPercent

```
\LocaleNumericPercent{⟨dialect⟩}
```

This will typically use \% (the percent symbol %).

The per mill symbol is obtained using:

\LocaleNumericPermill

```
\LocaleNumericPermill{⟨dialect⟩}
```

This will typically be a character outside the Basic Latin set, so the document will need to support this symbol if required.

2.6.2 Currencies

The official currency code (such as `GBP` or `USD`) is obtained using:

\LocaleCurrencyLabel

```
\LocaleCurrencyLabel{⟨dialect⟩}
```

This first checks if the `currency` attribute for the given dialect is `XXX` (which denotes an unknown currency, typically because there's no region associated with ⟨*dialect*⟩). If the currency for ⟨*dialect*⟩ is known (that is, the attribute value isn't `XXX`), then that currency's official code is produced. If the currency is unknown, then it will try to fallback on the main dialect's currency code if the main dialect has an associated region, otherwise it will fallback on the OS currency code.

Remember that with the language tag, you're not restricted to using official languages for a given region. For example, if you're writing in English in Belgium, it's valid to use `en-BE` as a locale. For example:

```
\def\LocaleMain{en-GB}
\def\LocaleOther{en-BE,pt-BR}

\input locale
```

```
en-GB: \LocaleCurrencyLabel{british}.
en-BE: \LocaleCurrencyLabel{enBE}.
pt-BR: \LocaleCurrencyLabel{brazilian}.

\bye
```

This produces:

> en-GB: GBP. en-BE: EUR. pt-BR: BRL.

There are a few currencies that have an unofficial currency code. These are typically currencies pegged to another currency with a fixed exchange rate of 1.0. The official ISO currency code for one region may be the code for the other currency. For example, the Isle of Man currency is the Manx pound, which is kept in parity with pound sterling. The currency code for `en-IM` is returned as `GBP` by Java (since IMP has no official recognition in ISO 4217), but `texosquery` recognises that this region has an unofficial currency code `IMP`. This can be obtained using

`\LocaleCurrencyRegionalLabel`

```
\LocaleCurrencyRegionalLabel{⟨dialect⟩}
```

This command is much the same as the previous command in that it will fallback on the main or OS currency code if not known. The dialect attribute `regionalcurrency` is queried for the required information. For most regions, this command will return the same as `\LocaleCurrencyLabel`.

For example:

```
\def\LocaleMain{en-GB}
\def\LocaleOther{en-IM,pt-BR}

\input locale

en-GB: \LocaleCurrencyRegionalLabel{british}.
en-IM: \LocaleCurrencyRegionalLabel{isleofmanenglish}.
pt-BR: \LocaleCurrencyRegionalLabel{brazilian}.

\bye
```

This produces:

> en-GB: GBP. en-IM: IMP. pt-BR: BRL.

If I had used `\LocaleCurrencyLabel` instead, the `en-IM` currency label would've been displayed as GBP.

The currency symbol (for example, $ or £) is obtained using

`\LocaleCurrencySymbol`

\LocaleCurrencySymbol{⟨*dialect*⟩}

Except in the case of $ this will include symbols outside the Basic Latin set. This means that if you use this command in your document, you need to ensure that the document encoding has been set up before `tex-locale.tex` is loaded. (The LaTeX `tex-locale.sty` package can automatically load fontspec or inputenc & fontenc, if required.) As with the previous commands, if the currency is unknown it will try to fallback on the main or OS currency. If the `currencysym` dialect attribute is the same as the currency code, then the `sym` currency attribute will be used instead.

Modifying the above example:

```
\font\nimbus="NimbusRoman-Regular" at 10pt
\nimbus

\def\LocaleMain{en-GB}
\def\LocaleOther{en-IM,pt-BR}

\input locale

en-GB: \LocaleCurrencySymbol{british}.
en-IM: \LocaleCurrencySymbol{isleofmanenglish}.
pt-BR: \LocaleCurrencySymbol{brazilian}.

\bye
```

This produces:

en-GB: £. en-IM: M£. pt-BR: R$.

The XeLaTeX equivalent is:

```
\documentclass{article}

\usepackage[main={en-GB},other={en-IM,pt-BR}]{tex-locale}
\setmainfont{NimbusRoman-Regular}

\begin{document}
en-GB: \LocaleCurrencySymbol{british}.
en-IM: \LocaleCurrencySymbol{isleofmanenglish}.
pt-BR: \LocaleCurrencySymbol{brazilian}.
\end{document}
```

Alternatively, you can use:

\LocaleCurrencyTeXSymbol

\LocaleCurrencyTeXSymbol{⟨*dialect*⟩}

This works in the same way as the previous command, except that it checks the `currencytex` dialect attribute. If this attribute is the same as the currency code, then the `tex` currency attribute is used instead.

This command uses control sequences instead of the actual currency character. Since both texosquery and tex-locale are designed for generic TEX use, texosquery just performs some limited tests for the existence of common command names (such as `\pounds` or `\euro`). If it can't find an appropriate command to use, the currency commands simply expand to a textual tag.

For example, when `texosquery.tex` is input, it checks for the existence of `\faGbp` and `\pounds`. If one of those commands exist, then `\texosquerycurrencypound` is defined to that commands, otherwise it's defined to just "pound". For the Euro symbol (`\texosquerycurrencyeuro`) the code checks for `\faEuro`, `\texteuro` and `\euro`. (See the texosquery package for more information about these currency commands.)

The LATEX `tex-locale.sty` package will automatically load the textcomp package by default. You can switch this to another package (for example, fontawesome) using the symbols package option (for example, symbols=fontawesome).

2.7 Current Locale

The commands described above mostly require a recognised tracklang dialect label in the argument to identify the locale. These labels aren't intuitively obvious. There are some predefined dialect labels that try to be compatible with known babel dialects, but neither babel nor polyglossia provide as much detail about the non-language aspects of the dialect (such as the region or variant). For example, there's no babel or polyglossia setting for English in the Isle of Man (`en-IM`). Users need to select the closest matching dialect (`british` in the case of `en-IM`). The tracklang package also allows an unofficial language and region combination. For example, a document written in English but with the regional information, such as currencies, matching those for Belgium, should be identified with `en-BE`. This isn't recognised as a predefined tracklang dialect, so tracklang creates its own dialect label (`enBE` in this case) when it parses this language tag.

This makes it a bit awkward to directly use the above commands and it's most likely that the required dialect will be the currently selected language setting, so `tex-locale.tex` provides some convenient wrapper commands, listed below. These `\CurrentLocale...` commands are redefined every time the locale is changed (except for `\CurrentLocaleDateTime`). For a document that doesn't use babel or polyglossia, these commands can be reset using `\selectlocale` (described on page 12). If `tex-locale.tex` detects any language hooks `\captions`⟨*language*⟩ (such as `\captionsenglish` or `\captionsbritish`) then code is added to those hooks to ensure that the locale is switched when the language changes.

TEX's `\show` command provides a useful way of checking the current settings. For example:

```
\def\LocaleMain{en-GB}
\def\LocaleOther{en-IM,pt-BR}

\input locale
```

```
\show\CurrentLocaleMonthName

\selectlocale{en-IM}
\show\CurrentLocaleMonthName

\selectlocale{pt-BR}
\show\CurrentLocaleMonthName

\bye
```

These cause three interruptions to the TEX build, which look like errors but are the results from each `\show` command. In the first case the transcript shows:

```
> \CurrentLocaleMonthName=macro:
->\LocaleMonthName {british}.
```

I haven't specifically set the locale at this point. At the end of the `tex-locale.tex` code, the main locale was automatically selected. So the current locale dialect label is `british` and `\CurrentLocaleMonthNam` has been defined as

```
\LocaleMonthName{british}
```

Note that there's no check for `\languagename` or similar command here. The dialect label is hard-coded into the definition of `\CurrentLocaleMonthName`, so it's fully-expandable (unless the month name contains any awkward non-expandable characters, which can occur with inputenc and non-ASCII characters).

The next instance of `\show` produces the following lines in the transcript:

```
> \CurrentLocaleMonthName=macro:
->\LocaleMonthName {isleofmanenglish}.
```

The command `\CurrentLocaleMonthName` was redefined when `\selectlocale{en-IM}` was used.

Similarly for the final instance of `\show`:

```
> \CurrentLocaleMonthName=macro:
->\LocaleMonthName {brazilian}.
```

All the commands that are redefined with each instance of `\selectlocale` are listed below. Note that `\selectlocale` also uses tracklang's `\SetCurrentTrackedDialect`, which also defines a set of commands that can be used to identify information about the current dialect. (See the tracklang documentation for further details.)

`\CurrentLocaleLanguageName`

```
\CurrentLocaleLanguageName
```

This is a shortcut for `\LocaleLanguageName{`⟨*dialect*⟩`}`.

`\CurrentLocaleLanguageNativeName`

```
\CurrentLocaleLanguageNativeName
```

This is a shortcut for `\LocaleLanguageNativeName{`⟨*dialect*⟩`}`.

`\CurrentLocaleRegionName`

```
\CurrentLocaleRegionName
```

This is a shortcut for `\LocaleRegionName{`⟨*dialect*⟩`}`.

`\CurrentLocaleRegionNativeName`

```
\CurrentLocaleRegionNativeName
```

This is a shortcut for `\LocaleRegionNativeName{`⟨*dialect*⟩`}`.

`\CurrentLocaleVariantName`

```
\CurrentLocaleVariantName
```

This is a shortcut for `\LocaleVariantName{`⟨*dialect*⟩`}`.

`\CurrentLocaleVariantNativeName`

```
\CurrentLocaleVariantNativeName
```

This is a shortcut for `\LocaleRegionNativeName{`⟨*dialect*⟩`}`.

2.7.1 Dates and Times

`\CurrentLocaleFirstDayIndex`

```
\CurrentLocaleFirstDayIndex
```

This is a shortcut for `\LocaleFirstDayIndex{`⟨*dialect*⟩`}`.

`\CurrentLocaleDayIndexFromRegion`

```
\CurrentLocaleDayIndexFromRegion{⟨index⟩}
```

This is a shortcut for `\LocaleDayIndexFromRegion{`⟨*dialect*⟩`}{`⟨*index*⟩`}`.

`\CurrentLocaleDayName`

```
\CurrentLocaleDayName{⟨index⟩}
```

This is a shortcut for `\LocaleDayName{`⟨*dialect*⟩`}{`⟨*index*⟩`}`.

`\CurrentLocaleShortDayName`

```
\CurrentLocaleShortDayName{⟨index⟩}
```

This is a shortcut for \LocaleShortDayName{⟨*dialect*⟩}{⟨*index*⟩}.

\CurrentLocaleStandaloneDayName

\CurrentLocaleStandaloneDayName{⟨*index*⟩}

This is a shortcut for \LocaleStandaloneDayName{⟨*dialect*⟩}{⟨*index*⟩}.

\CurrentLocaleStandaloneShortDayName

\CurrentLocaleStandaloneShortDayName{⟨*index*⟩}

This is a shortcut for \LocaleStandaloneShortDayName{⟨*dialect*⟩}{⟨*index*⟩}.

\CurrentLocaleMonthName

\CurrentLocaleMonthName{⟨*index*⟩}

This is a shortcut for \LocaleMonthName{⟨*dialect*⟩}{⟨*index*⟩}.

\CurrentLocaleShortMonthName

\CurrentLocaleShortMonthName{⟨*index*⟩}

This is a shortcut for \LocaleShortMonthName{⟨*dialect*⟩}{⟨*index*⟩}.

\CurrentLocaleStandaloneMonthName

\CurrentLocaleStandaloneMonthName{⟨*index*⟩}

This is a shortcut for \LocaleStandaloneMonthName{⟨*dialect*⟩}{⟨*index*⟩}.

\CurrentLocaleStandaloneShortMonthName

\CurrentLocaleStandaloneShortMonthName{⟨*index*⟩}

This is a shortcut for \LocaleStandaloneShortMonthName{⟨*dialect*⟩}{⟨*index*⟩}.

\CurrentLocaleDate

\CurrentLocaleDate

This is slightly more complicated that the above. It uses

\localedatechoice

\localedatechoice{⟨*full*⟩}{⟨*long*⟩}{⟨*medium*⟩}{⟨*short*⟩}

to determine whether to use the full date \LocaleFullDate{⟨*dialect*⟩}, the long date \LocaleLongDate {⟨*dialect*⟩}, the medium date \LocaleMediumDate{⟨*dialect*⟩} or the short date \LocaleShortDate {⟨*dialect*⟩}.

This command may be defined before tex-locale.tex is loaded or redefined afterwards. For example, to ensure that \CurrentLocaleDate uses the medium form:

```
\def\localedatechoice#1#2#3#4{#3}
\input locale
```

LaTeX users should instead use the date package option in the `tex-locale.sty` package interface:

```
\usepackage[date=medium]{tex-locale}
```

LaTeX users can also redefine the command afterwards. For example:

```
\renewcommand*{\localedatechoice}[4]{#3}
```

There is a similar command for the time:

\CurrentLocaleTime

```
\CurrentLocaleTime
```

This uses

\localetimechoice

```
\localetimechoice{⟨full⟩}{⟨long⟩}{⟨medium⟩}{⟨short⟩}
```

to determine whether to use the full time `\LocaleFullTime{⟨dialect⟩}`, the long time `\LocaleLongTime {⟨dialect⟩}`, the medium time `\LocaleMediumTime{⟨dialect⟩}` or the short time `\LocaleShortTime {⟨dialect⟩}`. As with the date choice, this command may be defined by plain TeX users before `tex-locale.tex` is input. LaTeX users should use the time package option instead. In both formats, the command may be redefined after tex-locale has been loaded.

There is also a command for the combined date and time:

\CurrentLocaleDateTime

```
\CurrentLocaleDateTime
```

This doesn't use the analogous `\LocaleFullDateTime` etc commands, but instead is simply defined as:

```
\CurrentLocaleDate\space\CurrentLocaleTime
```

This allows for a mix of date and time styles, reflecting the definitions of `\localedatechoice` and `\localetimechoice`.

If you want a specific style (ignoring `\localedatechoice` and `\localetimechoice`), you can use the following commands:

\CurrentLocaleFullDate

```
\CurrentLocaleFullDate
```

This just uses `\LocaleFullDate{⟨dialect⟩}`.

\CurrentLocaleLongDate

```
\CurrentLocaleLongDate
```

This just uses \LocaleLongDate{⟨*dialect*⟩}.

\CurrentLocaleMediumDate

```
\CurrentLocaleMediumDate
```

This just uses \LocaleMediumDate{⟨*dialect*⟩}.

\CurrentLocaleShortDate

```
\CurrentLocaleShortDate
```

This just uses \LocaleShortDate{⟨*dialect*⟩}.

\CurrentLocaleFullTime

```
\CurrentLocaleFullTime
```

This just uses \LocaleFullTime{⟨*dialect*⟩}.

\CurrentLocaleLongTime

```
\CurrentLocaleLongTime
```

This just uses \LocaleLongTime{⟨*dialect*⟩}.

\CurrentLocaleMediumTime

```
\CurrentLocaleMediumTime
```

This just uses \LocaleMediumTime{⟨*dialect*⟩}.

\CurrentLocaleShortTime

```
\CurrentLocaleShortTime
```

This just uses \LocaleShortTime{⟨*dialect*⟩}.

\CurrentLocaleFullDateTime

```
\CurrentLocaleFullDateTime
```

This just uses \LocaleFullDateTime{⟨*dialect*⟩}.

\CurrentLocaleLongDateTime

```
\CurrentLocaleLongDateTime
```

This just uses \LocaleLongDateTime{⟨*dialect*⟩}.

\CurrentLocaleMediumDateTime

```
\CurrentLocaleMediumDateTime
```

This just uses \LocaleMediumDateTime{⟨*dialect*⟩}.

\CurrentLocaleShortDateTime

```
\CurrentLocaleShortDateTime
```

This just uses \LocaleShortDateTime{⟨*dialect*⟩}.

2.7.2 Numeric Symbols

The currency for the current locale is similar in construct to the current locale's date and time commands listed above.

\CurrentLocaleCurrency

```
\CurrentLocaleCurrency
```

This uses

\localecurrchoice

```
\localecurrchoice{⟨label⟩}{⟨regional⟩}{⟨symbol⟩}{⟨TEX⟩}
```

to determine whether to use \LocaleCurrencyLabel{⟨*dialect*⟩}, \LocaleCurrencyRegionalLabel{⟨*dialect*⟩}, \LocaleCurrencySymbol{⟨*dialect*⟩} or \LocaleCurrencyTeXSymbol{⟨*dialect*⟩}. This can similarly be defined before `tex-locale.tex` is input or redefined afterwards. LaTeX users can use the currency package option.

\CurrentLocaleNumericGroupSep

```
\CurrentLocaleNumericGroupSep
```

This is a shortcut for \LocaleNumericGroupSep{⟨*dialect*⟩}.

\CurrentLocaleIfNumericUsesGroup

```
\CurrentLocaleIfNumericUsesGroup{⟨true code⟩}{⟨false code⟩}
```

This is a shortcut for \LocaleIfNumericUsesGroup{⟨*dialect*⟩}{⟨*true code*⟩}{⟨*false code*⟩}.

The next few commands have a slightly different pattern to the shortcut control sequence name. For brevity, the shortcut command omits the "Numeric" part of the corresponding name.

\CurrentLocaleDecimalSep

```
\CurrentLocaleDecimalSep
```

This is a shortcut for \LocaleNumericDecimalSep{⟨*dialect*⟩}.

\CurrentLocaleMonetarySep

```
\CurrentLocaleMonetarySep
```

This is a shortcut for `\LocaleNumericMonetarySep{⟨dialect⟩}`.

`\CurrentLocaleExponent`

```
\CurrentLocaleExponent
```

This is a shortcut for `\LocaleNumericExponent{⟨dialect⟩}`.

2.7.3 Current Locale Patterns

`\CurrentLocaleIntegerPattern`

```
\CurrentLocaleIntegerPattern
```

This is a convenient shortcut to access the integer pattern for the current dialect.

`\CurrentLocaleDecimalPattern`

```
\CurrentLocaleDecimalPattern
```

This is a convenient shortcut to access the decimal pattern for the current dialect.

`\CurrentLocaleCurrencyPattern`

```
\CurrentLocaleCurrencyPattern
```

This is a convenient shortcut to access the currency pattern for the current dialect.

`\CurrentLocalePercentPattern`

```
\CurrentLocalePercentPattern
```

This is a convenient shortcut to access the percent pattern for the current dialect.

`\CurrentLocaleApplyDateTimePattern`

```
\CurrentLocaleApplyDateTimePattern
```

This is a convenient shortcut for `\LocaleApplyDateTimePattern{⟨dialect⟩}`.

3 LaTeX Use

The tex-locale package is loaded in LaTeX with the usual `\usepackage` syntax:

```
\usepackage{tex-locale}
```

This does more than simply input `tex-locale.tex`. The options are listed below.

main=`{`⟨***tag***⟩`}` This option identifies the main locale for the document. The value should be a valid language tag. If this option is omitted, the main language will be obtained from the Java Runtime Environment, which should match your operating system's default locale. If any languages have already been tracked in your document with tracklang prior to loading tex-locale, the main language is set to the first tracked dialect. Normally you don't need to explicitly load tracklang.

other=`{`⟨***list***⟩`}` This option identifies other locales required by the document. The value should be a comma-separated list of language tags or the keyword locale to indicate the system's default locale. The value must be grouped to protect any commas from the key=value parser.

> This option has a cumulative effect.

symbols=`{`⟨***name***⟩`}` This option identifies the symbol package to automatically load. For example, symbols=textcomp or symbols=fontawesome. If another package is required, texosquery's currency symbol commands will need to be redefined as appropriate.

The keyword none indicates that no package is required. The default is symbols=none for XeLaTeX or LuaLaTeX, otherwise it's symbols=textcomp.

support=`{`⟨***value***⟩`}` This option identifies the language support value. Available values are:

- none: don't load any language support package;
- auto: load polyglossia for XeLaTeX or LuaLaTeX, otherwise load babel;
- babel: load babel regardless of the LaTeX format;
- polyglossia: load polyglossia without checking the LaTeX format (but remember that polyglossia doesn't work with PDFLaTeX);
- cjk: load CJK support with either xeCJK (XeLaTeX/LuaLaTeX) or CJKutf8. This isn't fully tested, and there's no support for non-UTF encoding.

fontenc=`{`⟨***value***⟩`}` This option indicates whether or not to load the fontenc package. The value should be one of: none (don't load), auto (automatically load with the encoding determined from the language settings), or ⟨*option*⟩ (indicating the package option to pass to fontenc). The default is fontenc=auto unless fontenc has already been loaded.

This option is ignored with XeLaTeX or LuaLaTeX.

inputenc=`{`⟨***value***⟩`}` This option indicates whether or not to load the inputenc package. The value should be one of: none (don't load), auto (automatically load with the encoding determined from the default obtained from texosquery), or ⟨*option*⟩ (indicating the package option to pass to inputenc). The default is inputenc=auto but inputenc won't be loaded if the appropriate encoding can't be determined. Note that loading inputenc before the texosquery lookup is performed will cause a problem with non-ASCII characters appearing in the result.

This option is ignored with XeLaTeX or LuaLaTeX.

datetime=⟨***value***⟩ This option indicates whether or not to load the datetime2 package. The value may be one of:

- false: don't load datetime2;
- iso: load datetime2 with the package option useregional=false and sets the date-time style to iso;
- text: load datetime2 with the package option useregional=text;
- num or numeric: load datetime2 with the package option useregional=numeric;
- locale: don't load datetime2 and set `\today` to `\CurrentLocaleDate`, if available.

date=⟨***value***⟩ This option indicates the preferred date style for `\CurrentLocaleDate`. The value may be one of: full, long, medium short. This option doesn't affect `\today` unless you have used datetime=locale.

time=⟨***value***⟩ This option indicates the preferred time style for `\CurrentLocaleTime`. The value may be one of: full, long, medium short.

timedate=⟨***boolean***⟩ This option indicates whether or not to support date and time patterns. (See Section 2.3.2.)

currency=⟨***value***⟩ This option indicates the preferred currency style for `\CurrentLocaleCurrency`. Available values: official (official identifier), unofficial unofficial identified, sym (the symbol, which may include non-ASCII characters) or tex (use the commands provided by texosquery). The default is currency=sym for XeLaTeX/LuaLaTeX or currency=tex otherwise.

The tex-locale package will use texosquery to lookup the default language tag and encoding, and then attempt to load the appropriate encoding and language support packages.

For example, consider the following document:

```
\documentclass{article}
\usepackage{tex-locale}
```

```
\begin{document}
Currency: \CurrentLocaleCurrency.
\end{document}
```

My default locale is en-GB with UTF-8 as the default file encoding, so if I compile this document with PDFLaTeX, then this is essentially equivalent to:

```
\documentclass{article}
\usepackage[en-GB]{tracklang}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[british]{babel}
\usepackage[useregional]{datetime2}
\begin{document}
Currency: \pounds.
\end{document}
```

(Other dependent packages omitted for clarity.) Whereas if I compile the document with XeLaTeX, then this is essentially equivalent to:

```
\documentclass{article}
\usepackage[en-GB]{tracklang}
\usepackage{fontspec}
\usepackage{polyglossia}
\setmainlanguage[variant=uk]{english}
\usepackage[useregional]{datetime2}
\begin{document}
Currency: £.
\end{document}
```

If I still want to use babel, then I can enforce this with:

```
\usepackage[support=babel]{tex-locale}
```

Or if I don't want babel or polyglossia:

```
\usepackage[support=none]{tex-locale}
```

You can determine which language package was used with:

\LocaleSupportPackageCase

```
\LocaleSupportPackageCase{⟨babel⟩}{⟨polyglossia⟩}{⟨neither⟩}
```

If either package has been loaded \selectlocale{⟨*locale*⟩} should automatically be implemented when the document language changes, so you can the use the current locale commands described in Section 2.7.

Example:

```
\usepackage[main={sr-Cyrl-RS},other={en-GB}]{tex-locale}

\LocaleSupportPackageCase
{\newcommand{\textenglish}[1]{\foreignlanguage{british}{#1}}}% babel
{% polyglossia
  \setmainfont{Liberation Serif}
  \newfontfamily\cyrillicfont{Liberation Serif}
}
{\newcommand{\textenglish}[1]{#1}}% none
```

4 The Code

4.1 LaTeX Code (tex-locale.sty)

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{tex-locale}[2018/08/26 v1.0 (NLCT) localisation support]
```

Make life easier by using etoolbox:

```
\RequirePackage{etoolbox}
```

Also need xfor to break out of `\@for` loop.

```
\RequirePackage{xfor}
```

Load tracklang using package interface just in case any languages have been passed through the document class options. Require at least v1.3.3 since the dialect label mappings are needed in this file.

```
\RequirePackage{tracklang}[2016/11/03]
```

If any languages have been tracked, set the main language to the first tracked dialect.

```
\AnyTrackedLanguages
{%
   \ForEachTrackedDialect{\locale@this@dialect}
   {
      \ifx\LocaleMain\undefined
        \edef\LocaleMain{\GetTrackedLanguageTag{\locale@this@dialect}}
        \let\@locale@trackedmain\LocaleMain
      \else
        \ifx\LocaleOther\undefined
          \edef\LocaleOther{\GetTrackedLanguageTag{\locale@this@dialect}}
        \else
          \edef\LocaleOther{\LocaleOther,\GetTrackedLanguageTag{\locale@this@dialect}
        \fi
      \fi
   }
}
{}
```

Need to determine if we're using XeLaTeX or LuaLaTeX.

```
\RequirePackage{ifxetex}
\RequirePackage{ifluatex}
```

`\@locale@ifxeorlua` Short cut to check if we're using either:

```
\ifxetex
```

```
    \newcommand*{\@locale@ifxeorlua}[2]{#1}
  \else
    \ifluatex
      \newcommand*{\@locale@ifxeorlua}[2]{#1}
    \else
      \newcommand*{\@locale@ifxeorlua}[2]{#2}
    \fi
  \fi
```

Need xkeyval for key-value interface:

```
\RequirePackage{xkeyval}
```

Define package options.

Each `\DeclareOptionX` needs a corresponding `\DeclareOption` so that it can be passed as a document class option, so define a command that will implement both.

ale@declareoption

```
\newcommand*{\@locale@declareoption}[2]{%
  \DeclareOptionX{#1}{#2}%
  \DeclareOption{#1}{#2}%
}
```

main This option identifies the main locale. This will default to the operating system's locale if not set. The value should be the language tag or the keyword `locale`.

```
\define@key{tex-locale.sty}{main}{%
```

If this has already been set in the earlier tracked dialect check, move the original main to the other list.

```
  \ifx\@locale@trackedmain\undefined
  \else
    \ifx\LocaleOther\undefined
      \let\LocaleOther\@locale@trackedmain
    \else
      \edef\LocaleOther{\@locale@trackedmain,\LocaleOther}%
    \fi
    \let\@locale@trackedmain\undefined
  \fi
  \def\LocaleMain{#1}%
}
```

other This option identifies the other locales. The value should be a comma-separated list of language tags or the keyword `locale`. The value must be grouped to protect any commas from the key-value parser. Note this option has a cumulative effect.

```
\define@key{tex-locale.sty}{other}{%
  \ifx\LocaleOther\undefined
```

```
      \def\LocaleOther{#1}%
    \else
      \edef\LocaleOther{\LocaleOther,#1}%
    \fi
  }
```

`symbols` This option identifies the symbol package to automatically load. For example, textcomp or `fontawesome`. If another package is required, texosquery's currency symbol commands will need to be redefined as appropriate. The key word `none` indicates that no package is required. The default is `none` for XeLaTeX or LuaLaTeX and textcomp otherwise. `textcomp`

```
\@locale@ifxeorlua{\def\@locale@symbols{none}}{\def\@locale@symbols{textcomp}}
\define@key{tex-locale.sty}{symbols}{\def\@locale@symbols{#1}}
```

To make it easier to add extra support options, assign a numeric value to each option so that `\ifcase` can be used. Recognised values: 0 (none), 1 (auto), 2 (babel), 3 (polyglossia), 4 (cjk).

`@locale@supportopt`

```
\newcount\@locale@supportopt
```

The default value is auto unless a known language package has already been loaded.

```
\@ifpackageloaded{polyglossia}
{\@locale@supportopt=0\relax}
{
  \@ifpackageloaded{babel}
  {\@locale@supportopt=0\relax}
  {
    \@ifpackageloaded{CJK}
    {\@locale@supportopt=0\relax}
    {\@locale@supportopt=1\relax}
  }
}
```

`support` This option identifies the language support package.

```
\define@choicekey{tex-locale.sty}{support}%
 [\@locale@support@val\@locale@support@nr]%
 {none,auto,babel,polyglossia,cjk}
{\@locale@supportopt=\@locale@support@nr\relax}
```

`fontspec` This option identifies whether or not to load fontspec if XeLaTeX or LuaLaTeX is in use. Defaults to true and is ignored if neither XeLaTeX nor LuaLaTeX are in use.

```
\define@boolkey{tex-locale.sty}[@locale@]{fontspec}[true]{}
\@locale@ifxeorlua{\@locale@fontspectrue}{\@locale@fontspecfalse}
```

`fontenc` This option identifies whether or not to load fontenc if neither XeLaTeX nor LuaLaTeX is in use. The value should be either `none` (don't load) or `auto` (determined from the main language or script) or the

option to pass to fontenc. If the auto option is used but the appropriate encoding can't be determined, fontenc won't be loaded. The default value is auto unless fontenc has already been loaded. This option is ignored if either XeLaTeX or LuaLaTeX are in use.

```
\define@key{tex-locale.sty}{fontenc}{%
 \edef\@locale@fontenc{#1}%
 \ifdefstring{\@locale@fontenc}{false}{\def\@locale@fontenc{none}}{}%
}
\@ifpackageloaded{fontenc}
{\def\@locale@fontenc{none}}
{\def\@locale@fontenc{auto}}
```

`inputenc` This option identifies whether or not to load inputenc if neither XeLaTeX nor LuaLaTeX is in use. The value should be either none (don't load) or auto (determined from the locale's default encoding) or the option to pass to inputenc. If the auto option is used but the appropriate encoding can't be determined, inputenc won't be loaded. The default value is auto. This option is ignored if either XeLaTeX or LuaLaTeX are in use. Note that loading inputenc before the query is performed will cause a problem with non-ASCII characters appearing in the result.

```
\define@key{tex-locale.sty}{inputenc}{%
  \edef\@locale@inputenc{#1}%
  \ifdefstring{\@locale@inputenc}{false}{\def\@locale@inputenc{none}}{}%
}
\@ifpackageloaded{inputenc}
{\def\@locale@inputenc{none}}
{\def\@locale@inputenc{auto}}
```

`\@locale@load@dtm` This command is used to load datetime2, if required, and setup the style.

```
\@ifpackageloaded{datetime2}
{
```

User has already loaded datetime2. Assume they have already set their preferred style. (They would also need to have loaded babel/polyglossia before datetime2, which rather reduces the point of the tex-locale package.)

```
  \newcommand\@locale@load@dtm{}
}
{
  \newcommand\@locale@load@dtm{%
```

First check if datetime2 is installed. If it isn't, then default to datetime2.

```
    \IfFileExists{datetime2.sty}%
      {\@locale@load@regional@dtm}%
      {\@locale@set@today}%
  }%
}
```

`@load@regional@dtm` Load datetime2 with regional settings.

```
\newcommand\@locale@load@regional@dtm{%
  \localedatechoice
    {%full
     \PassOptionsToPackage{showdow}{datetime2}%
     \PassOptionsToPackage{useregional=text}{datetime2}%
    }%
    {%long
      \PassOptionsToPackage{useregional=text}{datetime2}%
    }%
    {%medium
      \PassOptionsToPackage{useregional=text}{datetime2}%
    }%
    {%short
      \PassOptionsToPackage{useregional=numeric}{datetime2}%
    }%
  \RequirePackage{datetime2}%
  \localedatechoice{}{}{\DTMlangsetup*{abbr}}{}%
}%
```

`@locale@set@today` Redefine `\today`.

```
\newcommand*{\@locale@set@today}{%
```

If `\CurrentLocaleDate` is empty then the query failed, in which case don't change `\today`.

```
  \ifdefempty\CurrentLocaleDate
  {}%
  {%
    \renewcommand{\today}{\CurrentLocaleDate}%
    \ForEachTrackedDialect{\locale@this@dialect}
    {%
```

Add to `\date`⟨*lang*⟩ hook.

```
      \SetCurrentTrackedDialect{\locale@this@dialect}%
      \@TrackLangAddToHook
       {\renewcommand{\today}{\CurrentLocaleDate}}
       {date}%
    }%
  }%
}
```

`datetime` Provide an option to adjust the date and time settings.

```
\define@choicekey{tex-locale.sty}{datetime}%
 [\@locale@datetime@val\@locale@datetime@nr]%
 {false,iso,text,num,numeric,locale}
{%
  \ifcase\@locale@datetime@nr
```

```
    \def\@locale@load@dtm{}%
  \or
    \def\@locale@load@dtm{%
      \PassOptionsToPackage{useregional=false}{datetime2}%
      \RequirePackage{datetime2}%
      \DTMsetstyle{iso}%
    }%
  \or
    \def\@locale@load@dtm{%
      \@locale@load@regional@dtm
    }%
  \or
    \def\@locale@load@dtm{%
      \PassOptionsToPackage{useregional=numeric}{datetime2}%
      \RequirePackage{datetime2}%
    }%
  \or
    \def\@locale@load@dtm{%
      \PassOptionsToPackage{useregional=numeric}{datetime2}%
      \RequirePackage{datetime2}%
    }%
  \or
    \def\@locale@load@dtm{\@locale@set@today}%
  \fi
}
```

`iso` Shortcut for datetime2.

```
\@locale@declareoption{iso}{%
  \def\@locale@load@dtm{%
    \PassOptionsToPackage{useregional=false}{datetime2}%
    \RequirePackage{datetime2}%
    \DTMsetstyle{iso}%
  }%
}
```

`date` Preferred date style for `\CurrentLocaleDate`.

```
\define@choicekey{tex-locale.sty}{date}%
[\@locale@date@val\@locale@date@nr]{full,long,medium,short}
{%
  \ifcase\@locale@date@nr
    \def\localedatechoice##1##2##3##4{##1}%
  \or
    \def\localedatechoice##1##2##3##4{##2}%
  \or
```

```
      \def\localedatechoice##1##2##3##4{##3}%
    \or
      \def\localedatechoice##1##2##3##4{##4}%
    \fi
  }
  \newcommand*{\localedatechoice}[4]{#2}%
```

`time` Preferred date style for `\CurrentLocaleTime`.

```
  \define@choicekey{tex-locale.sty}{time}%
  [\@locale@time@val\@locale@time@nr]{full,long,medium,short}
  {%
    \ifcase\@locale@time@nr
      \def\localetimechoice##1##2##3##4{##1}%
    \or
      \def\localetimechoice##1##2##3##4{##2}%
    \or
      \def\localetimechoice##1##2##3##4{##3}%
    \or
      \def\localetimechoice##1##2##3##4{##4}%
    \fi
  }
  \newcommand*{\localetimechoice}[4]{#3}%
```

`timedata` Determine whether or not to use `-M` and `-Z`.

```
  \define@choicekey{tex-locale.sty}{timedata}%
  [\@locale@timedata@val\@locale@timedata@nr]{true,false}[true]
  {%
    \ifcase\@locale@timedata@nr
      \def\LocaleIfDateTimePatternsSupported##1##2{##1}%
    \or
      \def\LocaleIfDateTimePatternsSupported##1##2{##2}%
    \fi
  }
```

`currency` Preferred currency style for `\CurrentLocaleCurrency`.

```
  \define@choicekey{tex-locale.sty}{currency}%
  [\@locale@currency@val\@locale@currency@nr]%
  {official,unofficial,sym,tex}
  {%
    \ifcase\@locale@currency@nr
      \def\localecurrchoice##1##2##3##4{##1}%
    \or
      \def\localecurrchoice##1##2##3##4{##2}%
    \or
```

```
    \def\localecurrchoice##1##2##3##4{##3}%
  \or
    \def\localecurrchoice##1##2##3##4{##4}%
  \fi
}
\@locale@ifxeorlua
 {\newcommand*{\localecurrchoice}[4]{#3}}
 {\newcommand*{\localecurrchoice}[4]{#4}}
```

Process package options. First process any options that have been passed via the document class.

```
\@for\CurrentOption:=\@declaredoptions\do{%
  \ifx\CurrentOption\@empty
  \else
    \@expandtwoargs
      \in@{,\CurrentOption,}{,\@classoptionslist,\@curroptions,}%
    \ifin@
      \@use@ption
      \expandafter\let\csname ds@\CurrentOption\endcsname\@empty
    \fi
  \fi
}
```

Now process options passed to the package:

```
\ProcessOptionsX
```

Check if babel or polyglossia have already been loaded (user may have used support or support without realising it's already been loaded). Since polyglossia pretends babel has been loaded, only babel test is required.

```
\@ifpackageloaded{babel}
{%
  \@locale@supportopt=0\relax
}
{}
```

If the main language hasn't been set, define it (needed before the generic code is input.)

```
\ifx\LocaleMain\undefined
  \def\LocaleMain{locale}
\fi
\ifx\LocaleOther\undefined
  \def\LocaleOther{}
\fi
```

`le@postparse@hook` Post-parser hook.

```
\newcommand*{\@locale@postparse@hook}{%
  \input{tex-locale-support.def}%
}
```

If symbol support is required, load the package now:

```
\ifdefstring{\@locale@symbols}{none}
{}
{\RequirePackage{\@locale@symbols}}
```

Load texosquery using package interface (needs to be done after the symbol support is loaded since `texosquery.tex` detects some common symbol commands).

```
\RequirePackage{texosquery}
```

The encoding needs to be set up before the main query in case there are any non-ASCII characters returned by the query (for example, the currency symbol or in month or day names).

May need to track all the listed dialects before `tex-locale.tex` is loaded.

`\@locale@trackall` Track the main language if set.

```
\newcommand\@locale@trackall{%
  \ifdefstring\LocaleMain{locale}%
  {%
    \ifx\LocaleOStag\empty
      \PackageWarning{tex-locale}{Unable to determine locale
        (check shell escape)}%
    \else
      \TrackLanguageTag{\LocaleOStag}%
    \fi
  }%
  {\TrackLanguageTag{\LocaleMain}}%
```

Track the other languages.

```
  \@for\locale@this@dialect:=\LocaleOther\do{%
    \ifdefstring\locale@this@dialect{locale}
    {%
      \ifx\LocaleOStag\empty
        \PackageWarning{tex-locale}{Unable to determine locale
          (check shell escape)}%
      \else
        \TrackLanguageTag{\LocaleOStag}%
      \fi
    }%
    {\TrackLanguageTag{\locale@this@dialect}}%
  }%
  \let\@locale@trackall\relax
}
```

`cale@loadinputenc`

```
\newcommand{\@locale@loadinputenc}{}
```

It may be necessary to use texosquery here to determine the language tag (-b) and/or the codeset (-C). It makes more sense to minimise the number of shell escapes, so try to determine what extra information we need here. The required parameters are stored in \@locale@pre@query@params

@pre@query@params

```
\newcommand*\@locale@pre@query@params{}%
```

Now we need some commands that will parse the result, depending on the parameters.

re@query@parsetag Only the language tag (-b) is required. The shell escape will only return a single result. \LocaleOStag will either be set to the language tag, if successful, or will be empty, if in dry run mode.

```
\newcommand*\@locale@pre@query@parsetag{%
  \ifx\LocaleStyQueryFile\undefined
   \TeXOSQuery{\@locale@result}{\@locale@pre@query@params}%
  \else
   \ifx\LocaleStyQueryFile\empty
     \TeXOSQuery{\@locale@result}{\@locale@pre@query@params}%
   \else
     \ifx\TeXOSQueryFromFile\undefined
       \PackageError{locale}{texosquery too old to support
       \string\LocaleStyQueryFile. At least v1.4 required}
       {You need to update your version of texosquery}
       \def\@locale@result{}%
     \else
       \PackageInfo{locale}{Fetching query results from `\LocaleStyQueryFile'}%
       \TeXOSQueryFromFile{\@locale@result}{\LocaleStyQueryFile}%
     \fi
   \fi
  \fi
  \edef\LocaleOStag{\@locale@result}%
}%
```

uery@parsecodeset Only the codeset (-C) is required. The shell escape will only return a single result. \LocaleOScodeset will either be set to the codeset, if successful, or will be empty, if in dry run mode.

```
\newcommand*\@locale@pre@query@parsecodeset{%
  \TeXOSQuery{\@locale@result}{\@locale@pre@query@params}%
  \edef\LocaleOScodeset{\@locale@result}%
}%
```

y@parsetagcodeset Both the language tag (-b) and the codeset (-C) are required. The shell escape will return two arguments.

```
\newcommand*\@locale@pre@query@parsetagcodeset{%
  \def\LocaleOStag{}%
  \def\LocaleOScodeset{}%
  \TeXOSQuery{\@locale@result}{\@locale@pre@query@params}%
```

```
    \ifx\@locale@result\@empty
    \else
      \edef\LocaleOStag{\expandafter\@firstoftwo\@locale@result}%
      \edef\LocaleOScodeset{\expandafter\@secondoftwo\@locale@result}%
    \fi
  }%
```

e@pre@query@parse This will be set to the appropriate command after determining what information is actually required.

```
  \let\@locale@pre@query@parse\relax
```

If XeLaTeX or LuaLaTeX is in use and the fontspec option is on, load fontspec.

```
  \@locale@ifxeorlua
  {%
    \if@locale@fontspec\RequirePackage{fontspec}\fi
```

The language tag is needed if support.

```
    \ifnum\@locale@supportopt=1\relax
      \ifdefstring\LocaleMain{locale}
      {%
         \def\@locale@pre@query@params{\string-b }%
         \let\@locale@pre@query@parse\@locale@pre@query@parsetag
      }%
      {%
        \@for\locale@this@dialect:=\LocaleOther\do{%
         \ifdefstring\locale@this@dialect{locale}
         {%
           \def\@locale@pre@query@params{\string-b }%
           \let\@locale@pre@query@parse\@locale@pre@query@parsetag
           \@endfortrue
         }%
         {}%
        }%
      }%
```

Perform the shell escape if required.

```
    \@locale@pre@query@parse
    \fi
  }
  {%
```

The language tag is needed if fontenc.

```
    \ifdefstring{\@locale@fontenc}{auto}
    {%
      \ifdefstring\LocaleMain{locale}
      {%
```

```
      \def\@locale@pre@query@params{\string-b }%
      \let\@locale@pre@query@parse\@locale@pre@query@parsetag
    }%
    {%
      \@for\locale@this@dialect:=\LocaleOther\do{%
      \ifdefstring\locale@this@dialect{locale}
      {%
        \def\@locale@pre@query@params{\string-b }%
        \let\@locale@pre@query@parse\@locale@pre@query@parsetag
        \@endfortrue
      }%
      {}%
     }%
    }%
  }%
  {}%
```

The codeset is needed if inputenc. The `-C` (`--codeset-lcs`) switch is used rather than the `-cs` (`--codeset`) switch to make it more compatible with inputenc.

```
  \ifdefstring{\@locale@inputenc}{auto}
  {%
     \ifx\@locale@pre@query@params\@empty
       \let\@locale@pre@query@parse\@locale@pre@query@parsecodeset
     \else
       \let\@locale@pre@query@parse\@locale@pre@query@parsetagcodeset
     \fi
     \edef\@locale@pre@query@params{\@locale@pre@query@params\string-C}%
  }%
  {}%
```

Perform the shell escape if required.

```
  \@locale@pre@query@parse
```

Do we need to load fontenc?

```
  \ifdefstring{\@locale@fontenc}{none}
  {}
  {%
    \ifdefstring{\@locale@fontenc}{auto}
    {%
```

If fontenc, set up some script to fontenc mappings.

```
      \input{tex-locale-scripts-enc.def}%
      \@locale@trackall
```

Iterate through all the dialects.

```
      \ForEachTrackedDialect{\locale@this@dialect}%
```

```
 {
   \edef\@locale@lang{\TrackedLanguageFromDialect{\locale@this@dialect}}%
   \@locale@if@langenc@map{\locale@this@dialect}%
   {%
       \edef\@locale@fontenc@opt{%
         \@locale@get@langenc@map{\locale@this@dialect}}%
       \expandafter\PassOptionsToPackage\expandafter
         {\@locale@fontenc@opt}{fontenc}%
   }%
   {%
     \@locale@if@langenc@map{\@locale@lang}%
     {%
       \edef\@locale@fontenc@opt{%
         \@locale@get@langenc@map{\@locale@lang}}%
       \expandafter\PassOptionsToPackage\expandafter
         {\@locale@fontenc@opt}{fontenc}%
     }%
     {%
       \edef\@locale@script{\GetTrackedDialectScript{\locale@this@dialect}}%
       \ifx\@locale@script\empty
        \edef\@locale@script{\TrackLangGetDefaultScript{\@locale@lang}}%
       \fi
       \@locale@if@scriptenc@map{\@locale@script}%
       {%
         \edef\@locale@fontenc@opt{%
           \@locale@get@scriptenc@map{\@locale@script}}%
         \expandafter\PassOptionsToPackage\expandafter
           {\@locale@fontenc@opt}{fontenc}%
       }%
       {}%
     }%
   }%
 }%
 \ifx\@locale@fontenc@opt\empty
   \PackageWarning{tex-locale}{Option ‘fontenc=auto’ failed.
     Can’t determine an appropriate font encoding for dialect(s).
     (Dialect list: \@tracklang@dialects.)
     Either set the encoding explicitly or switch to XeLaTeX%
   }%
 \else
   \RequirePackage{fontenc}%
 \fi
}
```

```
    {%
      \expandafter\PassOptionsToPackage\expandafter{\@locale@fontenc}{fontenc}%
      \RequirePackage{fontenc}%
    }
  }
```

Do we need to load inputenc? This needs to be done before the main `tex-locale.tex` shell escape.

```
  \ifdefstring{\@locale@inputenc}{none}
  {}
  {%
    \ifdefstring{\@locale@inputenc}{auto}
    {%
```

If the earlier shell escape was successful, `\LocaleOScodeset` will be set.

```
      \ifx\LocaleOScodeset\empty
```

Query failed.

```
        \PackageWarning{tex-locale}{Option `inputenc=auto' failed.
        (Check shell escape.) Default file encoding unavailable}
      \else
```

If the file `\LocaleOScodeset.def` exists, then the codeset should hopefully be valid.

```
        \IfFileExists{\LocaleOScodeset.def}%
        {%
          \renewcommand{\@locale@loadinputenc}{%
            \RequirePackage{inputenc}%
            \inputencoding{\LocaleOScodeset}%
          }
          \let\@locale@inputenc\LocaleOScodeset
        }%
        {%
```

May have a different name. Try known encoding mappings.

```
          \input{tex-locale-encodings.def}%
          \@locale@ifhasencmap{\LocaleOScodeset}
          {
             \edef\@locale@inputenc{\@locale@getencmap\LocaleOScodeset}
             \renewcommand{\@locale@loadinputenc}{%
               \RequirePackage{inputenc}%
               \inputencoding{\@locale@inputenc}%
             }
          }
          {%
            \PackageWarning{tex-locale}{Option `inputenc=auto' failed.
             Don't know how to interpret codeset `\LocaleOScodeset'}%
          }%
```

```
        }
      \fi
    }
    {%
      \renewcommand{\@locale@loadinputenc}{%
        \RequirePackage{inputenc}%
        \inputencoding{\@locale@inputenc}%
      }
    }
  }
}
```

`ocale@loadscripts` Only load tracklang-scripts if required.

```
\newcommand{\@locale@loadscripts}{%
  \RequirePackage{tracklang-scripts}%
  \let\@locale@loadscripts\relax
}
```

The LaTeX kernel provides `\@thirdofthree` but not `\@secondofthree`.

`\@secondofthree`

```
\providecommand*{\@secondofthree}[3]{#2}
```

`supportbabelorpoly` `\@locale@ifsupportbabelorpoly{⟨neither case⟩}{⟨babel case⟩}{⟨polyglossia case⟩}`

Should babel or polyglossia be loaded? Initialise to false.

```
\newcommand*{\@locale@ifsupportbabelorpoly}[3]{#1}
```

`cale@ifsupportcjk` Should CJK be loaded? Initialise to false.

```
\newcommand*{\@locale@ifsupportcjk}[2]{#2}
```

`e@ifsupportpinyin` Should pinyin be loaded? Initialise to false.

```
\newcommand*{\@locale@ifsupportpinyin}[2]{#2}
```

`\@locale@cjklist` List of languages supported by CJK. This just makes it easier to test the language without multiple conditions.

```
\newcommand*{\@locale@cjklist}{}
\listadd{\@locale@cjklist}{chinese}
\listadd{\@locale@cjklist}{japanese}
\listadd{\@locale@cjklist}{korean}
\listadd{\@locale@cjklist}{thai}
```

\@locale@ifcjk `\@locale@ifcjk{⟨language⟩}{⟨true case⟩}{⟨false case⟩}`

Check if given root language is in the CJK list.

```
\newcommand*{\@locale@ifcjk}[1]{%
  \xifinlist{#1}{\@locale@cjklist}%
}
```

cale@iflatinscript `\@locale@iflatinscript{⟨dialect⟩}{⟨true case⟩}{⟨false case⟩}`

Check if given dialect has the script explicitly set to Latn.

```
\newcommand*{\@locale@iflatinscript}[1]{%
  \ifcsstring{@tracklang@script@#1}{Latn}%
}
```

Should babel or polyglossia be loaded?

```
\ifcase\@locale@supportopt
```

No support required.

```
\or
```

auto option.

```
  \@locale@trackall
  \ForEachTrackedDialect{\locale@this@dialect}%
  {%
```

Get root language name.

```
    \edef\this@root@lang{\TrackedLanguageFromDialect{\locale@this@dialect}}%
```

Is this language in the CJK list?

```
    \@locale@ifcjk{\this@root@lang}%
    {
      \let\@locale@ifsupportcjk\@firstoftwo
```

Is pinyin needed?

```
      \@locale@iflatinscript{\locale@this@dialect}%
      {\let\@locale@ifsupportpinyin\@firstoftwo}%
      {}
    }
    {}
```

No point checking for polyglossia support if not using XeLaTeX or LuaLaTeX.

```
    \@locale@ifxeorlua
    {
```

```
        \@locale@ifsupportbabelorpoly
        {
```

Haven't yet determined support for babel or polyglossia. Does the file `gloss-`⟨*language*⟩`.ldf` exist?

```
          \IfFileExists{gloss-\this@root@lang.ldf}
          {
            \let\@locale@ifsupportbabelorpoly\@thirdofthree
          }
          {
```

Does the file ⟨*language*⟩`.ldf` exist?

```
            \IfFileExists{\this@root@lang.ldf}
            {\let\@locale@ifsupportbabelorpoly\@secondofthree}
            {}
          }
        }
```

Don't bother checking if already determined that babel needs to be used.

```
        {}
        {
```

Already found one language supported by polyglossia. Now check this one. Does the file `gloss-`⟨*language*⟩`.ld` exist?

```
          \IfFileExists{gloss-\this@root@lang.ldf}
          {}
          {
```

No support for this language with polyglossia. Does the file ⟨*language*⟩`.ldf` exist?

```
            \IfFileExists{\this@root@lang.ldf}
            {\let\@locale@ifsupportbabelorpoly\@secondofthree}
            {}
          }
        }
      }
      {
```

Not using XeLaTeX or LuaLaTeX so no polyglossia support. Does the file ⟨*language*⟩`.ldf` exist?

```
        \IfFileExists{\this@root@lang.ldf}
        {\let\@locale@ifsupportbabelorpoly\@secondofthree}
        {}
      }
    }
  \or
```

babel option.

```
    \let\@locale@ifsupportbabelorpoly\@secondofthree
  \or
```

polyglossia option.

```
    \let\@locale@ifsupportbabelorpoly\@thirdofthree
  \or
```

cjk option.

```
    \let\@locale@currentiscjk\@secondoftwo
```

Is pinyin needed?

```
    \ForEachTrackedDialect{\locale@this@dialect}%
    {%
      \edef\this@root@lang{\TrackedLanguageFromDialect{\locale@this@dialect}}%
      \@locale@ifcjk{\this@root@lang}%
      {
        \@locale@iflatinscript{\locale@this@dialect}%
        {\let\@locale@ifsupportpinyin\@firstoftwo}%
        {}
      }
      {}
    }
```

End of case statement

```
  \fi
```

Does CJK need to be loaded?

```
  \@locale@ifsupportcjk
  {
```

Is UTF-8 support needed?

```
    \@locale@ifxeorlua
    {
      \RequirePackage{xeCJK}%
    }
    {
      \ifdefstring\@locale@inputenc{utf8}%
      {%
        \RequirePackage{CJKutf8}%
```

CJKutf8 automatically loads inputenc with the utf8 option.

```
        \renewcommand{\@locale@loadinputenc}{}%
```

Need to ensure UTF-8 characters are correctly set up when the query is made.

```
        \newcommand*{\localeprequery}{\begin{CJK}{UTF8}{}\makeatletter}
        \newcommand*{\localepostquery}{\end{CJK}}
      }%
      {%
        \RequirePackage{CJK}%
```

Non-UTF encoding. Not implemented as I'm not familiar with these encodings.

```
        \PackageWarning{tex-locale}{Unsupported encoding `\@locale@loadinputenc'}%
      }%
    }
```

Load pinyin if needed.

```
    \@locale@ifsupportpinyin
    {\RequirePackage{pinyin}}%
    {}%
  }
  {}
```

Load inputenc if required.

```
  \@locale@loadinputenc
```

`localenopolypunct` Just does its argument (scoped) if polyglossia hasn't been loaded. (Made robust if polyglossia is loaded.)

```
\newcommand{\localenopolypunct}[1]{{#1}}
```

`locale@nopolypunct` Robust form used with polyglossia.

```
\newrobustcmd{\@locale@nopolypunct}[1]{%
  {%
    \@tracklang@ifundef{no\languagename @punctuation}{}%
    {\csname no\languagename @punctuation\endcsname}%
    #1%
  }%
}
```

Load the generic code.

```
\input{tex-locale}
```

Load datetime2 if required:

```
\@locale@load@dtm
```

`SupportPackageCase` `\LocaleSupportPackageCase{⟨babel⟩}{⟨polyglossia⟩}{⟨neither⟩}`

Provide a user-level command to determine whether babel or polyglossia was used. This doesn't test if CJK was loaded, which may have additionally been loaded.

```
\@ifpackageloaded{polyglossia}
{\newcommand{\LocaleSupportPackageCase}[3]{#2}}
{%
  \@ifpackageloaded{babel}
  {\newcommand{\LocaleSupportPackageCase}[3]{#1}}%
```

```
    {\newcommand{\LocaleSupportPackageCase}[3]{#3}}%
  }
```

4.2 Generic Code (tex-locale.tex)

Does the category code of @ need changing?

\@locale@restore@at

```
  \ifnum\catcode`\@=11\relax
    \def\@locale@restore@at{}%
  \else
    \expandafter\edef\csname @locale@restore@at\endcsname{%
      \noexpand\catcode`\noexpand\@=\number\catcode`\@\relax
    }%
   \catcode`\@=11\relax
  \fi
```

First check if this file has already been loaded:

```
  \ifx\@locale@parse@query\undefined
  \else
    \@locale@restore@at
    \expandafter\endinput
  \fi
```

Version info.

```
  \expandafter\def\csname ver@tex-locale.tex\endcsname{2018/08/26 v1.0
  (NLCT) localisation support}
```

Load tracklang and texosquery:

```
  \input tracklang
  \input texosquery
```

\@locale@err

```
  \ifx\PackageError\undefined
    \def\@locale@err#1#2{%
      \errhelp{#2}%
      \errmessage{tex-locale: #1}}
  \else
    \def\@locale@err#1#2{\PackageError{tex-locale}{#1}{#2}}
  \fi
```

\@locale@warn Use tracklang's warning to allow all warnings to be switched off at the same time.

```
  \def\@locale@warn{\@tracklang@pkgwarn{tex-locale}}
```

\@locale@info Information message.

```
\ifx\PackageInfo\undefined
  \def\@locale@info#1{%
    {%
      \newlinechar=`\^^J
      \def\MessageBreak{^^J}%
      \message{^^Jtex-locale Info: #1^^J}%
    }%
  }
\else
  \def\@locale@info#1{\PackageInfo{tex-locale}{#1}}
\fi
```

Check tracklang is at least v1.3.4.

```
\ifx\@tracklang@pkgwarn\undefined
  \@locale@err{tracklang version is too old. At least v1.3.4 required}
   {You need to update tracklang to at least v1.3.4}%
\fi
```

Check texosquery is at least v1.4.

```
\ifx\@texosquery@argquote\undefined
  \@locale@err{texosquery version is too old. At least v1.4 required}
   {You need to update texosquery to at least v1.4}%
\fi
```

\LocaleMain If `\LocaleMain` hasn't been defined, define it. This macro stores the language tag of the document's main region or `locale` to use the OS locale. The default is `locale`. This is a user-level command so it can be set before loading `tex-locale.tex`.

```
\ifx\LocaleMain\undefined
  \def\LocaleMain{locale}
\fi
```

Sanitize just in case.

```
\@tracklang@sanitize\LocaleMain
```

\LocaleOther If `\LocaleOther` hasn't been defined, define it. This macro stores a comma-separated list of language tags or `locale` for additional regions. The default is empty. This is a user-level command so it can be set before loading `tex-locale.tex`.

```
\ifx\LocaleOther\undefined
  \def\LocaleOther{}
\else
```

Sanitize just in case.

```
  \@tracklang@sanitize\LocaleOther
\fi
```

\@locale@os@tag Command to keep track of the OS locale. This is initialised as empty but will be set if the texosquery call is successful.

```
\def\@locale@os@tag{}
```

@locale@os@default The keyword used to indicate the OS locale.

```
\def\@locale@os@default{locale}
```

Sanitize since it needs to be compared with \LocaleMain.

```
\@tracklang@sanitize\@locale@os@default
```

@unknown@currency Unknown currency designator.

```
\def\@locale@unknown@currency{XXX}
```

@locale@os@region The OS region. Initialised as empty but will be set if the texosquery call is successful.

```
\def\@locale@os@region{}
```

ocale@os@groupsep The OS numeric group sep. Initialised as empty but will be set if the texosquery call is successful.

```
\def\@locale@os@groupsep{}
```

@locale@os@decsep The OS numeric group sep. Initialised as empty but will be set if the texosquery call is successful.

```
\def\@locale@os@decsep{}
```

@locale@os@cursep The OS currency separator. Initialised as empty but will be set if the texosquery call is successful.

```
\def\@locale@os@cursep{}
```

\@locale@os@exp The OS exponent symbol. Initialised as empty but will be set if the texosquery call is successful.

```
\def\@locale@os@exp{}
```

cale@os@usesgroup The OS numeric uses group value. Initialised as empty but will be set if the texosquery call is successful.

```
\def\@locale@os@usesgroup{}
```

e@os@currencycode The OS currency code. Initialised as unknown code but will be set if the texosquery call is successful.

```
\def\@locale@os@currencycode{XXX}
```

gionalcurrencycode The OS regional currency code. Initialised as empty but will be set if the texosquery call is successful.

```
\def\@locale@os@regionalcurrencycode{XXX}
```

le@os@currencysym The OS currency symbol. Initialised as empty but will be set if the texosquery call is successful.

```
\def\@locale@os@currencysym{}
```

le@os@currencytex The OS currency TEX code. Initialised as empty but will be set if the texosquery call is successful.

```
\def\@locale@os@currencytex{}
```

\LocaleMainFile The document's main file. (The modification date is queried if not empty.) If \jobname includes double-quotes, these need to be stripped to avoid interfering with the shell-escape (especially in restricted mode).

```
\ifx\LocaleMainFile\undefined
  \edef\LocaleMainFile{\expandafter\texosquerystripquotes{\jobname}.tex}
\fi
```

PatternsSupported The -M and -Z lookup is optional. This may be defined before texosquery.tex is input.

```
\ifx\LocaleIfDateTimePatternsSupported\undefined
 \def\LocaleIfDateTimePatternsSupported#1#2{#2}
\fi
```

localedatetimefmt Allow the date/time to be wrapped in a formatting command.

```
\def\localedatetimefmt#1{#1}
```

4.2.1 Setting Command Line Switches

QueryCodesetParam texosquery has two different actions for obtaining the codeset: --codeset-lcs (-C) and --codeset (-cs). The second returns the codeset name as recognised by Java. The first returns a modified version that's closer to the inputenc setting. This macro defaults to using -C but may be defined before this file is loaded to use -cs instead. (This command isn't used by locale.sty when determining the input encoding for inputenc.)

```
\ifx\LocaleQueryCodesetParam\undefined
  \edef\LocaleQueryCodesetParam{\string-C}
\fi
```

cale@query@params It's more efficient to have a single texosquery call, but the parameters need to be determined first. (Use short arg with \string just in case the hyphen character has a special meaning.)

```
\edef\@locale@query@params{%
```

Since we need to use texosquery anyway, may as well look up the OS information at the same time.

```
  \string-o \string-r \string-a
```

May as well get the PDF date in case we have a TeX format that doesn't provide \pdfcreationdate.

```
  \string-n
```

Default locale data (requires texosquery v1.2):

```
  \string-N
```

May as well get the default codeset (requires texosquery v1.2). Note that locale.sty may have already found this if the inputenc package was automatically loaded.

```
  \LocaleQueryCodesetParam\space
}
```

The remaining arguments need to be appended programatically.

If the date-time patterns are needed, get the full date-time information.

```
\LocaleIfDateTimePatternsSupported
{%
  \edef\@locale@query@params{\@locale@query@params \string-M }
}
{}
```

May as well get the modification date of the document file since we need to make a system call anyway, but only if `\LocaleMainFile` isn't empty.

```
\ifx\LocaleMainFile\empty
\else
 \edef\@locale@query@params{\@locale@query@params
   \string-d \@texosquery@argquote{\LocaleMainFile}
 }
\fi
```

The `-D` switch also requires texosquery v1.2. First deal with the main language.

```
\ifx\LocaleMain\@locale@os@default
 \LocaleIfDateTimePatternsSupported
 {%
   \edef\@locale@query@params{\@locale@query@params
     \string-D \string-Z
   }
 }
 {
   \edef\@locale@query@params{\@locale@query@params
     \string-D
   }
 }
\else
 \LocaleIfDateTimePatternsSupported
 {%
   \edef\@locale@query@params{\@locale@query@params
     \string-D \LocaleMain\space\string-Z \LocaleMain\space
   }
 }
 {
   \edef\@locale@query@params{\@locale@query@params
     \string-D \LocaleMain\space
   }
 }
\fi
```

Iterate through the list of other languages.

```
\@tracklang@for\@locale@tag:=\LocaleOther\do{%
  \ifx\@locale@tag\@locale@os@default
```

```
    \LocaleIfDateTimePatternsSupported
    {%
      \edef\@locale@query@params{\@locale@query@params
        \string-D \string-Z
      }
    }
    {%
      \edef\@locale@query@params{\@locale@query@params
        \string-D
      }
    }
   \else
    \LocaleIfDateTimePatternsSupported
    {%
      \edef\@locale@query@params{\@locale@query@params
        \string-D \@locale@tag\space\string-Z \@locale@tag\space
      }
    }
    {%
      \edef\@locale@query@params{\@locale@query@params
        \string-D \@locale@tag\space
      }
    }
   \fi
}
```

4.2.2 System Call

\localeprequery Allow for a hook immediately before the query.

```
\csname localeprequery\endcsname
```

Now run texosquery unless \LocaleQueryFile has been defined, in which case use \TeXOSQueryFromFile

```
\ifx\LocaleQueryFile\undefined
```

No file provided.

```
  \TeXOSQuery{\@locale@result}{\@locale@query@params}
\else
  \ifx\LocaleQueryFile\empty
    \TeXOSQuery{\@locale@result}{\@locale@query@params}
  \else
```

\TeXOSQueryFromFile was added to texosquery v1.4, so check it's available.

```
    \ifx\TeXOSQueryFromFile\undefined
      \@locale@err{texosquery too old to support
      \string\LocaleQueryFile. At least v1.4 required}
```

```
      {You need to update your version of texosquery}
      \def\@locale@result{}
    \else
      \@locale@info{Fetching query results from `\LocaleQueryFile'}%
      \TeXOSQueryFromFile{\@locale@result}{\LocaleQueryFile}
    \fi
  \fi
\fi
```

Make the result global in case the pre and post query hooks have introduced a local scope.

```
\global\let\@locale@result\@locale@result
```

\localepostquery Allow for a hook immediately after the query.

```
\csname localepostquery\endcsname
```

If the result is empty then the query failed (texosquery not installed correctly or JRE not installed or shell escape not permitted or dry run mode on). The result now needs to be parsed, but first define some convenient commands that can be used by the parser.

4.2.3 Attributes

LocaleSetAttribute `\LocaleSetAttribute{⟨label⟩}{⟨attribute label⟩}{⟨attribute value⟩}`

Provide convenient way of defining attributes. The label depends on the attribute type. For example, it could be the dialect label or the region code.

```
\def\LocaleSetAttribute#1#2#3{%
  \expandafter\def\csname locale@#2@#1\endcsname{#3}%
}
```

caleAppToAttribute `\LocaleAppToAttribute{⟨label⟩}{⟨attribute label⟩}{⟨value⟩}`

Append ⟨*value*⟩ to this attribute value.

```
\def\LocaleAppToAttribute#1#2#3{%
  \LocaleIfHasAttribute{#1}{#2}%
  {%
    \expandafter\expandafter\expandafter\def
      \expandafter\expandafter\csname locale@#2@#1\expandafter\endcsname
      \expandafter\expandafter\expandafter{\csname locale@#2@#1\endcsname#3}%
  }%
  {\expandafter\def\csname locale@#2@#1\endcsname{#3}}%
}
```

leXpAppToAttribute

\LocaleXpAppToAttribute{⟨*label*⟩}{⟨*attribute label*⟩}{⟨*value*⟩}

As above but expand first token of ⟨*value*⟩.

```
\def\LocaleXpAppToAttribute#1#2#3{%
  \LocaleIfHasAttribute{#1}{#2}%
  {%
    \expandafter\expandafter\expandafter\def
      \expandafter\expandafter\csname locale@#2@#1\expandafter\endcsname
      \expandafter\expandafter\expandafter{%
        \csname locale@#2@#1\expandafter\endcsname#3}%
  }%
  {%
   \expandafter\def\csname locale@#2@#1\expandafter\endcsname\expandafter{#3}%
  }%
}
```

AddToAttributeList

\LocaleAddToAttributeList{⟨*label*⟩}{⟨*attribute label*⟩}{⟨*value*⟩}

Adds ⟨*value*⟩ to this attribute's list (without repetition).

```
\def\LocaleAddToAttributeList#1#2#3{%
  \LocaleIfHasAttribute{#1}{#2}%
  {%
    \LocaleIfInAttributeList{#1}{#2}{#3}%
    {}%
    {\LocaleAppToAttribute{#1}{#2}{,#3}}%
  }%
  {\LocaleSetAttribute{#1}{#2}{#3}}%
}
```

pAddToAttributeList

\LocaleXpAddToAttributeList{⟨*label*⟩}{⟨*attribute label*⟩}{⟨*value*⟩}

Adds ⟨*value*⟩ to this attribute's list (without repetition).

```
\def\LocaleXpAddToAttributeList#1#2#3{%
  \LocaleIfHasAttribute{#1}{#2}%
  {%
    \LocaleIfXpInAttributeList{#1}{#2}{#3}%
    {}%
    {\LocaleXpAppToAttribute{#1}{#2}{\expandafter,#3}}%
```

```
  }%
  {\expandafter\def\csname locale@#2@#1\expandafter\endcsname\expandafter{#3}}%
}
```

caleshowattribute Debugging command.

```
\def\localeshowattribute#1#2{%
  \LocaleIfHasAttribute{#1}{#2}%
  {%
    \expandafter\show\csname locale@#2@#1\endcsname
  }%
  {\@locale@err{Attribute ‘#2’ not defined for ‘#1’}%
  {\string\localeshowattribute\space was asked to show this
   attribute for the given attribute type, but the
   associated command hasn’t been defined}}%
}
```

leProvideAttribute `\LocaleProvideAttribute{⟨label⟩}{⟨attribute label⟩}{⟨attribute value⟩}`

Only set the attribute if it hasn't already been set for this label.

```
\def\LocaleProvideAttribute#1#2#3{%
  \LocaleIfHasAttribute{#1}{#2}%
  {}%
  {\LocaleSetAttribute{#1}{#2}{#3}}%
}
```

LocaleLetAttribute `\LocaleLetAttribute{⟨label⟩}{⟨attribute label⟩}{⟨cs⟩}`

Set the attribute value to the definition of the control sequence ⟨*cs*⟩.

```
\def\LocaleLetAttribute#1#2#3{%
  \expandafter\let\csname locale@#2@#1\endcsname#3%
}
```

ttributeOrDefValue `\LocaleGetAttributeOrDefValue{⟨label⟩}{⟨attribute label⟩}{⟨def value⟩}`

Gets the attribute or the default value if unset.

```
\def\LocaleGetAttributeOrDefValue#1#2#3{%
  \@tracklang@ifundef{locale@#2@#1}%
```

```
    {#3\@locale@undef@action{#1}{#2}}%
    {\csname locale@#2@#1\endcsname}%
  }
```

LocaleGetAttribute

\LocaleGetAttribute{⟨*label*⟩}{⟨*attribute label*⟩}

```
  \def\LocaleGetAttribute#1#2{%
   \LocaleGetAttributeOrDefValue{#1}{#2}{}%
  }
```

cale@undef@action Action if an undefined attribute is referenced. Does nothing by default but may be redefined for debugging purposes.

```
  \def\@locale@undef@action#1#2{}
```

caleIfHasAttribute

\LocaleIfHasAttribute{⟨*label*⟩}{⟨*attribute label*⟩}{⟨*true*⟩}{⟨*false*⟩}

Tests if the given attribute has been assigned.

```
  \def\LocaleIfHasAttribute#1#2#3#4{%
    \@tracklang@ifundef{locale@#2@#1}%
    {#4}%
    {#3}%
  }
```

achInAttributeList

\LocaleForEachInAttributeList{⟨*label*⟩}{⟨*attribute label*⟩}{⟨*cs*⟩}{⟨*body*⟩}

Where an attribute value is a comma-separated list, this iterates over each item in that list, setting ⟨*cs*⟩ to that item and performing ⟨*body*⟩.

```
  \def\LocaleForEachInAttributeList#1#2#3#4{%
    \LocaleIfHasAttribute{#1}{#2}%
    {%
      \expandafter\@tracklang@for\expandafter#3\expandafter:\expandafter
        =\csname locale@#2@#1\endcsname\do{#4}%
    }%
    {}%
  }
```

eIfInAttributeList `\LocaleIfInAttributeList{⟨label⟩}{⟨attribute label⟩}{⟨item⟩}{⟨true⟩}{⟨false⟩}`

Where an attribute value is a comma-separated list, this tests if ⟨*item*⟩ is in that list.

```
\def\LocaleIfInAttributeList#1#2#3#4#5{%
  \LocaleIfHasAttribute{#1}{#2}%
  {%
    \expandafter\let\expandafter\@locale@attrlist\csname locale@#2@#1\endcsname
    \@tracklang@ifinlist{#3}{\@locale@attrlist}{#4}{#5}%
  }%
  {#5}%
}
```

fXpInAttributeList `\LocaleIfXpInAttributeList{⟨label⟩}{⟨attribute label⟩}{⟨item⟩}{⟨true⟩}{⟨false⟩}`

As above but expands the first token of ⟨*item*⟩.

```
\def\LocaleIfXpInAttributeList#1#2#3#4#5{%
  \LocaleIfHasAttribute{#1}{#2}%
  {%
    \expandafter\let\expandafter\@locale@attrlist\csname locale@#2@#1\endcsname
    \expandafter\@tracklang@ifinlist\expandafter{#3}{\@locale@attrlist}{#4}{#5}%
  }%
  {#5}%
}
```

sNonEmptyAttribute `\LocaleIfHasNonEmptyAttribute{⟨label⟩}{⟨attribute label⟩}{⟨true⟩}{⟨false⟩}`

Tests if the given attribute has been assigned a non-empty value.

```
\def\LocaleIfHasNonEmptyAttribute#1#2#3#4{%
  \@tracklang@ifundef{locale@#2@#1}%
  {#4}%
  {%
    \expandafter\ifx\csname locale@#2@#1\endcsname\empty
      #4%
    \else
      #3%
    \fi
  }%
}
```

aleIfAttributeEqCs

\LocaleIfAttributeEqCs{⟨*label*⟩}{⟨*attribute label*⟩}{⟨*cs*⟩}{⟨*true*⟩}{⟨*false*⟩}

If the attribute value is the same as the definition of the control sequence ⟨*cs*⟩ to ⟨*true*⟩ otherwise do false.

```
\def\LocaleIfAttributeEqCs#1#2#3#4#5{%
  \expandafter\ifx\csname locale@#2@#1\endcsname#3%
    #4%
  \else
    #5%
  \fi
}
```

fAttributeEqCsName

\LocaleIfAttributeEqCsName{⟨*label*⟩}{⟨*attribute label*⟩}{⟨*cs name*⟩}{⟨*true*⟩}{⟨*false*⟩}

If the attribute value is the same as the definition of the control sequence name ⟨*cs name*⟩ to ⟨*true*⟩ otherwise do false.

```
\def\LocaleIfAttributeEqCsName#1#2#3#4#5{%
  \expandafter\ifx
    \csname locale@#2@#1\expandafter\endcsname
    \csname #3\endcsname
    #4%
  \else
    #5%
  \fi
}
```

leIfAttributeEqNum

\LocaleIfAttributeEqNum{⟨*label*⟩}{⟨*attribute label*⟩}{⟨*n*⟩}{⟨*true*⟩}{⟨*false*⟩}

If the numeric attribute value is equal to ⟨*n*⟩ do ⟨*true*⟩ otherwise do ⟨*false*⟩.

```
\def\LocaleIfAttributeEqNum#1#2#3#4#5{%
  \LocaleIfHasNonEmptyAttribute{#1}{#2}%
  {%
    \expandafter\ifnum\csname locale@#2@#1\endcsname=#3
      #4%
    \else
      #5%
    \fi
```

```
  }%
  {#5}%
}
```

ameAttributeValues

\LocaleIfSameAttributeValues{⟨*label*⟩}{⟨*attribute 1*⟩}{⟨*attribute 2*⟩}{⟨*true*⟩}{⟨*false*⟩}

Tests if two different attributes for the same ⟨*label*⟩ have matching values.

```
\def\LocaleIfSameAttributeValues#1#2#3#4#5{%
  \expandafter\ifx
    \csname locale@#2@#1\expandafter\endcsname
    \csname locale@#3@#1\endcsname
    #4%
  \else
    #5%
  \fi
}
```

wdialectattribute

Debugging command.

```
\def\localeshowdialectattribute#1#2{%
  \localeshowattribute{#1}{dialect@#2}%
}
```

etDialectAttribute

\LocaleSetDialectAttribute{⟨*dialect label*⟩}{⟨*attribute label*⟩}{⟨*attribute value*⟩}

Sets dialect attribute.

```
\def\LocaleSetDialectAttribute#1#2#3{%
  \LocaleSetAttribute{#1}{dialect@#2}{#3}%
}
```

deDialectAttribute

\LocaleProvideDialectAttribute{⟨*dialect label*⟩}{⟨*attribute label*⟩}{⟨*attribute value*⟩}

Provides dialect attribute.

```
\def\LocaleProvideDialectAttribute#1#2#3{%
  \LocaleProvideAttribute{#1}{dialect@#2}{#3}%
}
```

letDialectAttribute

\LocaleLetDialectAttribute{⟨*dialect label*⟩}{⟨*attribute label*⟩}{⟨*cs*⟩}

Set the dialect attribute value to the definition of the control sequence ⟨*cs*⟩.

```
\def\LocaleLetDialectAttribute#1#2#3{%
  \LocaleLetAttribute{#1}{dialect@#2}{#3}%
}
```

ToDialectAttribute

\LocaleAppToDialectAttribute{⟨*dialect label*⟩}{⟨*attribute label*⟩}{⟨*attribute value*⟩}

Append to dialect attribute value.

```
\def\LocaleAppToDialectAttribute#1#2#3{%
  \LocaleAppToAttribute{#1}{dialect@#2}{#3}%
}
```

ToDialectAttribute

\LocaleXpAppToDialectAttribute{⟨*dialect label*⟩}{⟨*attribute label*⟩}{⟨*value*⟩}

Append to dialect attribute value (expand first token of ⟨*value*⟩).

```
\def\LocaleXpAppToDialectAttribute#1#2#3{%
  \LocaleXpAppToAttribute{#1}{dialect@#2}{#3}%
}
```

alectAttributeList

\LocaleAddToDialectAttributeList{⟨*label*⟩}{⟨*attribute label*⟩}{⟨*value*⟩}

Adds ⟨*value*⟩ to this attribute's list (without repetition).

```
\def\LocaleAddToDialectAttributeList#1#2{%
  \LocaleAddToAttributeList{#1}{dialect@#2}%
}
```

alectAttributeList

\LocaleXpAddToDialectAttributeList{⟨*label*⟩}{⟨*attribute label*⟩}{⟨*value*⟩}

Adds ⟨*value*⟩ to this attribute's list (without repetition).

```
\def\LocaleXpAddToDialectAttributeList#1#2{%
  \LocaleXpAddToAttributeList{#1}{dialect@#2}%
}
```

etDialectAttribute

\LocaleGetDialectAttribute{⟨*dialect label*⟩}{⟨*attribute label*⟩}

Gets dialect attribute.

```
\def\LocaleGetDialectAttribute#1#2{%
   \LocaleGetAttribute{#1}{dialect@#2}%
}
```

ttributeOrDefValue

\LocaleGetDialectAttributeOrDefValue{⟨*dialect label*⟩}{⟨*attribute label*⟩}{⟨*def value*⟩}

Gets attribute for given dialect or ⟨*def value*⟩ if unset.

```
\def\LocaleGetDialectAttributeOrDefValue#1#2{%
   \LocaleGetAttributeOrDefValue{#1}{dialect@#2}%
}
```

asDialectAttribute

\LocaleIfHasDialectAttribute{⟨*dialect label*⟩}{⟨*attribute label*⟩}{⟨*true*⟩}{⟨*false*⟩}

```
\def\LocaleIfHasDialectAttribute#1#2{%
  \LocaleIfHasAttribute{#1}{dialect@#2}%
}
```

alectAttributeList

\LocaleForEachInDialectAttributeList{⟨*label*⟩}{⟨*attribute label*⟩}{⟨*cs*⟩}{⟨*body*⟩}

Where an attribute value is a comma-separated list, this iterates over each item in that list, setting ⟨*cs*⟩ to that item and performing ⟨*body*⟩.

```
\def\LocaleForEachInDialectAttributeList#1#2{%
  \LocaleForEachInAttributeList{#1}{dialect@#2}%
}
```

alectAttributeList

\LocaleIfInDialectAttributeList{⟨*label*⟩}{⟨*attribute label*⟩}{⟨*item*⟩}{⟨*true*⟩}{⟨*false*⟩}

Where an attribute value is a comma-separated list, this tests if ⟨*item*⟩ is in that list.

```
\def\LocaleIfInDialectAttributeList#1#2{%
  \LocaleIfInAttributeList{#1}{dialect@#2}%
}
```

alectAttributeList

\LocaleIfXpInDialectAttributeList{⟨*label*⟩}{⟨*attribute label*⟩}{⟨*item*⟩}{⟨*true*⟩}{⟨*false*⟩}

As above but expands the first token of ⟨*item*⟩.

```
\def\LocaleIfXpInDialectAttributeList#1#2{%
  \LocaleIfXpInAttributeList{#1}{dialect@#2}%
}
```

tNonEmptyAttribute

\LocaleIfHasDialectNonEmptyAttribute{⟨*dialect label*⟩}{⟨*attribute label*⟩}{⟨*true*⟩}{⟨*false*⟩}

```
\def\LocaleIfHasDialectNonEmptyAttribute#1#2{%
  \LocaleIfHasNonEmptyAttribute{#1}{dialect@#2}%
}
```

alectAttributeEqCs

\LocaleIfDialectAttributeEqCs{⟨*dialect label*⟩}{⟨*attribute label*⟩}{⟨*cs*⟩}{⟨*true*⟩}{⟨*false*⟩}

If the attribute value for the given dialect is the same as the definition of the control sequence ⟨*cs*⟩ do ⟨*true*⟩ otherwise do ⟨*false*⟩.

```
\def\LocaleIfDialectAttributeEqCs#1#2#3{%
  \LocaleIfAttributeEqCs{#1}{dialect@#2}{#3}%
}
```

tAttributeEqCsName

\LocaleIfDialectAttributeEqCsName{⟨*dialect label*⟩}{⟨*attribute label*⟩}{⟨*cs name*⟩}{⟨*true*⟩}{⟨*false*⟩}

If the attribute value for the given dialect is the same as the definition of the control sequence name ⟨*cs name*⟩ do ⟨*true*⟩ otherwise do ⟨*false*⟩.

```
\def\LocaleIfDialectAttributeEqCsName#1#2#3{%
  \LocaleIfAttributeEqCsName{#1}{dialect@#2}{#3}%
}
```

lectAttributeEqNum

\LocaleIfDialectAttributeEqNum{⟨*dialect label*⟩}{⟨*attribute label*⟩}{⟨*n*⟩}{⟨*true*⟩}{⟨*false*⟩}

If the numeric attribute value for the given dialect is equal to the number ⟨*n*⟩ do ⟨*true*⟩ otherwise do ⟨*false*⟩.

```
\def\LocaleIfDialectAttributeEqNum#1#2#3{%
  \LocaleIfAttributeEqNum{#1}{dialect@#2}{#3}%
}
```

ectAttributeValues

\LocaleIfSameDialectAttributeValues{⟨*dialect label*⟩}{⟨*attribute 1*⟩}{⟨*attribute 2*⟩}{⟨*true*⟩}{⟨*false*⟩}

Tests if two different dialect attributes for the same ⟨*dialect label*⟩ have matching values.

```
\def\LocaleIfSameDialectAttributeValues#1#2#3{%
  \LocaleIfSameAttributeValues{#1}{dialect@#2}{dialect@#3}%
}
```

owregionattribute

Debugging command.

```
\def\localeshowregionattribute#1#2{%
  \localeshowattribute{#1}{region@#2}%
}
```

SetRegionAttribute

\LocaleSetRegionAttribute{⟨*region code*⟩}{⟨*attribute label*⟩}{⟨*attribute value*⟩}

Sets region attribute.

```
\def\LocaleSetRegionAttribute#1#2#3{%
  \LocaleSetAttribute{#1}{region@#2}{#3}%
}
```

ideRegionAttribute

\LocaleProvideRegionAttribute{⟨*region code*⟩}{⟨*attribute label*⟩}{⟨*attribute value*⟩}

Provides region attribute.

```
\def\LocaleProvideRegionAttribute#1#2#3{%
  \LocaleProvideAttribute{#1}{region@#2}{#3}%
}
```

LetRegionAttribute

```
\LocaleLetRegionAttribute{⟨region code⟩}{⟨attribute label⟩}{⟨cs⟩}
```

Set the region attribute value to the definition of the control sequence ⟨*cs*⟩.

```
\def\LocaleLetRegionAttribute#1#2#3{%
  \LocaleLetAttribute{#1}{region@#2}{#3}%
}
```

pToRegionAttribute

```
\LocaleAppToRegionAttribute{⟨region code⟩}{⟨attribute label⟩}{⟨value⟩}
```

Append to region attribute.

```
\def\LocaleAppToRegionAttribute#1#2#3{%
  \LocaleAppToAttribute{#1}{region@#2}{#3}%
}
```

pToRegionAttribute

```
\LocaleXpAppToRegionAttribute{⟨region code⟩}{⟨attribute label⟩}{⟨value⟩}
```

Append to region attribute (expand first token of ⟨*value*⟩).

```
\def\LocaleXpAppToRegionAttribute#1#2#3{%
  \LocaleXpAppToAttribute{#1}{region@#2}{#3}%
}
```

egionAttributeList

```
\LocaleAddToRegionAttributeList{⟨label⟩}{⟨attribute label⟩}{⟨value⟩}
```

Adds ⟨*value*⟩ to this attribute's list (without repetition).

```
\def\LocaleAddToRegionAttributeList#1#2{%
  \LocaleAddToAttributeList{#1}{region@#2}%
}
```

egionAttributeList

```
\LocaleXpAddToRegionAttributeList{⟨label⟩}{⟨attribute label⟩}{⟨value⟩}
```

Adds ⟨*value*⟩ to this attribute's list (without repetition).

```
\def\LocaleXpAddToRegionAttributeList#1#2{%
  \LocaleXpAddToAttributeList{#1}{region@#2}%
}
```

GetRegionAttribute

```
\LocaleGetRegionAttribute{⟨region code⟩}{⟨attribute label⟩}
```

Gets region attribute.

```
\def\LocaleGetRegionAttribute#1#2{%
   \LocaleGetAttribute{#1}{region@#2}%
}
```

ttributeOrDefValue

```
\LocaleGetRegionAttributeOrDefValue{⟨region code⟩}{⟨attribute label⟩}{⟨def value⟩}
```

Gets attribute for given region or ⟨*def value*⟩ if unset.

```
\def\LocaleGetRegionAttributeOrDefValue#1#2{%
   \LocaleGetAttributeOrDefValue{#1}{region@#2}%
}
```

HasRegionAttribute

```
\LocaleIfHasRegionAttribute{⟨region code⟩}{⟨attribute label⟩}{⟨true⟩}{⟨false⟩}
```

```
\def\LocaleIfHasRegionAttribute#1#2{%
  \LocaleIfHasAttribute{#1}{region@#2}%
}
```

egionAttributeList

```
\LocaleForEachInRegionAttributeList{⟨label⟩}{⟨attribute label⟩}{⟨cs⟩}{⟨body⟩}
```

Where an attribute value is a comma-separated list, this iterates over each item in that list, setting ⟨*cs*⟩ to that item and performing ⟨*body*⟩.

```
\def\LocaleForEachInRegionAttributeList#1#2{%
  \LocaleForEachInAttributeList{#1}{region@#2}%
}
```

egionAttributeList

```
\LocaleIfInRegionAttributeList{⟨label⟩}{⟨attribute label⟩}{⟨item⟩}{⟨true⟩}
{⟨false⟩}
```

Where an attribute value is a comma-separated list, this tests if ⟨*item*⟩ is in that list.

```
\def\LocaleIfInRegionAttributeList#1#2{%
```

```
  \LocaleIfInAttributeList{#1}{region@#2}%
}
```

egionAttributeList

\LocaleIfXpInRegionAttributeList{⟨*label*⟩}{⟨*attribute label*⟩}{⟨*item*⟩}{⟨*true*⟩}{⟨*false*⟩}

As above but expands the first token of ⟨*item*⟩.

```
\def\LocaleIfXpInRegionAttributeList#1#2{%
  \LocaleIfXpInAttributeList{#1}{region@#2}%
}
```

nNonEmptyAttribute

\LocaleIfHasRegionNonEmptyAttribute{⟨*region code*⟩}{⟨*attribute label*⟩}{⟨*true*⟩}{⟨*false*⟩}

```
\def\LocaleIfHasRegionNonEmptyAttribute#1#2{%
  \LocaleIfHasNonEmptyAttribute{#1}{region@#2}%
}
```

egionAttributeEqCs

\LocaleIfRegionAttributeEqCs{⟨*region code*⟩}{⟨*attribute label*⟩}{⟨*cs*⟩}{⟨*true*⟩}{⟨*false*⟩}

If the attribute value for the given region is the same as the definition of the control sequence ⟨*cs*⟩ do ⟨*true*⟩ otherwise do ⟨*false*⟩.

```
\def\LocaleIfRegionAttributeEqCs#1#2#3{%
  \LocaleIfAttributeEqCs{#1}{region@#2}{#3}%
}
```

nAttributeEqCsName

\LocaleIfRegionAttributeEqCsName{⟨*region code*⟩}{⟨*attribute label*⟩}{⟨*cs name*⟩}{⟨*true*⟩}{⟨*false*⟩}

If the attribute value for the given region is the same as the definition of the control sequence name ⟨*cs name*⟩ do ⟨*true*⟩ otherwise do ⟨*false*⟩.

```
\def\LocaleIfRegionAttributeEqCsName#1#2#3{%
  \LocaleIfAttributeEqCsName{#1}{region@#2}{#3}%
}
```

gionAttributeEqNum

\LocaleIfRegionAttributeEqNum{⟨*region label*⟩}{⟨*attribute label*⟩}{⟨*n*⟩}{⟨*true*⟩}{⟨*false*⟩}

If the numeric attribute value for the given region is equal to the number ⟨*n*⟩ do ⟨*true*⟩ otherwise do ⟨*false*⟩.

```
\def\LocaleIfRegionAttributeEqNum#1#2#3{%
  \LocaleIfAttributeEqNum{#1}{region@#2}{#3}%
}
```

ionAttributeValues

\LocaleIfSameRegionAttributeValues{⟨*region code*⟩}{⟨*attribute 1*⟩}{⟨*attribute 2*⟩}{⟨*true*⟩}{⟨*false*⟩}

Tests if two different region attributes for the same ⟨*region code*⟩ have matching values.

```
\def\LocaleIfSameRegionAttributeValues#1#2#3{%
  \LocaleIfSameAttributeValues{#1}{region@#2}{region@#3}%
}
```

currencyattribute

Debugging command.

```
\def\localeshowcurrencyattribute#1#2{%
  \localeshowattribute{#1}{currency@#2}%
}
```

tCurrencyAttribute

\LocaleSetCurrencyAttribute{⟨*currency code*⟩}{⟨*attribute label*⟩}{⟨*attribute value*⟩}

Sets currency attribute.

```
\def\LocaleSetCurrencyAttribute#1#2#3{%
  \LocaleSetAttribute{#1}{currency@#2}{#3}%
}
```

eCurrencyAttribute

\LocaleProvideCurrencyAttribute{⟨*currency code*⟩}{⟨*attribute label*⟩}{⟨*attribute value*⟩}

Provides currency attribute.

```
\def\LocaleProvideCurrencyAttribute#1#2#3{%
  \LocaleProvideAttribute{#1}{currency@#2}{#3}%
}
```

tCurrencyAttribute `\LocaleLetCurrencyAttribute{⟨currency code⟩}{⟨attribute label⟩}{⟨cs⟩}`

Set the currency attribute value to the definition of the control sequence ⟨*cs*⟩.

```
\def\LocaleLetCurrencyAttribute#1#2#3{%
  \LocaleLetAttribute{#1}{currency@#2}{#3}%
}
```

oCurrencyAttribute `\LocaleAppToCurrencyAttribute{⟨currency code⟩}{⟨attribute label⟩}{⟨value⟩}`

Append to currency attribute value.

```
\def\LocaleAppToCurrencyAttribute#1#2#3{%
  \LocaleAppToAttribute{#1}{currency@#2}{#3}%
}
```

oCurrencyAttribute `\LocaleXpAppToCurrencyAttribute{⟨currency code⟩}{⟨attribute label⟩}{⟨value⟩}`

Append to currency attribute value (expand first token of ⟨*value*⟩).

```
\def\LocaleXpAppToCurrencyAttribute#1#2#3{%
  \LocaleXpAppToAttribute{#1}{currency@#2}{#3}%
}
```

rencyAttributeList `\LocaleAddToCurrencyAttributeList{⟨label⟩}{⟨attribute label⟩}{⟨value⟩}`

Adds ⟨*value*⟩ to this attribute's list (without repetition).

```
\def\LocaleAddToCurrencyAttributeList#1#2{%
  \LocaleAddToAttributeList{#1}{currency@#2}%
}
```

rencyAttributeList `\LocaleXpAddToCurrencyAttributeList{⟨label⟩}{⟨attribute label⟩}{⟨value⟩}`

Adds ⟨*value*⟩ to this attribute's list (without repetition).

```
\def\LocaleXpAddToCurrencyAttributeList#1#2{%
  \LocaleXpAddToAttributeList{#1}{currency@#2}%
}
```

tCurrencyAttribute

```
\LocaleGetCurrencyAttribute{⟨currency code⟩}{⟨attribute label⟩}
```

Gets currency attribute.

```
\def\LocaleGetCurrencyAttribute#1#2{%
   \LocaleGetAttribute{#1}{currency@#2}%
}
```

ttributeOrDefValue

```
\LocaleGetCurrencyAttributeOrDefValue{⟨currency code⟩}{⟨attribute label⟩}{⟨def value⟩}
```

Gets attribute for given currency or ⟨*def value*⟩ if unset.

```
\def\LocaleGetCurrencyAttributeOrDefValue#1#2{%
   \LocaleGetAttributeOrDefValue{#1}{currency@#2}%
}
```

sCurrencyAttribute

```
\LocaleIfHasCurrencyAttribute{⟨currency code⟩}{⟨attribute label⟩}{⟨true⟩}
{⟨false⟩}
```

```
\def\LocaleIfHasCurrencyAttribute#1#2{%
  \LocaleIfHasAttribute{#1}{currency@#2}%
}
```

rencyAttributeList

```
\LocaleForEachInCurrencyAttributeList{⟨label⟩}{⟨attribute label⟩}{⟨cs⟩}{⟨body⟩}
```

Where an attribute value is a comma-separated list, this iterates over each item in that list, setting ⟨*cs*⟩ to that item and performing ⟨*body*⟩.

```
\def\LocaleForEachInCurrencyAttributeList#1#2{%
  \LocaleForEachInAttributeList{#1}{currency@#2}%
}
```

rencyAttributeList

```
\LocaleIfInCurrencyAttributeList{⟨label⟩}{⟨attribute label⟩}{⟨item⟩}{⟨true⟩}
{⟨false⟩}
```

Where an attribute value is a comma-separated list, this tests if ⟨*item*⟩ is in that list.

```
\def\LocaleIfInCurrencyAttributeList#1#2{%
  \LocaleIfInAttributeList{#1}{currency@#2}%
}
```

rencyAttributeList

\LocaleIfXpInCurrencyAttributeList{⟨*label*⟩}{⟨*attribute label*⟩}{⟨*item*⟩}{⟨*true*⟩}{⟨*false*⟩}

As above but expands the first token of ⟨*item*⟩.

```
\def\LocaleIfXpInCurrencyAttributeList#1#2{%
  \LocaleIfXpInAttributeList{#1}{currency@#2}%
}
```

yNonEmptyAttribute

\LocaleIfHasCurrencyNonEmptyAttribute{⟨*currency code*⟩}{⟨*attribute label*⟩}{⟨*true*⟩}{⟨*false*⟩}

```
\def\LocaleIfHasCurrencyNonEmptyAttribute#1#2{%
  \LocaleIfHasNonEmptyAttribute{#1}{currency@#2}%
}
```

rencyAttributeEqCs

\LocaleIfCurrencyAttributeEqCs{⟨*currency code*⟩}{⟨*attribute label*⟩}{⟨*cs*⟩}{⟨*true*⟩}{⟨*false*⟩}

If the attribute value for the given currency is the same as the definition of the control sequence ⟨*cs*⟩ do ⟨*true*⟩ otherwise do ⟨*false*⟩.

```
\def\LocaleIfCurrencyAttributeEqCs#1#2#3{%
  \LocaleIfAttributeEqCs{#1}{currency@#2}{#3}%
}
```

yAttributeEqCsName

\LocaleIfCurrencyAttributeEqCsName{⟨*currency code*⟩}{⟨*attribute label*⟩}{⟨*cs name*⟩}{⟨*true*⟩}{⟨*false*⟩}

If the attribute value for the given currency is the same as the definition of the control sequence name ⟨*cs name*⟩ do ⟨*true*⟩ otherwise do ⟨*false*⟩.

```
\def\LocaleIfCurrencyAttributeEqCsName#1#2#3{%
  \LocaleIfAttributeEqCsName{#1}{currency@#2}{#3}%
}
```

encyAttributeEqNum

\LocaleIfCurrencyAttributeEqNum{⟨*currency label*⟩}{⟨*attribute label*⟩}{⟨*n*⟩}{⟨*true*⟩}{⟨*false*⟩}

If the numeric attribute value for the given currency is equal to the number ⟨*n*⟩ do ⟨*true*⟩ otherwise do ⟨*false*⟩.

```
\def\LocaleIfCurrencyAttributeEqNum#1#2#3{%
  \LocaleIfAttributeEqNum{#1}{currency@#2}{#3}%
}
```

ncyAttributeValues

\LocaleIfSameCurrencyAttributeValues{⟨*currency code*⟩}{⟨*attribute 1*⟩}{⟨*attribute 2*⟩}{⟨*true*⟩}{⟨*false*⟩}

Tests if two different currency attributes for the same ⟨*currency code*⟩ have matching values.

```
\def\LocaleIfSameCurrencyAttributeValues#1#2#3{%
  \LocaleIfSameAttributeValues{#1}{currency@#2}{currency@#3}%
}
```

4.2.4 Parser Commands

Define commands that are needed to parse the result.

cale@parse@result

Start parsing the result. There are more than nine arguments, so do this in bits. The first six arguments are: OS name, OS version, OS architecture, PDF date-time, BCP 47 tag for default locale, the default file encoding.

```
\def\@locale@parse@result#1#2#3#4#5#6{%
  \def\LocaleOSname{#1}%
  \def\LocaleOSversion{#2}%
  \def\LocaleOSarch{#3}%
  \def\LocaleNowStamp{#4}%
  \@locale@parse@default#5% remove outer group
  \def\LocaleOScodeset{#6}%
```

Is date-time pattern support required?

```
  \LocaleIfDateTimePatternsSupported
  {%
     \let\@locale@next\@locale@parse@datetimeinfo
  }%
  {%
     \def\LocaleDateTimeInfo{}%
```

Was a file modification date included?

```
     \ifx\LocaleMainFile\empty
```

```
      \def\LocaleFileMod{}%
      \let\@locale@next\@locale@parse@maindata
    \else
      \let\@locale@next\@locale@parse@filemod
    \fi
  }%
  \@locale@next
}
```

arse@datetimeinfo

```
\def\@locale@parse@datetimeinfo#1{%
  \def\LocaleDateTimeInfo{#1}%
```

Was a file modification date included?

```
  \ifx\LocaleMainFile\empty
    \def\LocaleFileMod{}%
    \let\@locale@next\@locale@parse@maindata
  \else
    \let\@locale@next\@locale@parse@filemod
  \fi
  \@locale@next
}
```

ale@parse@default Parse the result of -N

```
\def\@locale@parse@default#1#2#3#4#5#6#7#8#9{%
  \def\LocaleOStag{#1}%
```

Parse (but don't track) tag.

```
  \@tracklang@parselangtag{#1}%
```

Can now get the region code.

```
  \let\@locale@os@region\@TrackLangEnvTerritory
  \def\@locale@os@groupsep{#2}%
  \def\@locale@os@decsep{#3}%
  \def\@locale@os@exp{#4}%
  \def\@locale@os@usesgroup{#5}%
  \def\@locale@os@currencycode{#6}%
  \def\@locale@os@regionalcurrencycode{#7}%
  \def\@locale@os@currencysym{#8}%
  \def\@locale@os@currencytex{#9}%
```

Provide currency attributes.

```
  \LocaleSetRegionAttribute{\@locale@os@region}{currency}{#7}%
  \LocaleProvideCurrencyAttribute{#7}{official}{#6}%
  \LocaleProvideCurrencyAttribute{#7}{sym}{#8}%
  \LocaleProvideCurrencyAttribute{#7}{tex}{#9}%
```

```
    \@locale@parse@default@cursep
  }

rse@default@cursep
  \def\@locale@parse@default@cursep#1{%
    \def\@locale@os@cursep{#1}%
  }

ale@parse@filemod
  \def\@locale@parse@filemod#1{%
    \def\LocaleFileMod{#1}%
    \@locale@parse@maindata
  }

le@parse@maindata
  \def\@locale@parse@maindata#1{%
    \@locale@parse@maindatablock#1% remove outer group
  }

rse@maindatablock
  \def\@locale@parse@maindatablock#1{%
    \@locale@parse@maindatalocaleblock#1% remove outer group
    \@locale@parse@dateblock
  }

indatalocaleblock
  \def\@locale@parse@maindatalocaleblock#1#2#3#4#5#6#7{%
    \def\LocaleMain{#1}%
    \TrackLanguageTag{#1}%
    \let\LocaleMainDialect\TrackLangLastTrackedDialect
    \let\@locale@dialect\TrackLangLastTrackedDialect
```

Get the region code if provided.

```
    \IfTrackedLanguageHasIsoCode{3166-1}{\@locale@dialect}%
    {%
      \edef\@locale@region{%
       \TrackedIsoCodeFromLanguage{3166-1}{\@locale@dialect}}%
    }%
    {\def\@locale@region{}}%
    \let\LocaleMainRegion\@locale@region
    \ifx\LocaleMainRegion\empty
    \else
      \LocaleLetRegionAttribute{\LocaleMainRegion}{dialect}{\@locale@dialect}%
    \fi
```

Save language tag. (Provides a convenient mapping from dialect label to tag.)

```
    \LocaleSetDialectAttribute{\@locale@dialect}{langtag}{#1}%
```

Provide reverse mapping from tag to dialect label.

```
    \LocaleLetAttribute{#1}{tagtodialect}{\@locale@dialect}%
```

Save display names.

```
    \LocaleSetDialectAttribute{\@locale@dialect}{langname}{#2}%
    \LocaleSetDialectAttribute{\@locale@dialect}{nativelangname}{#3}%
    \LocaleSetDialectAttribute{\@locale@dialect}{regionname}{#4}%
    \LocaleSetDialectAttribute{\@locale@dialect}{nativeregionname}{#5}%
    \LocaleSetDialectAttribute{\@locale@dialect}{variantname}{#6}%
    \LocaleSetDialectAttribute{\@locale@dialect}{nativevariantname}{#7}%
  }
```

erdatalocaleblock

```
  \def\@locale@parse@otherdatalocaleblock#1#2#3#4#5#6#7{%
    \TrackLanguageTag{#1}%
    \let\@locale@dialect\TrackLangLastTrackedDialect
```

Get the region code if provided.

```
    \IfTrackedLanguageHasIsoCode{3166-1}{\@locale@dialect}%
    {%
      \edef\@locale@region{%
       \TrackedIsoCodeFromLanguage{3166-1}{\@locale@dialect}}%
       \ifx\@locale@region\empty
       \else
         \LocaleXpAddToRegionAttributeList
           {\@locale@region}{dialect}{\@locale@dialect}%
       \fi
    }%
    {\def\@locale@region{}}%
```

Save language tag.

```
    \LocaleSetDialectAttribute{\@locale@dialect}{langtag}{#1}%
```

Provide reverse mapping from tag to dialect label.

```
    \LocaleLetAttribute{#1}{tagtodialect}{\@locale@dialect}%
```

Save attributes.

```
    \LocaleSetDialectAttribute{\@locale@dialect}{langname}{#2}%
    \LocaleSetDialectAttribute{\@locale@dialect}{nativelangname}{#3}%
    \LocaleSetDialectAttribute{\@locale@dialect}{regionname}{#4}%
    \LocaleSetDialectAttribute{\@locale@dialect}{nativeregionname}{#5}%
    \LocaleSetDialectAttribute{\@locale@dialect}{variantname}{#6}%
    \LocaleSetDialectAttribute{\@locale@dialect}{nativevariantname}{#7}%
  }
```

```
e@parse@dateblock
                    \def\@locale@parse@dateblock#1{%
                      \@locale@parse@dates#1% remove outer group
                      \@locale@parse@datefmtblock
                    }

ocale@parse@dates
                    \def\@locale@parse@dates#1#2#3#4#5{%
                      \LocaleSetDialectAttribute{\@locale@dialect}{fulldate}{#1}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{longdate}{#2}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{meddate}{#3}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{shortdate}{#4}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{firstday}{#5}%
                    }

arse@datefmtblock
                    \def\@locale@parse@datefmtblock#1{%
                      \@locale@parse@datefmts#1% remove outer group
                      \@locale@parse@timeblock
                    }

le@parse@datefmts
                    \def\@locale@parse@datefmts#1#2#3#4{%
                      \LocaleSetDialectAttribute{\@locale@dialect}{fulldatefmt}{#1}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{longdatefmt}{#2}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{meddatefmt}{#3}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{shortdatefmt}{#4}%
                    }

e@parse@timeblock
                    \def\@locale@parse@timeblock#1{%
                      \@locale@parse@times#1% remove outer group
                      \@locale@parse@timefmtblock
                    }

ocale@parse@times
                    \def\@locale@parse@times#1#2#3#4{%
                      \LocaleSetDialectAttribute{\@locale@dialect}{fulltime}{#1}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{longtime}{#2}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{medtime}{#3}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{shorttime}{#4}%
                    }
```

```
parse@timefmtblock
                    \def\@locale@parse@timefmtblock#1{%
                      \@locale@parse@timefmts#1% remove outer group
                      \@locale@parse@datetimeblock
                    }

le@parse@timefmts
                    \def\@locale@parse@timefmts#1#2#3#4{%
                      \LocaleSetDialectAttribute{\@locale@dialect}{fulltimefmt}{#1}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{longtimefmt}{#2}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{medtimefmt}{#3}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{shorttimefmt}{#3}%
                    }

rse@datetimeblock
                    \def\@locale@parse@datetimeblock#1{%
                      \@locale@parse@datetimes#1% remove outer group
                      \@locale@parse@datetimefmtblock
                    }

e@parse@datetimes
                    \def\@locale@parse@datetimes#1#2#3#4{%
                      \LocaleSetDialectAttribute{\@locale@dialect}{fulldatetime}{#1}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{longdatetime}{#2}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{meddatetime}{#3}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{shortdatetime}{#4}%
                    }

e@datetimefmtblock
                    \def\@locale@parse@datetimefmtblock#1{%
                      \@locale@parse@datetimefmts#1% remove outer group
                      \@locale@parse@weekdayblock
                    }

parse@datetimefmts
                    \def\@locale@parse@datetimefmts#1#2#3#4{%
                      \LocaleSetDialectAttribute{\@locale@dialect}{fulldatetimefmt}{#1}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{longdatetimefmt}{#2}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{meddatetimefmt}{#3}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{shortdatetimefmt}{#4}%
                    }

parse@weekdayblock
                    \def\@locale@parse@weekdayblock#1{%
```

```
    \@locale@parse@weekdays#1% remove outer group
    \@locale@parse@shortweekdayblock
  }

le@parse@weekdays

  \def\@locale@parse@weekdays#1#2#3#4#5#6#7{%
    \LocaleSetDialectAttribute{\@locale@dialect}{day.0}{#1}%
    \LocaleSetDialectAttribute{\@locale@dialect}{day.1}{#2}%
    \LocaleSetDialectAttribute{\@locale@dialect}{day.2}{#3}%
    \LocaleSetDialectAttribute{\@locale@dialect}{day.3}{#4}%
    \LocaleSetDialectAttribute{\@locale@dialect}{day.4}{#5}%
    \LocaleSetDialectAttribute{\@locale@dialect}{day.5}{#6}%
    \LocaleSetDialectAttribute{\@locale@dialect}{day.6}{#7}%
  }

@shortweekdayblock

  \def\@locale@parse@shortweekdayblock#1{%
    \@locale@parse@shortweekdays#1% remove outer group
    \@locale@parse@monthblock
  }

rse@shortweekdays

  \def\@locale@parse@shortweekdays#1#2#3#4#5#6#7{%
    \LocaleSetDialectAttribute{\@locale@dialect}{shortday.0}{#1}%
    \LocaleSetDialectAttribute{\@locale@dialect}{shortday.1}{#2}%
    \LocaleSetDialectAttribute{\@locale@dialect}{shortday.2}{#3}%
    \LocaleSetDialectAttribute{\@locale@dialect}{shortday.3}{#4}%
    \LocaleSetDialectAttribute{\@locale@dialect}{shortday.4}{#5}%
    \LocaleSetDialectAttribute{\@locale@dialect}{shortday.5}{#6}%
    \LocaleSetDialectAttribute{\@locale@dialect}{shortday.6}{#7}%
  }

@parse@monthblock

  \def\@locale@parse@monthblock#1{%
    \@locale@parse@months#1% remove outer group
    \@locale@parse@shortmonthblock
  }

cale@parse@months

  \def\@locale@parse@months#1#2#3#4#5#6#7#8#9{%
    \LocaleSetDialectAttribute{\@locale@dialect}{month.1}{#1}%
    \LocaleSetDialectAttribute{\@locale@dialect}{month.2}{#2}%
    \LocaleSetDialectAttribute{\@locale@dialect}{month.3}{#3}%
    \LocaleSetDialectAttribute{\@locale@dialect}{month.4}{#4}%
```

```
    \LocaleSetDialectAttribute{\@locale@dialect}{month.5}{#5}%
    \LocaleSetDialectAttribute{\@locale@dialect}{month.6}{#6}%
    \LocaleSetDialectAttribute{\@locale@dialect}{month.7}{#7}%
    \LocaleSetDialectAttribute{\@locale@dialect}{month.8}{#8}%
    \LocaleSetDialectAttribute{\@locale@dialect}{month.9}{#9}%
```

Grab the remaining three arguments:

```
    \@locale@parse@endmonths
  }
```

```
e@parse@endmonths
  \def\@locale@parse@endmonths#1#2#3{%
    \LocaleSetDialectAttribute{\@locale@dialect}{month.10}{#1}%
    \LocaleSetDialectAttribute{\@locale@dialect}{month.11}{#2}%
    \LocaleSetDialectAttribute{\@locale@dialect}{month.12}{#3}%
  }
```

```
e@shortmonthblock
  \def\@locale@parse@shortmonthblock#1{%
    \@locale@parse@shortmonths#1% remove outer group
    \@locale@parse@standalone@weekdayblock
  }
```

```
parse@shortmonths
  \def\@locale@parse@shortmonths#1#2#3#4#5#6#7#8#9{%
    \LocaleSetDialectAttribute{\@locale@dialect}{shortmonth.1}{#1}%
    \LocaleSetDialectAttribute{\@locale@dialect}{shortmonth.2}{#2}%
    \LocaleSetDialectAttribute{\@locale@dialect}{shortmonth.3}{#3}%
    \LocaleSetDialectAttribute{\@locale@dialect}{shortmonth.4}{#4}%
    \LocaleSetDialectAttribute{\@locale@dialect}{shortmonth.5}{#5}%
    \LocaleSetDialectAttribute{\@locale@dialect}{shortmonth.6}{#6}%
    \LocaleSetDialectAttribute{\@locale@dialect}{shortmonth.7}{#7}%
    \LocaleSetDialectAttribute{\@locale@dialect}{shortmonth.8}{#8}%
    \LocaleSetDialectAttribute{\@locale@dialect}{shortmonth.9}{#9}%
```

Grab the remaining three arguments:

```
    \@locale@parse@endshortmonths
  }
```

```
rse@endshortmonths
  \def\@locale@parse@endshortmonths#1#2#3{%
    \LocaleSetDialectAttribute{\@locale@dialect}{shortmonth.10}{#1}%
    \LocaleSetDialectAttribute{\@locale@dialect}{shortmonth.11}{#2}%
    \LocaleSetDialectAttribute{\@locale@dialect}{shortmonth.12}{#3}%
  }
```

```
lone@weekdayblock
                    \def\@locale@parse@standalone@weekdayblock#1{%
                      \@locale@parse@standalone@weekdays#1% remove outer group
                      \@locale@parse@standalone@shortweekdayblock
                    }

andalone@weekdays
                    \def\@locale@parse@standalone@weekdays#1#2#3#4#5#6#7{%
                      \LocaleSetDialectAttribute{\@locale@dialect}{standalone.day.0}{#1}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{standalone.day.1}{#2}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{standalone.day.2}{#3}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{standalone.day.3}{#4}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{standalone.day.4}{#5}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{standalone.day.5}{#6}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{standalone.day.6}{#7}%
                    }

@shortweekdayblock
                    \def\@locale@parse@standalone@shortweekdayblock#1{%
                      \@locale@parse@standalone@shortweekdays#1% remove outer group
                      \@locale@parse@standalone@monthblock
                    }

one@shortweekdays
                    \def\@locale@parse@standalone@shortweekdays#1#2#3#4#5#6#7{%
                      \LocaleSetDialectAttribute{\@locale@dialect}{standalone.shortday.0}{#1}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{standalone.shortday.1}{#2}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{standalone.shortday.2}{#3}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{standalone.shortday.3}{#4}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{standalone.shortday.4}{#5}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{standalone.shortday.5}{#6}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{standalone.shortday.6}{#7}%
                    }

ndalone@monthblock
                    \def\@locale@parse@standalone@monthblock#1{%
                      \@locale@parse@standalone@months#1% remove outer group
                      \@locale@parse@standalone@shortmonthblock
                    }

@standalone@months
                    \def\@locale@parse@standalone@months#1#2#3#4#5#6#7#8#9{%
                      \LocaleSetDialectAttribute{\@locale@dialect}{standalone.month.1}{#1}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{standalone.month.2}{#2}%
```

```
    \LocaleSetDialectAttribute{\@locale@dialect}{standalone.month.3}{#3}%
    \LocaleSetDialectAttribute{\@locale@dialect}{standalone.month.4}{#4}%
    \LocaleSetDialectAttribute{\@locale@dialect}{standalone.month.5}{#5}%
    \LocaleSetDialectAttribute{\@locale@dialect}{standalone.month.6}{#6}%
    \LocaleSetDialectAttribute{\@locale@dialect}{standalone.month.7}{#7}%
    \LocaleSetDialectAttribute{\@locale@dialect}{standalone.month.8}{#8}%
    \LocaleSetDialectAttribute{\@locale@dialect}{standalone.month.9}{#9}%
```

Grab the remaining three arguments:

```
    \@locale@parse@endstandalone@months
  }

standalone@months

  \def\@locale@parse@endstandalone@months#1#2#3{%
    \LocaleSetDialectAttribute{\@locale@dialect}{standalone.month.10}{#1}%
    \LocaleSetDialectAttribute{\@locale@dialect}{standalone.month.11}{#2}%
    \LocaleSetDialectAttribute{\@locale@dialect}{standalone.month.12}{#3}%
  }

e@shortmonthblock

  \def\@locale@parse@standalone@shortmonthblock#1{%
    \@locale@parse@standalone@shortmonths#1% remove outer group
    \@locale@parse@numericblock
  }

alone@shortmonths

  \def\@locale@parse@standalone@shortmonths#1#2#3#4#5#6#7#8#9{%
    \LocaleSetDialectAttribute{\@locale@dialect}{standalone.shortmonth.1}{#1}%
    \LocaleSetDialectAttribute{\@locale@dialect}{standalone.shortmonth.2}{#2}%
    \LocaleSetDialectAttribute{\@locale@dialect}{standalone.shortmonth.3}{#3}%
    \LocaleSetDialectAttribute{\@locale@dialect}{standalone.shortmonth.4}{#4}%
    \LocaleSetDialectAttribute{\@locale@dialect}{standalone.shortmonth.5}{#5}%
    \LocaleSetDialectAttribute{\@locale@dialect}{standalone.shortmonth.6}{#6}%
    \LocaleSetDialectAttribute{\@locale@dialect}{standalone.shortmonth.7}{#7}%
    \LocaleSetDialectAttribute{\@locale@dialect}{standalone.shortmonth.8}{#8}%
    \LocaleSetDialectAttribute{\@locale@dialect}{standalone.shortmonth.9}{#9}%
```

Grab the remaining three arguments:

```
    \@locale@parse@endstandalone@shortmonths
  }

alone@shortmonths

  \def\@locale@parse@endstandalone@shortmonths#1#2#3{%
    \LocaleSetDialectAttribute{\@locale@dialect}{standalone.shortmonth.10}{#1}%
    \LocaleSetDialectAttribute{\@locale@dialect}{standalone.shortmonth.11}{#2}%
```

```
    \LocaleSetDialectAttribute{\@locale@dialect}{standalone.shortmonth.12}{#3}%
  }

parse@numericblock

  \def\@locale@parse@numericblock#1{%
    \@locale@parse@numeric#1% remove outer group
    \@locale@parse@numericfmtblock
  }

ale@parse@numeric

  \def\@locale@parse@numeric#1#2#3#4#5#6#7#8#9{%
    \LocaleSetDialectAttribute{\@locale@dialect}{groupsep}{#1}%
    \LocaleSetDialectAttribute{\@locale@dialect}{decsep}{#2}%
    \LocaleSetDialectAttribute{\@locale@dialect}{exp}{#3}%
    \LocaleSetDialectAttribute{\@locale@dialect}{usesgroup}{#4}%
    \LocaleSetDialectAttribute{\@locale@dialect}{currency}{#5}%
    \LocaleSetDialectAttribute{\@locale@dialect}{regionalcurrency}{#6}%
    \LocaleSetDialectAttribute{\@locale@dialect}{currencysym}{#7}%
    \LocaleSetDialectAttribute{\@locale@dialect}{currencytex}{#8}%
    \LocaleSetDialectAttribute{\@locale@dialect}{currencysep}{#9}%
```

Set currency attributes.

```
    \LocaleProvideCurrencyAttribute{#6}{official}{#5}%
    \LocaleProvideCurrencyAttribute{#6}{sym}{#7}%
    \LocaleProvideCurrencyAttribute{#6}{tex}{#8}%
    \LocaleAddToAttributeList{currencies}{official}{#5}%
    \LocaleAddToAttributeList{currencies}{regional}{#6}%
```

If this dialect has an associated region, map the region to the currency code.

```
    \ifx\@locale@region\empty
    \else
      \LocaleProvideRegionAttribute{\@locale@region}{currency}{#6}%
      \LocaleXpAddToCurrencyAttributeList{#5}{region}{\@locale@region}%
      \LocaleIfSameDialectAttributeValues{\@locale@dialect}%
      {regionalcurrency}{currency}%
      {}%
      {%
        \LocaleXpAddToCurrencyAttributeList{#6}{region}{\@locale@region}%
      }%
    \fi
```

Grab remaining arguments.

```
    \@locale@parse@persym
  }
```

```
cale@parse@persym
                    \def\@locale@parse@persym#1#2{%
                      \LocaleSetDialectAttribute{\@locale@dialect}{percent}{#1}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{permill}{#2}%
                    }

e@numericfmtblock
                    \def\@locale@parse@numericfmtblock#1{%
                      \@locale@parse@numericfmt#1% remove outer group
                      \LocaleIfDateTimePatternsSupported
                      {\@locale@parse@timezones}%
                      {\@locale@parse@otherdata}%
                    }

@parse@numericfmt
                    \def\@locale@parse@numericfmt#1#2#3#4{%
                      \LocaleSetDialectAttribute{\@locale@dialect}{decfmt}{#1}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{intfmt}{#2}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{curfmt}{#3}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{perfmt}{#4}%
                    }

e@parse@timezones
                    \def\@locale@parse@timezones#1{%
                      \@locale@parse@timezonemap#1\relax
                      \@locale@parse@otherdata
                    }

parse@timezonemap
                    \def\@locale@parse@timezonemap#1{%
                      \ifx\relax#1\relax
                        \let\@locale@next\relax
                      \else
                        \@locale@save@timezonemap#1%
                        \let\@locale@next\@locale@parse@timezonemap
                      \fi
                      \@locale@next
                    }%

@save@timezonemap
                    \def\@locale@save@timezonemap#1#2#3#4#5{%
                      \LocaleSetDialectAttribute{\@locale@dialect}{timezone.#1.short}{#2}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{timezone.#1.long}{#3}%
                      \LocaleSetDialectAttribute{\@locale@dialect}{timezone.#1.shortdst}{#4}%
```

```
    \LocaleSetDialectAttribute{\@locale@dialect}{timezone.#1.longdst}{#5}%
    \LocaleAddToAttributeList{timezone}{id}{#1}%
  }

e@parse@otherdata

  \def\@locale@endparse@result{}

e@parse@otherdata

  \def\@locale@parse@otherdata#1{%
   \ifx\@locale@endparse@result#1\relax
```

Finished.

```
    \let\@locale@next\relax
   \else
    \def\@locale@next{\@locale@parse@otherdata@localeblock#1}%
   \fi
   \@locale@next
  }

rdata@localeblock

  \def\@locale@parse@otherdata@localeblock#1{%
    \@locale@parse@otherdatalocaleblock#1% remove outer group
    \@locale@parse@dateblock
  }

  \ifx\@locale@result\empty
```

Set defaults.

```
    \ifTeXOSQueryDryRun
      \@locale@warn{Dry run mode on. No data provided by texosquery. Check
      TeX's shell escape status}%
    \else
    \@locale@warn{No data provided by texosquery. Check
      TeX's shell escape status and texosquery's setup}%
    \fi
    \def\LocaleOSname{}%
    \def\LocaleOSversion{}%
    \def\LocaleOSarch{}%
    \def\LocaleNowStamp{}%
    \def\LocaleOStag{}%
    \def\LocaleOScodeset{}%
    \def\LocaleFileMod{}%
    \def\LocaleMain{}%
    \def\LocaleMainDialect{}%
    \def\LocaleMainRegion{}%
```

```
  \def\LocaleDateTimeInfo{}%
\else
  \expandafter\@locale@parse@result\@locale@result\@locale@endparse@result
\fi
```

4.2.5 Locale User Commands

\LocaleLanguageTag

\LocaleLanguageTag{⟨*dialect*⟩}

Get the language tag of the given dialect.

```
\def\LocaleLanguageTag#1{%
 \LocaleGetDialectAttribute{#1}{langtag}%
}
```

LocaleLanguageName

\LocaleLanguageName{⟨*dialect*⟩}

Get the language name of the given dialect according to the OS default language.

```
\def\LocaleLanguageName#1{%
 \LocaleGetDialectAttribute{#1}{langname}%
}
```

LanguageNativeName

\LocaleLanguageNativeName{⟨*dialect*⟩}

Get the language name of the given dialect according to that dialect.

```
\def\LocaleLanguageNativeName#1{%
 \LocaleGetDialectAttribute{#1}{nativelangname}%
}
```

\LocaleRegionName

\LocaleRegionName{⟨*dialect*⟩}

Get the region name of the given dialect according to the OS default language.

```
\def\LocaleRegionName#1{%
 \LocaleGetDialectAttribute{#1}{regionname}%
}
```

leRegionNativeName `\LocaleRegionNativeName{⟨dialect⟩}`

Get the region name of the given dialect according to that dialect.

```
\def\LocaleRegionNativeName#1{%
 \LocaleGetDialectAttribute{#1}{nativeregionname}%
}
```

\LocaleVariantName `\LocaleVariantName{⟨dialect⟩}`

Get the variant name of the given dialect according to the OS default language.

```
\def\LocaleVariantName#1{%
 \LocaleGetDialectAttribute{#1}{variantname}%
}
```

eVariantNativeName `\LocaleVariantNativeName{⟨dialect⟩}`

Get the variant name of the given dialect according to that dialect.

```
\def\LocaleVariantNativeName#1{%
 \LocaleGetDialectAttribute{#1}{nativevariantname}%
}
```

\LocaleFullDate `\LocaleFullDate{⟨dialect⟩}`

Get the full date for the given dialect.

```
\def\LocaleFullDate#1{%
 \localedatetimefmt{\LocaleGetDialectAttribute{#1}{fulldate}}%
}
```

\LocaleLongDate `\LocaleLongDate{⟨dialect⟩}`

Get the long date for the given dialect.

```
\def\LocaleLongDate#1{%
 \localedatetimefmt{\LocaleGetDialectAttribute{#1}{longdate}}%
}
```

\LocaleMediumDate

\LocaleMediumDate{⟨*dialect*⟩}

Get the medium date for the given dialect.

```
\def\LocaleMediumDate#1{%
 \localedatetimefmt{\LocaleGetDialectAttribute{#1}{meddate}}%
}
```

\LocaleShortDate

\LocaleShortDate{⟨*dialect*⟩}

Get the short date for the given dialect.

```
\def\LocaleShortDate#1{%
 \localedatetimefmt{\LocaleGetDialectAttribute{#1}{shortdate}}%
}
```

ocaleFirstDayIndex

\LocaleFirstDayIndex{⟨*dialect*⟩}

Get the index for the first day of the week for the given dialect. This starts with 0 for Monday to be compatible with pgfcalendar (and datetime2 which similar does this to be compatible with pgfcalendar).

```
\def\LocaleFirstDayIndex#1{%
 \LocaleGetDialectAttributeOrDefValue{#1}{firstday}{-1}%
}
```

\dtmMondayIndex

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

\dtmTuesdayIndex

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

dtmWednesdayIndex

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

\dtmThursdayIndex

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

\dtmFridayIndex

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

\dtmSaturdayIndex

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

\dtmSundayIndex

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

The ISO-8601 standard has day of week indexing starting with 1 for Monday. The datetime package has day of week indexing starting with 1 for Sunday, so provide convenient ways to convert if necessary.

omZeroMonToOneSun Convert from pgfcalendar's Monday=0 indexing to datetime's Sunday=1 indexing.

```
\def\LocaleDayIndexFromZeroMonToOneSun#1{%
  \ifcase#1
  2% Monday 0 -> 2
  \or
  3% Tuesday 1 -> 3
  \or
  4% Wednesday 2 -> 4
  \or
  5% Thursday 3 -> 5
  \or
  6% Friday 4 -> 6
  \or
  7% Saturday 5 -> 7
  \or
  1% Sunday 6 -> 1
  \else
  -1% invalid
  \fi
}
```

omZeroMonToOneMon Convert from pgfcalendar's Monday=0 indexing to ISO-8601's Monday=1 indexing.

```
\def\LocaleDayIndexFromZeroMonToOneMon#1{%
  \ifcase#1
  1% Monday 0 -> 1
  \or
  2% Tuesday 1 -> 2
  \or
  3% Wednesday 2 -> 3
  \or
  4% Thursday 3 -> 4
  \or
  5% Friday 4 -> 5
  \or
```

```
  6% Saturday 5 -> 6
  \or
  7% Sunday 6 -> 7
  \else
  -1% invalid
  \fi
}
```

romOneSunToZeroMon Convert from datetime's Sunday=1 indexing to pgfcalendar's Monday=0 indexing.

```
\def\LocaleDayIndexFromOneSunToZeroMon#1{%
  \ifcase#1
  % no 0
  \or
  6% Sunday 1 -> 6
  \or
  0% Monday 2 -> 0
  \or
  1% Tuesday 3 -> 1
  \or
  2% Wednesday 4 -> 2
  \or
  3% Thursday 5 -> 3
  \or
  4% Friday 6 -> 4
  \or
  5% Saturday 7 -> 5
  \else
  -1% invalid
  \fi
}
```

romOneMonToZeroMon Convert from ISO-8601's Monday=1 indexing to pgfcalendar's Monday=0 indexing.

```
\def\LocaleDayIndexFromOneMonToZeroMon#1{%
  \ifcase#1
  % no 0
  \or
  0% Monday 1 -> 0
  \or
  1% Tuesday 2 -> 1
  \or
  2% Wednesday 3 -> 2
  \or
  3% Thursday 4 -> 3
  \or
```

```
    4% Friday 5 -> 4
    \or
    5% Saturday 6 -> 5
    \or
    6% Sunday 7 -> 6
    \else
    -1% invalid
    \fi
  }
```

ayIndexFromRegion Convert from the region's day of the week (starting from 1) to pgfcalendar's 0=Monday indexing. The first argument is the dialect label.

```
  \def\LocaleDayIndexFromRegion#1#2{%
```

Get the region's first day of the week.

```
    \ifcase\LocaleFirstDayIndex{#1}
```

Monday is the first day of the week. That is, the region's using 1=Monday indexing.

```
      \LocaleDayIndexFromOneMonToZeroMon{#2}%
    \or
```

Tuesday is the first day of the week. That is, the region's using 1=Tuesday indexing. (These mid-week cases are unlikely, but add them for completeness.)

```
      \ifnum#2=7 % Monday
        0%
      \else
        #2%
      \fi
    \or
```

Wednesday is the first day of the week. That is, the region's using 1=Wednesday indexing.

```
      \ifcase#1
      % no 0
      \or
      2% Wednesday 1 -> 2
      \or
      3% Thursday 2 -> 3
      \or
      4% Friday 3 -> 4
      \or
      5% Saturday 4 -> 5
      \or
      6% Sunday 5 -> 6
      \or
      0% Monday 6 -> 0
      \or
```

```
    1% Tuesday 7 -> 1
    \else
    -1% invalid
    \fi
  \or
```

Thursday is the first day of the week. That is, the region's using 1=Thursday indexing.

```
    \ifcase#1
    % no 0
    \or
    3% Thursday 1 -> 3
    \or
    4% Friday 2 -> 4
    \or
    5% Saturday 3 -> 5
    \or
    6% Sunday 4 -> 6
    \or
    0% Monday 5 -> 0
    \or
    1% Tuesday 6 -> 1
    \or
    2% Wednesday 7 -> 2
    \else
    -1% invalid
    \fi
  \or
```

Friday is the first day of the week. That is, the region's using 1=Friday indexing.

```
    \ifcase#1
    % no 0
    \or
    4% Friday 1 -> 4
    \or
    5% Saturday 2 -> 5
    \or
    6% Sunday 3 -> 6
    \or
    0% Monday 4 -> 0
    \or
    1% Tuesday 5 -> 1
    \or
    2% Wednesday 6 -> 2
    \or
    3% Thursday 7 -> 3
```

```
      \else
      -1% invalid
      \fi
    \or
```

Saturday is the first day of the week. That is, the region's using 1=Saturday indexing.

```
      \ifcase#1
      % no 0
      \or
      5% Saturday 1 -> 5
      \or
      6% Sunday 2 -> 6
      \or
      0% Monday 3 -> 0
      \or
      1% Tuesday 4 -> 1
      \or
      2% Wednesday 5 -> 2
      \or
      3% Thursday 6 -> 3
      \or
      4% Friday 7 -> 4
      \else
      -1% invalid
      \fi
    \or
```

Sunday is the first day of the week. That is, the region's using 1=Sunday indexing.

```
      \LocaleDayIndexFromOneSunToZeroMon{#2}%
    \else
      #2%
    \fi
  }
```

eDayIndexToRegion Convert from pgfcalendar's 0=Monday indexing to the region's first day of the week (starting from 1). The first argument is the dialect label.

```
  \def\LocaleDayIndexToRegion#1#2{%
```

Get the region's first day of the week.

```
    \ifcase\LocaleFirstDayIndex{#1}
```

Monday is the first day of the week. That is, the region's using 1=Monday indexing.

```
      \LocaleDayIndexFromZeroMonToOneMon{#2}%
    \or
```

Tuesday is the first day of the week. That is, the region's using 1=Tuesday indexing. (These mid-week cases are unlikely, but add them for completeness.)

```
      \ifnum#2=0 % Monday
        7%
      \else
        #2%
      \fi
    \or
```

Wednesday is the first day of the week. That is, the region's using 1=Wednesday indexing.

```
      \ifcase#1
       6% Monday 0 -> 6
      \or
       7% Tuesday 1 -> 7
      \or
       1% Wednesday 2 -> 1
      \or
       2% Thursday 3 -> 2
      \or
       3% Friday 4 -> 3
      \or
       4% Saturday 5 -> 4
      \or
       5% Sunday 6 -> 5
      \else
      -1% invalid
      \fi
    \or
```

Thursday is the first day of the week. That is, the region's using 1=Thursday indexing.

```
      \ifcase#1
       5% Monday 0 -> 5
      \or
       6% Tuesday 1 -> 6
      \or
       7% Wednesday 2 -> 7
      \or
       1% Thursday 3 -> 1
      \or
       2% Friday 4 -> 2
      \or
       3% Saturday 5 -> 3
      \or
       4% Sunday 6 -> 4
      \else
      -1% invalid
      \fi
```

```
  \or
```
Friday is the first day of the week. That is, the region's using 1=Friday indexing.
```
    \ifcase#1
     4% Monday 0 -> 4
    \or
     5% Tuesday 1 -> 5
    \or
     6% Wednesday 2 -> 6
    \or
     7% Thursday 3 -> 7
    \or
     1% Friday 4 -> 1
    \or
     2% Saturday 5 -> 2
    \or
     3% Sunday 6 -> 3
    \else
    -1% invalid
    \fi
  \or
```
Saturday is the first day of the week. That is, the region's using 1=Saturday indexing.
```
    \ifcase#1
     3% Monday 0 -> 3
    \or
     4% Tuesday 1 -> 4
    \or
     5% Wednesday 2 -> 5
    \or
     6% Thursday 3 -> 6
    \or
     7% Friday 4 -> 7
    \or
     1% Saturday 5 -> 1
    \or
     2% Sunday 6 -> 2
    \else
    -1% invalid
    \fi
  \or
```
Sunday is the first day of the week. That is, the region's using 1=Sunday indexing.
```
    \LocaleDayIndexFromZeroMonToOneSun{#2}%
  \else
```

```
    #2%
  \fi
}
```

\LocaleFullTime `\LocaleFullTime{⟨dialect⟩}`

Get the full time for the given dialect.

```
\def\LocaleFullTime#1{%
 \localedatetimefmt{\LocaleGetDialectAttribute{#1}{fulltime}}%
}
```

\LocaleLongTime `\LocaleLongTime{⟨dialect⟩}`

Get the long time for the given dialect.

```
\def\LocaleLongTime#1{%
 \localedatetimefmt{\LocaleGetDialectAttribute{#1}{longtime}}%
}
```

\LocaleMediumTime `\LocaleMediumTime{⟨dialect⟩}`

Get the medium time for the given dialect.

```
\def\LocaleMediumTime#1{%
 \localedatetimefmt{\LocaleGetDialectAttribute{#1}{medtime}}%
}
```

\LocaleShortTime `\LocaleShortTime{⟨dialect⟩}`

Get the short time for the given dialect.

```
\def\LocaleShortTime#1{%
 \localedatetimefmt{\LocaleGetDialectAttribute{#1}{shorttime}}%
}
```

LocaleFullDateTime `\LocaleFullDateTime{⟨dialect⟩}`

Get the full date and time for the given dialect.

```
\def\LocaleFullDateTime#1{%
 \localedatetimefmt{\LocaleGetDialectAttribute{#1}{fulldatetime}}%
}
```

LocaleLongDateTime `\LocaleLongDateTime{⟨dialect⟩}`

Get the long date and time for the given dialect.

```
\def\LocaleLongDateTime#1{%
 \localedatetimefmt{\LocaleGetDialectAttribute{#1}{longdatetime}}%
}
```

caleMediumDateTime `\LocaleMediumDateTime{⟨dialect⟩}`

Get the medium date and time for the given dialect.

```
\def\LocaleMediumDateTime#1{%
 \localedatetimefmt{\LocaleGetDialectAttribute{#1}{meddatetime}}%
}
```

ocaleShortDateTime `\LocaleShortDateTime{⟨dialect⟩}`

Get the short date and time for the given dialect.

```
\def\LocaleShortDateTime#1{%
 \localedatetimefmt{\LocaleGetDialectAttribute{#1}{shortdatetime}}%
}
```

\LocaleDayName `\LocaleDayName{⟨dialect⟩}{⟨n⟩}`

Gets the day of week name identified by the index ⟨*n*⟩ (0 for Monday, 1 for Tuesday, etc) in the given dialect.

```
\def\LocaleDayName#1#2{%
 \LocaleGetDialectAttribute{#1}{day.\number#2}%
}
```

LocaleShortDayName

\LocaleShortDayName{⟨*dialect*⟩}{⟨*n*⟩}

Gets the abbreviated day of week name identified by the index ⟨*n*⟩ (0 for Monday, 1 for Tuesday, etc) in the given dialect.

```
\def\LocaleShortDayName#1#2{%
 \LocaleGetDialectAttribute{#1}{shortday.\number#2}%
}
```

\LocaleMonthName

\LocaleMonthName{⟨*dialect*⟩}{⟨*n*⟩}

Gets the month name identified by the index ⟨*n*⟩ (1 for January, 2 for February, etc) in the given dialect.

```
\def\LocaleMonthName#1#2{%
 \LocaleGetDialectAttribute{#1}{month.\number#2}%
}
```

caleShortMonthName

\LocaleShortMonthName{⟨*dialect*⟩}{⟨*n*⟩}

Gets the month week name identified by the index ⟨*n*⟩ (1 for January, 2 for February, etc) in the given dialect.

```
\def\LocaleShortMonthName#1#2{%
 \LocaleGetDialectAttribute{#1}{shortmonth.\number#2}%
}
```

Some languages have a different form for month or week day names used in a standalone context, such as in a column header. These are provided with the `\LocaleStandalone...` commands defined below. For most languages they will be the same as the above.

eStandaloneDayName

\LocaleStandaloneDayName{⟨*dialect*⟩}{⟨*n*⟩}

Gets the day of week name identified by the index ⟨*n*⟩ (0 for Monday, 1 for Tuesday, etc) in the given dialect.

```
\def\LocaleStandaloneDayName#1#2{%
 \LocaleGetDialectAttribute{#1}{standalone.day.\number#2}%
}
```

daloneShortDayName

\LocaleStandaloneShortDayName{⟨*dialect*⟩}{⟨*n*⟩}

Gets the abbreviated day of week name identified by the index ⟨*n*⟩ (0 for Monday, 1 for Tuesday, etc) in the given dialect.

```
\def\LocaleStandaloneShortDayName#1#2{%
 \LocaleGetDialectAttribute{#1}{standalone.shortday.\number#2}%
}
```

tandaloneMonthName

\LocaleStandaloneMonthName{⟨*dialect*⟩}{⟨*n*⟩}

Gets the month week name identified by the index ⟨*n*⟩ (1 for January, 2 for February, etc) in the given dialect.

```
\def\LocaleStandaloneMonthName#1#2{%
 \LocaleGetDialectAttribute{#1}{standalone.month.\number#2}%
}
```

loneShortMonthName

\LocaleStandaloneShortMonthName{⟨*dialect*⟩}{⟨*n*⟩}

Gets the month week name identified by the index ⟨*n*⟩ (1 for January, 2 for February, etc) in the given dialect.

```
\def\LocaleStandaloneShortMonthName#1#2{%
 \LocaleGetDialectAttribute{#1}{standalone.shortmonth.\number#2}%
}
```

aleNumericGroupSep

\LocaleNumericGroupSep{⟨*dialect*⟩}

Gets the numeric group separator for the given dialect.

```
\def\LocaleNumericGroupSep#1{%
 \LocaleGetDialectAttribute{#1}{groupsep}%
}
```

eNumericDecimalSep

\LocaleNumericDecimalSep{⟨*dialect*⟩}

Gets the numeric decimal separator for the given dialect.

```
\def\LocaleNumericDecimalSep#1{%
 \LocaleGetDialectAttribute{#1}{decsep}%
}
```

NumericMonetarySep `\LocaleNumericMonetarySep{⟨dialect⟩}`

Gets the numeric monetary separator for the given dialect.

```
\def\LocaleNumericMonetarySep#1{%
 \LocaleGetDialectAttribute{#1}{currencysep}%
}
```

aleNumericExponent `\LocaleNumericExponent{⟨dialect⟩}`

Gets the exponent symbol for the given dialect.

```
\def\LocaleNumericExponent#1{%
 \LocaleGetDialectAttribute{#1}{exp}%
}
```

ocaleCurrencyLabel `\LocaleCurrencyLabel{⟨dialect⟩}`

Gets the currency label for the given dialect.

```
\def\LocaleCurrencyLabel#1{%
 \LocaleIfDialectAttributeEqCs{#1}{currency}{\@locale@unknown@currency}%
 {%
```

Unknown currency symbol.

```
    \ifx\LocaleMainRegion\empty
      \@locale@os@currencycode
    \else
      \LocaleGetDialectAttributeOrDefValue{\LocaleMainDialect}{currency}%
      {\@locale@os@currencycode}%
    \fi
  }%
  {\LocaleGetDialectAttribute{#1}{currency}}%
}
```

rencyRegionalLabel `\LocaleCurrencyRegionalLabel{⟨dialect⟩}`

Gets the regional currency label for the given dialect.

```
\def\LocaleCurrencyRegionalLabel#1{%
 \LocaleIfDialectAttributeEqCs{#1}{regionalcurrency}%
 {\@locale@unknown@currency}%
 {%
```

Unknown currency symbol.

```
    \ifx\LocaleMainRegion\empty
      \@locale@os@regionalcurrencycode
    \else
      \LocaleGetDialectAttributeOrDefValue{\LocaleMainDialect}{regionalcurrency}%
      {\@locale@os@regionalcurrencycode}%
    \fi
 }%
 {\LocaleGetDialectAttribute{#1}{regionalcurrency}}%
}
```

caleCurrencySymbol

`\LocaleCurrencySymbol{⟨dialect⟩}`

Gets the currency symbol for the given dialect. (May have non-ASCII characters.)

```
\def\LocaleCurrencySymbol#1{%
 \LocaleIfDialectAttributeEqCs{#1}{currency}{\@locale@unknown@currency}%
 {%
```

Unknown currency code. Does the main locale have a region?

```
    \ifx\LocaleMainRegion\empty
```

No main region, try the default OS currency.

```
      \ifx\@locale@os@currencysym\empty
        \LocaleGetDialectAttribute{#1}{currencysym}%
      \else
        \@locale@os@currencysym
      \fi
    \else
```

Try the main dialect attribute.

```
      \LocaleGetDialectAttributeOrDefValue{\LocaleMainDialect}{currencysym}%
      {\LocaleGetDialectAttribute{#1}{currencysym}}%
    \fi
 }%
 {%
    \LocaleIfSameDialectAttributeValues{#1}%
    {currency}{currencysym}%
```

```
    {%
      \LocaleGetCurrencyAttributeOrDefValue
      {\LocaleGetDialectAttribute{#1}{currency}}{sym}%
      {\LocaleGetDialectAttribute{#1}{currencysym}}%
    }%
    {%
      \LocaleGetDialectAttribute{#1}{currencysym}%
    }%
  }%
 }
```

eCurrencyTeXSymbol

\LocaleCurrencyTeXSymbol{⟨*dialect*⟩}

Gets the currency symbol for the given dialect. (May include texosquery currency control symbols.)

```
\def\LocaleCurrencyTeXSymbol#1{%
 \LocaleIfDialectAttributeEqCs{#1}{currency}{\@locale@unknown@currency}%
 {%
```

Unknown currency code. Does the main locale have a region?

```
    \ifx\LocaleMainRegion\empty
      \ifx\@locale@os@currencytex\empty
        \LocaleGetDialectAttribute{#1}{currencytex}%
      \else
        \@locale@os@currencytex
      \fi
    \fi
  }%
  {%
    \LocaleIfSameDialectAttributeValues{#1}%
    {currency}{currencytex}%
    {%
      \LocaleGetCurrencyAttributeOrDefValue
      {\LocaleGetDialectAttribute{#1}{currency}}{tex}%
      {\LocaleGetDialectAttribute{#1}{currencytex}}%
    }%
    {%
      \LocaleGetDialectAttribute{#1}{currencytex}%
    }%
  }%
 }
```

caleNumericPercent

\LocaleNumericPercent{⟨*dialect*⟩}

Gets the percent symbol for this dialect.

```
\def\LocaleNumericPercent#1{%
 \LocaleGetDialectAttribute{#1}{percent}%
}
```

caleNumericPermill

\LocaleNumericPermill{⟨*dialect*⟩}

Gets the permill symbol for this dialect.

```
\def\LocaleNumericPermill#1{%
 \LocaleGetDialectAttribute{#1}{permill}%
}
```

4.2.6 Patterns

plyDateTimePattern

\LocaleApplyDateTimePattern{⟨*dialect*⟩}{⟨*attribute*⟩}{⟨*date-time data*⟩}

```
\def\LocaleApplyDateTimePattern#1#2#3{%
 \LocaleIfHasDialectNonEmptyAttribute{#1}{#2}%
 {%
```

The data might be provided with the control sequence \LocaleDateTimeInfo, which may be empty and will need expanding.

```
    \ifx\empty#3\empty
      \@locale@warn{No date-time data for pattern
        (attribute ‘#2’, dialect ‘#1’)}%
    \else
     \localedatetimefmt
      {\expandafter\texosqueryfmtdatetime
       \csname locale@dialect@#2@#1\expandafter\endcsname#3}%
    \fi
 }%
 {%
   \@locale@warn{No date-time pattern attribute ‘#2’ for dialect ‘#1’}%
 }%
}
```

If the date-time patterns are required, the name commands such as \texosqueryfmtpatMMM need defining

```
\LocaleIfDateTimePatternsSupported
{%
```

xosqueryfmtpatMMM Short month name.

```
\def\texosqueryfmtpatMMM#1{%
  \CurrentLocaleShortMonthName{#1}%
}
```

osqueryfmtpatMMMM Full month name.

```
\def\texosqueryfmtpatMMMM#1{%
  \CurrentLocaleMonthName{#1}%
}
```

xosqueryfmtpatLLL Short standalone month name.

```
\def\texosqueryfmtpatLLL#1{%
  \CurrentLocaleStandaloneShortMonthName{#1}%
}
```

osqueryfmtpatLLLL Full standalone month name.

```
\def\texosqueryfmtpatLLLL#1{%
  \CurrentLocaleStandaloneMonthName{#1}%
}
```

xosqueryfmtpatEEE Short day of week name.

```
\def\texosqueryfmtpatEEE#1{%
  \CurrentLocaleShortDayName{\LocaleDayIndexFromOneMonToZeroMon{#1}}%
}
```

osqueryfmtpatEEEE Full day of week name.

```
\def\texosqueryfmtpatEEEE#1{%
  \CurrentLocaleDayName{\LocaleDayIndexFromOneMonToZeroMon{#1}}%
}
```

squerytimezonefmt Allow the time zone name to be wrapped in a formatting command.

```
\def\texosquerytimezonefmt#1{#1}
```

ueryshorttimezone

```
\def\texosqueryshorttimezone#1{%
  \texosquerytimezonefmt
   {\LocaleGetDialectAttribute{\CurrentTrackedDialect}{timezone.#1.short}}%
}
```

queryshortdstzone

```
\def\texosqueryshortdstzone#1{%
  \texosquerytimezonefmt
   {\LocaleGetDialectAttribute{\CurrentTrackedDialect}{timezone.#1.shortdst}}%
}
```

querylongtimezone

```
\def\texosquerylongtimezone#1{%
  \texosquerytimezonefmt
   {\LocaleGetDialectAttribute{\CurrentTrackedDialect}{timezone.#1.long}}%
}
```

squerylongdstzone

```
\def\texosquerylongdstzone#1{%
  \texosquerytimezonefmt
   {\LocaleGetDialectAttribute{\CurrentTrackedDialect}{timezone.#1.longdst}}%
}
```

End of first argument of \LocaleIfDateTimePatternsSupported. (Do nothing if date-time patterns not required.)

```
}
{}
```

Set up the texosquery numbering commands to use the current locale symbols.

erypatfmtgroupsep

```
\def\texosquerypatfmtgroupsep{\CurrentLocaleNumericGroupSep}
```

querypatfmtdecsep

```
\def\texosquerypatfmtdecsep{\CurrentLocaleDecimalSep}
```

rypatfmtcurdecsep

```
\def\texosquerypatfmtcurdecsep{\CurrentLocaleMonetarySep}
```

xosquerypatfmtexp

```
\def\texosquerypatfmtexp{\CurrentLocaleExponent}
```

patfmtcurrencysign

```
\def\texosquerypatfmtcurrencysign{\CurrentLocaleCurrency}
```

patfmtpercentsign

```
\def\texosquerypatfmtpercentsign{\CurrentLocalePercent}
```

patfmtpermillsign

```
\def\texosquerypatfmtpermillsign{\CurrentLocalePermill}
```

4.2.7 Conditionals

IfNumericUsesGroup

```
\LocaleIfNumericUsesGroup{⟨dialect⟩}{⟨true⟩}{⟨false⟩}
```

Determines if the locale uses numerical group separator. The usesgroup attribute will be 1 for true and 0 for false, so we need a numerical comparison.

```
\def\LocaleIfNumericUsesGroup#1{%
 \LocaleIfDialectAttributeEqNum{#1}{usesgroup}{1}%
}
```

eIfHasLanguageName

```
\LocaleIfHasLanguageName{⟨dialect⟩}{⟨true⟩}{⟨false⟩}
```

Determines if the given dialect has a language name.

```
\def\LocaleIfHasLanguageName#1{%
 \LocaleIfHasDialectNonEmptyAttribute{#1}{langname}%
}
```

No need for a separate check for the native version. If it has one, then it will have the other.

aleIfHasRegionName

```
\LocaleIfHasRegionName{⟨dialect⟩}{⟨true⟩}{⟨false⟩}
```

Determines if the given dialect has a region name.

```
\def\LocaleIfHasRegionName#1{%
 \LocaleIfHasDialectNonEmptyAttribute{#1}{regionname}%
}
```

leIfHasVariantName

```
\LocaleIfHasVariantName{⟨dialect⟩}{⟨true⟩}{⟨false⟩}
```

Determines if the given dialect has a variant name.

```
\def\LocaleIfHasVariantName#1{%
 \LocaleIfHasDialectNonEmptyAttribute{#1}{variantname}%
}
```

4.2.8 Post-Parsing

If this hook has been defined, use it.

```
\ifx\@locale@postparse@hook\undefined
\else
  \@locale@postparse@hook
\fi
```

4.2.9 Initialising Current Locale Commands

These commands are all initialised to do nothing (except for indexes which are initialised to `-1`), in the event that the dry run mode is on.

```
\def\CurrentLocaleLanguageName{}%
\def\CurrentLocaleLanguageNativeName{}%
\def\CurrentLocaleRegionName{}%
\def\CurrentLocaleRegionNativeName{}%
\def\CurrentLocaleVariantName{}%
\def\CurrentLocaleVariantNativeName{}%
\def\CurrentLocaleFirstDayIndex{-1}%
\def\CurrentLocaleDayIndexFromRegion{-1}%
\def\CurrentLocaleDayName{}%
\def\CurrentLocaleShortDayName{}%
\def\CurrentLocaleStandaloneDayName{}%
\def\CurrentLocaleStandaloneShortDayName{}%
\def\CurrentLocaleMonthName{}%
\def\CurrentLocaleShortMonthName{}%
\def\CurrentLocaleStandaloneMonthName{}%
\def\CurrentLocaleStandaloneShortMonthName{}%
\def\CurrentLocaleFullDate{}%
\def\CurrentLocaleLongDate{}%
\def\CurrentLocaleMediumDate{}%
\def\CurrentLocaleShortDate{}%
\def\CurrentLocaleFullTime{}%
\def\CurrentLocaleLongTime{}%
\def\CurrentLocaleMediumTime{}%
\def\CurrentLocaleShortTime{}%
\def\CurrentLocaleFullDateTime{}%
\def\CurrentLocaleLongDateTime{}%
\def\CurrentLocaleMediumDateTime{}%
\def\CurrentLocaleShortDateTime{}%
\def\CurrentLocaleDate{}%
\def\CurrentLocaleTime{}%
\def\CurrentLocaleCurrency{}%
\def\CurrentLocaleNumericGroupSep{}%
```

```
\def\CurrentLocaleIfNumericUsesGroup{}%
\def\CurrentLocaleDecimalSep{}%
\def\CurrentLocaleMonetarySep{}%
\def\CurrentLocaleExponent{}%
\def\CurrentLocalePercent{}%
\def\CurrentLocalePermill{}%
\def\CurrentLocaleIntegerPattern{}%
\def\CurrentLocaleDecimalPattern{}%
\def\CurrentLocaleCurrencyPattern{}%
\def\CurrentLocalePercentPattern{}%
\def\CurrentLocaleApplyDateTimePattern{}%
\def\CurrentTrackedDialect{}%
```

4.2.10 Switching Locale

`\@locale@select` Used when a language is changed. The argument is the dialect label.

```
\def\@locale@select#1{%
  \SetCurrentTrackedDialect{#1}%
  \def\CurrentLocaleLanguageName{\LocaleLanguageName{#1}}%
  \def\CurrentLocaleLanguageNativeName{\LocaleLanguageNativeName{#1}}%
  \def\CurrentLocaleRegionName{\LocaleRegionName{#1}}%
  \def\CurrentLocaleRegionNativeName{\LocaleRegionNativeName{#1}}%
  \def\CurrentLocaleVariantName{\LocaleVariantName{#1}}%
  \def\CurrentLocaleVariantNativeName{\LocaleVariantNativeName{#1}}%
  \def\CurrentLocaleFirstDayIndex{\LocaleFirstDayIndex{#1}}%
  \def\CurrentLocaleDayIndexFromRegion{\LocaleDayIndexFromRegion{#1}}%
  \def\CurrentLocaleDayName{\LocaleDayName{#1}}%
  \def\CurrentLocaleShortDayName{\LocaleShortDayName{#1}}%
  \def\CurrentLocaleStandaloneDayName{%
    \LocaleStandaloneDayName{#1}}%
  \def\CurrentLocaleStandaloneShortDayName{%
    \LocaleStandaloneShortDayName{#1}}%
  \def\CurrentLocaleMonthName{\LocaleMonthName{#1}}%
  \def\CurrentLocaleShortMonthName{\LocaleShortMonthName{#1}}%
  \def\CurrentLocaleStandaloneMonthName{%
    \LocaleStandaloneMonthName{#1}}%
  \def\CurrentLocaleStandaloneShortMonthName{%
    \LocaleStandaloneShortMonthName{#1}}%
  \def\CurrentLocaleFullDate{\LocaleFullDate{#1}}%
  \def\CurrentLocaleLongDate{\LocaleLongDate{#1}}%
  \def\CurrentLocaleMediumDate{\LocaleMediumDate{#1}}%
  \def\CurrentLocaleShortDate{\LocaleShortDate{#1}}%
  \def\CurrentLocaleFullTime{\LocaleFullTime{#1}}%
```

```
\def\CurrentLocaleLongTime{\LocaleLongTime{#1}}%
\def\CurrentLocaleMediumTime{\LocaleMediumTime{#1}}%
\def\CurrentLocaleShortTime{\LocaleShortTime{#1}}%
\def\CurrentLocaleFullDateTime{\LocaleFullDateTime{#1}}%
\def\CurrentLocaleLongDateTime{\LocaleLongDateTime{#1}}%
\def\CurrentLocaleMediumDateTime{\LocaleMediumDateTime{#1}}%
\def\CurrentLocaleShortDateTime{\LocaleShortDateTime{#1}}%
\def\CurrentLocaleDate{%
  \localedatechoice
   {\LocaleFullDate{#1}}%
   {\LocaleLongDate{#1}}%
   {\LocaleMediumDate{#1}}%
   {\LocaleShortDate{#1}}%
}%
\def\CurrentLocaleTime{%
  \localetimechoice
   {\LocaleFullTime{#1}}%
   {\LocaleLongTime{#1}}%
   {\LocaleMediumTime{#1}}%
   {\LocaleShortTime{#1}}%
}%
\def\CurrentLocaleCurrency{%
  \localecurrchoice
   {\LocaleCurrencyLabel{#1}}%
   {\LocaleCurrencyRegionalLabel{#1}}%
   {\LocaleCurrencySymbol{#1}}%
   {\LocaleCurrencyTeXSymbol{#1}}%
}%
\def\CurrentLocaleNumericGroupSep{%
  \LocaleNumericGroupSep{#1}%
}%
\def\CurrentLocaleIfNumericUsesGroup{%
  \LocaleIfNumericUsesGroup{#1}%
}%
\def\CurrentLocaleDecimalSep{%
  \LocaleNumericDecimalSep{#1}%
}%
\def\CurrentLocaleMonetarySep{%
  \LocaleNumericMonetarySep{#1}%
}%
\def\CurrentLocaleExponent{%
  \LocaleNumericExponent{#1}%
}%
```

```
    \def\CurrentLocalePercent{%
      \LocaleNumericPercent{#1}%
    }%
    \def\CurrentLocalePermill{%
      \LocaleNumericPermill{#1}%
    }%
    \def\CurrentLocaleIntegerPattern{%
      \LocaleGetDialectAttribute{#1}{intfmt}%
    }%
    \def\CurrentLocaleDecimalPattern{%
      \LocaleGetDialectAttribute{#1}{decfmt}%
    }%
    \def\CurrentLocaleCurrencyPattern{%
      \LocaleGetDialectAttribute{#1}{curfmt}%
    }%
    \def\CurrentLocalePercentPattern{%
      \LocaleGetDialectAttribute{#1}{perfmt}%
    }%
    \def\CurrentLocaleApplyDateTimePattern{%
      \LocaleApplyDateTimePattern{#1}%
    }%
  }
```

Provide the choice commands:

\localedatechoice

```
  \ifx\localedatechoice\undefined
    \def\localedatechoice#1#2#3#4{#2}
  \fi
```

\localetimechoice

```
  \ifx\localetimechoice\undefined
    \def\localetimechoice#1#2#3#4{#3}
  \fi
```

\localecurrchoice

```
  \ifx\localecurrchoice\undefined
    \def\localecurrchoice#1#2#3#4{#2}
  \fi
```

entLocaleDateTime Shortcut for date and time.

```
  \def\CurrentLocaleDateTime{\CurrentLocaleDate\space\CurrentLocaleTime}
```

Provide convenient shortcut commands for formatting numbers. These need a bit of help to split the argument on . and on E. First provide a general purpose command.

`\localenumfmt` The first argument is the pattern. The second argument is the value.

```
\def\localenumfmt#1#2{%
  \@locale@decfmt#2\empty.0E0\relax
 \ifnum\@locale@decfmt@int<0
   \let\@localenum@fmt\localenumfmtneg
 \else
   \ifnum\@locale@decfmt@int=0
     \ifnum\@locale@decfmt@frac=0
       \ifnum\@locale@decfmt@exp=0
         \let\@localenum@fmt\localenumfmtzero
       \else
         \let\@localenum@fmt\localenumfmtpos
       \fi
     \else
       \let\@localenum@fmt\localenumfmtpos
     \fi
   \else
     \let\@localenum@fmt\localenumfmtpos
   \fi
 \fi
 \@localenum@fmt
 {%
   \texosqueryfmtnumber{#1}%
   {\@locale@decfmt@int}%
   {\@locale@decfmt@frac}%
   {\@locale@decfmt@exp}%
 }%
}
```

`\localenumfmtpos` Wrapper for positive values.

```
\def\localenumfmtpos#1{#1}
```

`\localenumfmtneg` Wrapper for negative values.

```
\def\localenumfmtneg#1{#1}
```

`\localenumfmtzero` Wrapper for zero values.

```
\def\localenumfmtzero#1{#1}
```

`\localeint`

```
\def\localeint#1{%
 \localenumfmt{\CurrentLocaleIntegerPattern}{#1}%
}
```

\localedec

```
\def\localedec#1{%
 \localenumfmt{\CurrentLocaleDecimalPattern}{#1}%
}
```

\localecur

```
\def\localecur#1{%
 \localenumfmt{\CurrentLocaleCurrencyPattern}{#1}%
}
```

\localeper

```
\def\localeper#1{%
 \localenumfmt{\CurrentLocalePercentPattern}{#1}%
}
```

\@locale@decfmt

```
\def\@locale@decfmt#1.#2E#3\relax{%
```

Allow for mantissa present but missing fractional part.

```
 \@locale@decfmt@split#1E\empty\relax
 \ifx\@locale@decfmt@exp\empty
   \edef\@locale@decfmt@frac{\@locale@gobbleemptytorelax#2\empty\relax}%
   \edef\@locale@decfmt@exp{\@locale@gobbleemptytorelax#3\empty\relax}%
 \else
   \def\@locale@decfmt@frac{0}%
 \fi
}
```

gobbleemptytorelax Strip everything between \empty and \relax

```
\def\@locale@gobbleemptytorelax#1\empty#2\relax{#1}
```

cale@decfmt@split Split on E. (In case E but no . supplied.)

```
\def\@locale@decfmt@split#1E#2\empty#3\relax{%
 \edef\@locale@decfmt@int{\@locale@gobbleemptytorelax#1\empty\relax}%
 \edef\@locale@decfmt@exp{\@locale@gobbleemptytorelax#2\empty\relax}%
}
```

Iterate through all languages and add to caption hook.

```
\ForEachTrackedDialect{\locale@this@dialect}{%
  \SetCurrentTrackedDialect{\locale@this@dialect}%
  \expandafter\@TrackLangAddToHook\expandafter
   {\expandafter\@locale@select\expandafter{\locale@this@dialect}}
   {captions}%
}
```

Provide a command to select a language using the tracklang dialect label or the language tag.

\selectlocale

```
\def\selectlocale#1{%
  \ifx\selectlanguage\undefined
    \IfTrackedDialect{#1}%
    {\@locale@select{#1}}%
    {%
      \LocaleIfHasAttribute{#1}{tagtodialect}
      {%
        \edef\@locale@dialect{\LocaleGetAttribute{#1}{tagtodialect}}%
        \expandafter\@locale@select\expandafter{\@locale@dialect}%
      }%
      {%
        \ifTeXOSQueryDryRun
          \@locale@warn{Unknown locale ‘#1’}%
        \else
          \@locale@err{Unknown locale ‘#1’}%
          {The argument to \string\selectlocale\space must be either a
           tracklang dialect label or a tracked dialect language tag}%
        \fi
      }%
    }%
  \else
```

Need to find the correct label to use in \selectlanguage

```
    \IfTrackedDialect{#1}%
    {%
      \@locale@select@dialect{#1}%
    }%
    {%
      \LocaleIfHasAttribute{#1}{tagtodialect}
      {\@locale@select@dialect{\LocaleGetAttribute{#1}{tagtodialect}}}%
      {%
        \ifTeXOSQueryDryRun
          \@locale@warn{Unknown locale ‘#1’}%
        \else
          \@locale@err{Unknown locale ‘#1’}%
          {The argument to \string\selectlocale\space must be either a
           tracklang dialect label or a tracked dialect language tag}%
        \fi
      }%
    }%
  \fi
```

```
}
```

`le@select@dialect` Select language from tracklang dialect label.

```
\def\@locale@select@dialect#1{%
  \IfTrackedDialectHasMapping{#1}%
  {\edef\@locale@dialect{\GetTrackedDialectToMapping{#1}}}%
  {\edef\@locale@dialect{#1}}%
```

Is there a date hook available? (Match the check used by `\selectlanguage`.)

```
  \@tracklang@ifundef{date\@locale@dialect}%
  {%
```

Try the root language.

```
    \edef\@locale@dialect{\TrackedLanguageFromDialect{#1}}%
```

Is there a date hook available?

```
    \@tracklang@ifundef{date\@locale@dialect}%
    {%
      \@locale@warn{Can't determine correct label for
       \string\selectlanguage\space from tracklang dialect `#1'}%
      \SetCurrentTrackedDialect{#1}%
    }%
    {\expandafter\selectlanguage\expandafter{\@locale@dialect}}%
  }%
  {\expandafter\selectlanguage\expandafter{\@locale@dialect}}%
}
```

Select the main locale (but don't try any language change as this might upset polyglossia if the fonts haven't been set yet), so use the internal `\@locale@select`.

```
\ifx\LocaleMainDialect\empty
\else
  \expandafter\@locale@select\expandafter{\LocaleMainDialect}
\fi
```

Restore category code for `@` if necessary.

```
\@locale@restore@at
```

4.3 Scripts to fontenc mappings (`tex-locale-scripts-enc.def`)

Provides mappings from scripts to font encodings. Not all supported. This file is only loaded if the fontenc option is used. The `@` character is assumed to have category code when this file is input.

`ale@scriptenc@map` Maps ISO 15924 script label to LaTeX font encoding name.

```
\def\@locale@scriptenc@map#1#2{%
  \@tracklang@namedef{@locale@scriptenc@map@#1}{#2}%
}
```

`@get@scriptenc@map` Get the mapping.

```
\def\@locale@get@scriptenc@map#1{%
  \@tracklang@ifundef{@locale@scriptenc@map@#1}%
  {}%
  {\csname @locale@scriptenc@map@#1\endcsname}%
}
```

`@if@scriptenc@map` Test if there's a script mapping.

```
\def\@locale@if@scriptenc@map#1#2#3{%
  \@tracklang@ifundef{@locale@scriptenc@map@#1}%
  {#3}% false
  {#2}% true
}
```

`locale@langenc@map` Maps tracklang language or dialect labels to LaTeX font encoding name.

```
\def\@locale@langenc@map#1#2{%
  \@tracklang@namedef{@locale@langenc@map@#1}{#2}%
}
```

`le@if@langenc@map` Test if there's a lang mapping.

```
\def\@locale@if@langenc@map#1#2#3{%
  \@tracklang@ifundef{@locale@langenc@map@#1}%
  {#3}% false
  {#2}% true
}
```

`e@get@langenc@map` Get the mapping.

```
\def\@locale@get@langenc@map#1{%
  \@tracklang@ifundef{@locale@langenc@map@#1}%
  {}%
  {\csname @locale@langenc@map@#1\endcsname}%
}
```

Define mappings. Only a few currently supported. It's better to use fontspec instead of fontenc for non-Latin scripts.

Scripts first.

```
\@locale@scriptenc@map{Latn}{T1}
\@locale@scriptenc@map{Latf}{T1}
\@locale@scriptenc@map{Latg}{T1}
\@locale@scriptenc@map{Cyrl}{T2A,T2B,T2C}
```

Now languages or dialect labels:

```
\@locale@langenc@map{vietnamese}{T5}
\@locale@langenc@map{polish}{OT4}
```

```
\@locale@langenc@map{armenian}{OT6}
\@locale@langenc@map{greek}{LGR}
```

4.4 texosquery to inputenc mappings (tex-locale-encodings.def)

Encoding mappings.

@locale@newencmap

```
\def\@locale@newencmap#1#2{%
  \@tracklang@namedef{@locale@encmap@#1}{#2}}
```

ocale@ifhasencmap

```
\def\@locale@ifhasencmap#1#2#3{%
  \@tracklang@ifundef{@locale@encmap@#1}{#3}{#2}}
```

@locale@getencmap

```
\def\@locale@getencmap#1{%
  \@tracklang@nameuse{@locale@encmap@#1}}

\@locale@newencmap{iso88591}{latin1}
\@locale@newencmap{iso88592}{latin2}
\@locale@newencmap{iso88593}{latin3}
\@locale@newencmap{iso88594}{latin4}
\@locale@newencmap{iso88595}{latin5}
\@locale@newencmap{iso88599}{latin9}
\@locale@newencmap{windows1250}{cp1250}
\@locale@newencmap{windows1252}{cp1252}
\@locale@newencmap{windows1257}{cp1257}
\@locale@newencmap{ibm850}{cp850}
\@locale@newencmap{ibm852}{cp852}
\@locale@newencmap{ibm437}{cp437}
\@locale@newencmap{ibm865}{cp865}
\@locale@newencmap{usascii}{ascii}
\@locale@newencmap{xmaccentraleurope}{macce}
```

4.5 Language Support Setup

The language support setup performed in \@locale@postparse@hook. Load babel or polyglossia.

```
\@locale@ifsupportbabelorpoly
{}
{%
```

(babel support.) Set up some extra mappings that aren't in tracklang version 1.3. There's no intuitive way of testing if a dialect label is provided by babel. There's only a test for the root language by checking for the existence of the ldf file, so these are known babel options. This means that if there isn't a mapping, the root language will be used instead.

```
\def\@locale@providemap#1#2{%
  \IfTrackedDialectHasMapping{#1}%
  {}%
  {\SetTrackedDialectLabelMap{#1}{#2}}%
}%
\@locale@providemap{bahasa}{bahasa}%
\@locale@providemap{indonesian}{indonesian}%
\@locale@providemap{indon}{indon}%
\@locale@providemap{bahasam}{bahasam}%
\@locale@providemap{malay}{malay}%
\@locale@providemap{melayu}{melayu}%
\@locale@providemap{USenglish}{USenglish}%
\@locale@providemap{american}{american}%
\@locale@providemap{UKenglish}{UKenglish}%
\@locale@providemap{british}{british}%
\@locale@providemap{canadian}{canadian}%
\@locale@providemap{australian}{australian}%
\@locale@providemap{newzealand}{newzealand}%
\@locale@providemap{francais}{francais}%
\@locale@providemap{canadien}{canadien}%
\@locale@providemap{acadian}{acadian}%
\@locale@providemap{austrian}{austrian}%
\@locale@providemap{germanb}{germanb}%
\@locale@providemap{ngerman}{ngerman}%
\@locale@providemap{naustrian}{naustrian}%
\@locale@providemap{nswissgerman}{nswissgerman}%
\@locale@providemap{swissgerman}{swissgerman}%
\@locale@providemap{polutonikogreek}{greek}%
\@locale@providemap{nynorsk}{nynorsk}%
\@locale@providemap{portuguese}{portuguese}%
\@locale@providemap{brazilian}{brazilian}%
\@locale@providemap{brazil}{brazil}%
```

Work out how to pass the options since we may need to convert a tracklang dialect label to a recognised babel label.

```
\def\@locale@bbl@options{}%
\ForEachTrackedDialect{\this@dialect}%
{%
  \edef\this@root@lang{%
    \TrackedLanguageFromDialect{\this@dialect}}%
```

Is there a language file?

```
\IfFileExists{\this@root@lang.ldf}
{%
```

Check for Serbian, since we need to know the script.

```
  \ifdefstring{\this@root@lang}{serbian}%
  {%
    \@locale@loadscripts
    \IfTrackedDialectIsScriptCs{\this@dialect}%
    {\TrackLangScriptLatn}%
    {\def\locale@bbl@dialect{serbian}}%
    {\def\locale@bbl@dialect{serbianc}}%
  }%
  {%
    \IfFileExists{\this@dialect.ldf}%
    {%
      \let\locale@bbl@dialect\this@dialect
    }%
    {%
```

Do we have a mapping for this dialect label?

```
      \IfTrackedDialectHasMapping{\this@dialect}%
      {%
        \edef\locale@bbl@dialect{%
          \GetTrackedDialectToMapping{\this@dialect}}%
      }%
      {%
        \let\locale@bbl@dialect\this@root@lang
      }%
    }%
  }%
  \SetTrackedDialectLabelMap{\this@dialect}{\locale@bbl@dialect}%
  \ifx\this@dialect\LocaleMainDialect
    \ifx\@locale@bbl@options\empty
      \edef\@locale@bbl@options{main=\locale@bbl@dialect}%
    \else
      \edef\@locale@bbl@options{\@locale@bbl@options,%
        main=\locale@bbl@dialect}%
    \fi
  \else
    \ifx\@locale@bbl@options\empty
      \edef\@locale@bbl@options{\locale@bbl@dialect}%
    \else
      \edef\@locale@bbl@options{\@locale@bbl@options,%
```

```
                \locale@bbl@dialect}%
            \fi
          \fi
        }%
        {}%
      }%
      \ifx\@locale@bbl@options\@empty
        \ifTeXOSQueryDryRun
          \@locale@err{Can't determine 'babel' package
           options\MessageBreak (texosquery's dry run mode is on)}{}
        \else
          \@locale@err{Can't determine 'babel' package
           options (perhaps the shell escape failed, check
           '\jobname.log')}{}
        \fi
      \else
         \expandafter\PassOptionsToPackage\expandafter
           {\@locale@bbl@options}{babel}
         \RequirePackage{babel}%
      \fi
  }%
  {%
```

(polyglossia support).

Need a way of mapping scripts, regions and variants to known polyglossia keys. A lot of these variants aren't official BCP 47 tags.

```
\def\@set@locale@poly@map@script#1#2#3{%
  \@tracklang@namedef{@local@poly@map@script@#1@#2}{#3}%
}
\def\@if@locale@poly@map@script#1#2#3#4{%
  \@tracklang@ifundef{@local@poly@map@script@#1@#2}{#4}{#3}%
}
\def\@get@locale@poly@map@script#1#2{%
  \@tracklang@nameuse{@local@poly@map@script@#1@#2}%
}
\def\@set@locale@poly@map@region#1#2#3{%
  \@tracklang@namedef{@local@poly@map@region@#1@#2}{#3}%
}
\def\@if@locale@poly@map@region#1#2#3#4{%
  \@tracklang@ifundef{@local@poly@map@region@#1@#2}{#4}{#3}%
}
\def\@get@locale@poly@map@region#1#2{%
  \@tracklang@nameuse{@local@poly@map@region@#1@#2}%
}
```

```
\def\@set@locale@poly@map@variant#1#2#3{%
  \@tracklang@namedef{@local@poly@map@variant@#1@#2}{#3}%
}
\def\@if@locale@poly@map@variant#1#2#3#4{%
  \@tracklang@ifundef{@local@poly@map@variant@#1@#2}{#4}{#3}%
}
\def\@get@locale@poly@map@variant#1#2{%
  \@tracklang@nameuse{@local@poly@map@variant@#1@#2}%
}
\def\@set@locale@poly@map@sublang#1#2#3{%
  \@tracklang@namedef{@local@poly@map@sublang@#1@#2}{#3}%
}
\def\@if@locale@poly@map@sublang#1#2#3#4{%
  \@tracklang@ifundef{@local@poly@map@sublang@#1@#2}{#4}{#3}%
}
\def\@get@locale@poly@map@sublang#1#2{%
  \@tracklang@nameuse{@local@poly@map@sublang@#1@#2}%
}
\@set@locale@poly@map@region{arabic}{IQ}{locale=mashriq}
\@set@locale@poly@map@region{arabic}{SY}{locale=mashriq}
\@set@locale@poly@map@region{arabic}{JO}{locale=mashriq}
\@set@locale@poly@map@region{arabic}{LB}{locale=mashriq}
\@set@locale@poly@map@region{arabic}{PS}{locale=mashriq}
\@set@locale@poly@map@region{arabic}{LY}{locale=libya}
\@set@locale@poly@map@region{arabic}{DZ}{locale=algeria}
\@set@locale@poly@map@region{arabic}{TN}{locale=tunisia}
\@set@locale@poly@map@region{arabic}{MA}{locale=morocco}
\@set@locale@poly@map@region{arabic}{MR}{locale=mauritania}
\@set@locale@poly@map@variant{arabic}{islamic}{calendar=islamic}
\@set@locale@poly@map@variant{arabic}{maghrib}{numerals=maghrib}
\@set@locale@poly@map@variant{arabic}{abjad}{abjadjimnotail}
\@set@locale@poly@map@variant{bengali}{western}{numerals=Western}
\@set@locale@poly@map@variant{bengali}{devanagari}{numerals=Devanagari}
\@set@locale@poly@map@variant{bengali}{bengali}{numerals=Bengali}
\@set@locale@poly@map@region{english}{US}{variant=us}
\@set@locale@poly@map@region{english}{GB}{variant=uk}
\@set@locale@poly@map@region{english}{AU}{variant=australian}
\@set@locale@poly@map@region{english}{NZ}{variant=newzealand}
\@set@locale@poly@map@variant{farsi}{western}{numerals=western}
\@set@locale@poly@map@variant{farsi}{eastern}{numerals=eastern}
\@set@locale@poly@map@region{german}{DE}{variant=german}
\@set@locale@poly@map@region{german}{AU}{variant=austrian}
\@set@locale@poly@map@region{german}{CH}{variant=swiss}
```

```
\@set@locale@poly@map@variant{german}{1996}{spelling=new}
\@set@locale@poly@map@variant{german}{1901}{spelling=old}
\@set@locale@poly@map@script{german}{Latf}{script=fraktur}
\@set@locale@poly@map@variant{greek}{monoton}{variant=monotonic}
\@set@locale@poly@map@variant{greek}{polyton}{variant=polytonic}
\@set@locale@poly@map@variant{greek}{ancient}{variant=ancient}
\@set@locale@poly@map@variant{greek}{arabic}{numerals=arabic}
\@set@locale@poly@map@variant{hebrew}{arabic}{numerals=arabic}
\@set@locale@poly@map@variant{hebrew}{gregorian}{calendar=gregorian}
\@set@locale@poly@map@variant{hindi}{western}{numerals=Western}
\@set@locale@poly@map@variant{hindi}{devanagari}{numerals=Devanagari}
\@set@locale@poly@map@variant{latin}{classic}{variant=classic}
\@set@locale@poly@map@variant{latin}{modern}{variant=modern}
\@set@locale@poly@map@variant{latin}{medieval}{variant=medieval}
\@set@locale@poly@map@sublang{russian}{orv}{spelling=old}
\@set@locale@poly@map@variant{russian}{luna1918}{spelling=new}
\@set@locale@poly@map@script{serbian}{Latn}{script=Latin}
\@set@locale@poly@map@script{serbian}{Cyrl}{script=Cyrillic}
\@set@locale@poly@map@variant{syriac}{western}{numerals=western}
\@set@locale@poly@map@variant{syriac}{eastern}{numerals=eastern}
```

Load polyglossia.

```
\RequirePackage{polyglossia}
\ForEachTrackedDialect{\this@dialect}%
{%
  \edef\this@root@lang{%
    \TrackedLanguageFromDialect{\this@dialect}}%
  \edef\this@sublang{%
    \GetTrackedDialectSubLang{\this@dialect}}%
  \edef\this@region{%
    \TrackedIsoCodeFromLanguage{3166-1}{\this@dialect}}%
  \edef\this@script{%
    \GetTrackedDialectScript{\this@dialect}}%
  \edef\this@variant{%
    \GetTrackedDialectVariant{\this@dialect}}%
```

Try to determine the options. Check the script mappings.

```
\def\@locale@poly@options{}%
\ifx\this@script\empty
\else
  \@if@locale@poly@map@script{\this@root@lang}{\this@script}%
  {%
    \edef\@locale@poly@options{%
      \@get@locale@poly@map@script
        {\this@root@lang}{\this@script}}%
```

```
    }%
    {}%
  \fi
```

Check the region mappings.

```
  \ifx\this@region\empty
  \else
    \@if@locale@poly@map@region{\this@root@lang}{\this@region}%
    {%
      \ifx\@locale@poly@options\empty
        \edef\@locale@poly@options{%
          \@get@locale@poly@map@region{\this@root@lang}{\this@region}}%
      \else
        \edef\@locale@poly@options{\@locale@poly@options,%
          \@get@locale@poly@map@region{\this@root@lang}{\this@region}}%
      \fi
    }%
    {}%
  \fi
```

Check the sub-language mappings.

```
  \ifx\this@sublang\empty
  \else
    \@if@locale@poly@map@sublang{\this@root@lang}{\this@sublang}%
    {%
      \ifx\@locale@poly@options\empty
        \edef\@locale@poly@options{%
          \@get@locale@poly@map@sublang
            {\this@root@lang}{\this@sublang}}%
      \else
        \edef\@locale@poly@options{\@locale@poly@options,%
          \@get@locale@poly@map@sublang
            {\this@root@lang}{\this@sublang}}%
      \fi
    }%
    {}%
  \fi
```

Check the variant mappings.

```
  \ifx\this@subvariant\empty
  \else
    \@if@locale@poly@map@variant{\this@root@lang}{\this@variant}%
    {%
      \ifx\@locale@poly@options\empty
        \edef\@locale@poly@options{%
```

```
            \@get@locale@poly@map@variant
              {\this@root@lang}{\this@variant}}%
        \else
          \edef\@locale@poly@options{\@locale@poly@options,%
            \@get@locale@poly@map@variant
              {\this@root@lang}{\this@variant}}%
        \fi
      }%
      {}%
    \fi
```

Set the language using either `\setmainlanguage` or `\setotherlanguage`.

```
    \ifx\this@dialect\LocaleMainDialect
      \edef\@locale@tmp{\noexpand\setmainlanguage
          [\@locale@poly@options]{\this@root@lang}}%
    \else
       \edef\@locale@tmp{\noexpand\setotherlanguage
          [\@locale@poly@options]{\this@root@lang}}%
    \fi
    \@locale@tmp
  }%
```

Redefine command to temporarily switch off punctuation adjustments. For example, when formatting the date or time.

```
  \def\localenopolypunct{\@locale@nopolypunct}%
}
```

Main Index

attributes
  currencies
    official, 21
    regional, 21
  ⟨*currency*⟩
    official, 30, 31
    region, 31
    sym, 30, 31, 57
    tex, 30, 31, 58
  ⟨*dialect*⟩
    curfmt, 30, 32
    currency, 30, 55
    currencysep, 29
    currencysym, 30, 57
    currencytex, 30, 58
    day.0, 24
    day.1, 24
    day.2, 24
    day.3, 24
    day.4, 24
    day.5, 24
    day.6, 24
    decfmt, 29, 32
    decsep, 29
    exp, 29
    firstday, 23
    fulldate, 23
    fulldatefmt, 24, 40
    fulldatetime, 23
    fulldatetimefmt, 24, 40
    fulltime, 23
    fulltimefmt, 24, 40
    groupsep, 18, 29
    intfmt, 29, 32
    langname, 22
    langtag, 22
    longdate, 23
    longdatefmt, 24, 40
    longdatetime, 23
    longdatetimefmt, 24, 40
    longtime, 23
    longtimefmt, 24, 40
    meddate, 23
    meddatefmt, 24, 40
    meddatetime, 23
    meddatetimefmt, 24, 40
    medtime, 23
    medtimefmt, 24, 40
    month.1, 26
    month.10, 26
    month.11, 26
    month.12, 26
    month.2, 26
    month.3, 26
    month.4, 26
    month.5, 26
    month.6, 26
    month.7, 26
    month.8, 26
    month.9, 26
    nativelangname, 22
    nativeregionname, 22
    nativevariantname, 22
    percent, 29
    perfmt, 30, 32
    permill, 29
    regionalcurrency, 30, 56
    regionname, 22
    shortdate, 23
    shortdatefmt, 24, 40
    shortdatetime, 23
    shortdatetimefmt, 24, 40

shortday.0, 24
shortday.1, 25
shortday.2, 25
shortday.3, 25
shortday.4, 25
shortday.5, 25
shortday.6, 25
shortmonth.1, 26
shortmonth.10, 27
shortmonth.11, 27
shortmonth.12, 27
shortmonth.2, 26
shortmonth.3, 27
shortmonth.4, 27
shortmonth.5, 27
shortmonth.6, 27
shortmonth.7, 27
shortmonth.8, 27
shortmonth.9, 27
shorttime, 23
shorttimefmt, 24, 40
standalone.day.0, 25
standalone.day.1, 25
standalone.day.2, 25
standalone.day.3, 25
standalone.day.4, 25
standalone.day.5, 25
standalone.day.6, 25
standalone.month.1, 27
standalone.month.10, 28
standalone.month.11, 28
standalone.month.12, 28
standalone.month.2, 27
standalone.month.3, 27
standalone.month.4, 27
standalone.month.5, 27
standalone.month.6, 27
standalone.month.7, 27
standalone.month.8, 28
standalone.month.9, 28
standalone.shortday.0, 25
standalone.shortday.1, 25
standalone.shortday.2, 25
standalone.shortday.3, 25
standalone.shortday.4, 26
standalone.shortday.5, 26
standalone.shortday.6, 26
standalone.shortmonth.1, 28
standalone.shortmonth.10, 28
standalone.shortmonth.11, 28
standalone.shortmonth.12, 28
standalone.shortmonth.2, 28
standalone.shortmonth.3, 28
standalone.shortmonth.4, 28
standalone.shortmonth.5, 28
standalone.shortmonth.6, 28
standalone.shortmonth.7, 28
standalone.shortmonth.8, 28
standalone.shortmonth.9, 28
timezone.⟨*zone*⟩.long, 29
timezone.⟨*zone*⟩.longdst, 29
timezone.⟨*zone*⟩.short, 29
timezone.⟨*zone*⟩.shortdst, 29
usesgroup, 29
variantname, 22
⟨*lang tag*⟩
tagtodialect, 20
⟨*region*⟩
currency, 30
dialect, 30
timezone
id, 21, 29

babel package, 1, 5, 11–13, 58, 66, 68, 73, 77, 84–86, 88, 156, 157

CJK package, 5, 84, 87, 88
CJKutf8 package, 5, 9, 66, 87
\CurrentLocaleApplyDateTimePattern, 40
\CurrentLocaleCurrency, 67
\CurrentLocaleCurrencyPattern, 30
\CurrentLocaleDate, 67
\CurrentLocaleDateTime, 23
\CurrentLocaleDecimalPattern, 29
\CurrentLocaleIntegerPattern, 29
\CurrentLocalePercentPattern, 30
\CurrentLocaleTime, 67

datetime package, 129, 130
datetime2 package, 5, 45, 47, 67, 73, 88, 128

etex, 7
etoolbox package, 5, 70

\fhyn, 6
fontawesome package, 5, 58
fontenc package, 1, 5, 13, 57, 67, 72, 73, 81, 155
fontspec package, 1, 5, 13, 18, 57, 72, 80, 155

ifluatex package, 5
ifxetex package, 5
inputenc package, 1, 5, 13, 57, 59, 67, 73, 81, 83, 87, 88, 92

java, 43

\LocaleApplyDateTimePattern, 39, 65
\LocaleCurrencyLabel, 30, 64
\LocaleCurrencyRegionalLabel, 30, 64
\LocaleCurrencySymbol, 30, 64
\LocaleCurrencyTeXSymbol, 30, 64
\LocaleDateTimeInfo, 40
\LocaleDayIndexFromRegion, 60
\LocaleDayName, 24, 60
\LocaleFirstDayIndex, 23, 60
\LocaleFullDate, 23, 61
\LocaleFullDateTime, 23
\LocaleFullTime, 23, 62
\LocaleGetDialectAttribute, 32
\LocaleIfNumericUsesGroup, 29, 64
\LocaleLanguageName, 59
\LocaleLanguageNativeName, 60
\LocaleLongDate, 23, 61
\LocaleLongDateTime, 23
\LocaleLongTime, 23, 62
\LocaleMediumDate, 23, 61
\LocaleMediumDateTime, 23
\LocaleMediumTime, 23, 62
\LocaleMonthName, 26, 61
\LocaleNumericDecimalSep, 29, 64
\LocaleNumericExponent, 29, 65
\LocaleNumericGroupSep, 29, 64
\LocaleNumericMonetarySep, 29, 65
\LocaleNumericPercent, 29
\LocaleNumericPermill, 29
\LocaleRegionName, 60
\LocaleRegionNativeName, 60
\LocaleShortDate, 23, 61
\LocaleShortDateTime, 23
\LocaleShortDayName, 24, 61
\LocaleShortMonthName, 26, 61
\LocaleShortTime, 23, 62
\LocaleStandaloneDayName, 24, 61
\LocaleStandaloneMonthName, 26, 61
\LocaleStandaloneShortDayName, 24, 61
\LocaleStandaloneShortMonthName, 26, 61
\LocaleVariantName, 60

package options:
- auto, 85
- babel, 86
- cjk, 87
- currency, 64, 67
- date, 62, 67
- datetime, 67
- datetime2
  - iso, 75
  - locale, 73
- fontenc, 8, 67
  - auto, 80, 81, 154
- fontspec, 80
- inputenc, 8, 67
  - auto, 81
- main, 8, 66
- other, 8, 66
- polyglossia, 87
- support, 8, 66
  - auto, 80
  - babel, 77
  - polyglossia, 77
- symbols, 58, 66
- time, 62, 67
- timedata, 38
- timedate, 67
- utf8, 87

\pdfcreationdate, 15, 92
pdftex, 7

pgfcalendar package, 47, 128–131, 133
pinyin package, 84, 88
polyglossia package, 1, 5, 11–13, 58, 66, 68, 73, 77, 84–86, 88, 154, 156, 159

\selectlanguage, 12
\selectlocale, 21, 58, 68
\show, 17, 58

tex-locale package, 5–7, 32, 39, 47, 58, 62, 66, 67, 73
texosquery, 5–9, 11, 14, 15, 17, 21, 29, 33, 38, 39, 46, 49, 52–54, 56, 92
texosquery package, 1, 5, 6, 12, 32, 33, 35, 36, 39, 40, 58, 66, 67, 72, 78, 90–95, 142, 145
texosquery-jre8, 1, 5, 43, 47, 49, 52, 53
\texosquerycurrencyeuro, 58
\texosquerycurrencypound, 58
\texosqueryfmtdatetime, 39
\texosqueryfmtnumber, 33
\texosquerypatfmtgroupsep, 32
textcomp package, 1, 5, 58, 72
\thyn, 6
\today, 45, 67
tracklang dialect label GBwelsh package, 47
tracklang package, 1, 5, 12, 16, 18, 21, 22, 42, 45, 46, 51, 54, 58, 59, 66, 70, 89, 90, 153, 154, 157
tracklang-scripts package, 5, 84

xeCJK package, 5, 66
xetex, 7
xfor package, 5, 70
xkeyval package, 5, 71

Code Index

Symbols

\@ 89
\@TrackLangAddToHook 74, 152
\@TrackLangEnvTerritory 114
\@classoptionslist 77
\@curroptions 77
\@declaredoptions 77
\@empty 77, 80, 81, 159
\@endfortrue 80, 81
\@expandtwoargs 77
\@firstoftwo 80, 85, 87
\@for 77, 78, 80, 81
\@get@locale@poly@map@region 159, 162
\@get@locale@poly@map@script 159, 161
\@get@locale@poly@map@sublang 160, 162
\@get@locale@poly@map@variant 160, 163
\@if@locale@poly@map@region 159, 162
\@if@locale@poly@map@script 159, 161
\@if@locale@poly@map@sublang 160, 162
\@if@locale@poly@map@variant 160, 162
\@ifpackageloaded 72, 73, 77, 88
\@locale@attrlist 99
\@locale@bbl@options 157–159
\@locale@cjklist 85
\@locale@currency@nr 76
\@locale@currency@val 76
\@locale@currentiscjk 87
\@locale@date@nr 75
\@locale@date@val 75
\@locale@datetime@nr 74
\@locale@datetime@val 74
\@locale@decfmt 151
\@locale@decfmt@exp 151, 152
\@locale@decfmt@frac 151, 152
\@locale@decfmt@int 151, 152
\@locale@decfmt@split 152
\@locale@declareoption 75
\@locale@dialect 115–125, 153, 154
\@locale@endparse@result 125, 126
\@locale@err 90, 94, 97, 153, 159
\@locale@fontenc 73, 80, 81, 83
\@locale@fontenc@opt 82
\@locale@fontspecfalse 72
\@locale@fontspectrue 72
\@locale@get@langenc@map 82
\@locale@get@scriptenc@map 82
\@locale@getencmap 83
\@locale@gobbleemptytorelax 152
\@locale@if@langenc@map 82
\@locale@if@scriptenc@map 82
\@locale@ifcjk 85, 87
\@locale@ifhasencmap 83
\@locale@iflatinscript 85, 87
\@locale@ifsupportbabelorpoly ... 86, 87, 156
\@locale@ifsupportcjk 85, 87
\@locale@ifsupportpinyin 85, 87, 88
\@locale@ifxeorlua 72, 77, 80, 85, 87
\@locale@info 95
\@locale@inputenc 73, 81, 83, 84, 87
\@locale@lang 82
\@locale@langenc@map 155, 156
\@locale@load@dtm 75, 88
\@locale@load@regional@dtm 73, 75
\@locale@loadinputenc 83, 84, 87, 88
\@locale@loadscripts 158
\@locale@newencmap 156
\@locale@next 113, 114, 124, 125
\@locale@nopolypunct 163
\@locale@os@currencycode 114, 140
\@locale@os@currencysym 114, 141
\@locale@os@currencytex 114, 142
\@locale@os@cursep 115
\@locale@os@decsep 114
\@locale@os@default 93
\@locale@os@exp 114
\@locale@os@groupsep 114
\@locale@os@region 114
\@locale@os@regionalcurrencycode .. 114, 141
\@locale@os@usesgroup 114
\@locale@parse@dateblock 115, 125

\@locale@parse@datefmtblock 117
\@locale@parse@datefmts 117
\@locale@parse@dates 117
\@locale@parse@datetimeblock 118
\@locale@parse@datetimefmtblock 118
\@locale@parse@datetimefmts 118
\@locale@parse@datetimeinfo 113
\@locale@parse@datetimes 118
\@locale@parse@default 113
\@locale@parse@default@cursep 115
\@locale@parse@endmonths 120
\@locale@parse@endshortmonths 120
\@locale@parse@endstandalone@months .. 122
\@locale@parse@endstandalone@shortmonths 122
\@locale@parse@filemod 114
\@locale@parse@maindata 114, 115
\@locale@parse@maindatablock 115
\@locale@parse@maindatalocaleblock ... 115
\@locale@parse@monthblock 119
\@locale@parse@months 119
\@locale@parse@numeric 123
\@locale@parse@numericblock 122
\@locale@parse@numericfmt 124
\@locale@parse@numericfmtblock 123
\@locale@parse@otherdata 124
\@locale@parse@otherdata@localeblock . 125
\@locale@parse@otherdatalocaleblock .. 125
\@locale@parse@persym 123
\@locale@parse@query 89
\@locale@parse@result 126
\@locale@parse@shortmonthblock 119
\@locale@parse@shortmonths 120
\@locale@parse@shortweekdayblock 119
\@locale@parse@shortweekdays 119
\@locale@parse@standalone@monthblock . 121
\@locale@parse@standalone@months 121
\@locale@parse@standalone@shortmonthblock 121
\@locale@parse@standalone@shortmonths 122
\@locale@parse@standalone@shortweekdayblock 121
\@locale@parse@standalone@shortweekdays 121
\@locale@parse@standalone@weekdayblock 120
\@locale@parse@standalone@weekdays ... 121
\@locale@parse@timeblock 117
\@locale@parse@timefmtblock 117
\@locale@parse@timefmts 118
\@locale@parse@times 117
\@locale@parse@timezonemap 124
\@locale@parse@timezones 124
\@locale@parse@weekdayblock 118
\@locale@parse@weekdays 119
\@locale@poly@options 161–163
\@locale@postparse@hook 147
\@locale@pre@query@params 79–81
\@locale@pre@query@parse 80, 81
\@locale@pre@query@parsecodeset 81
\@locale@pre@query@parsetag 80, 81
\@locale@pre@query@parsetagcodeset 81
\@locale@providemap 157
\@locale@query@params 94
\@locale@region 115, 116, 123
\@locale@restore@at 89, 154
\@locale@result 79, 80, 94, 95, 125, 126
\@locale@save@timezonemap 124
\@locale@script 82
\@locale@scriptenc@map 155
\@locale@select 152–154
\@locale@select@dialect 153
\@locale@set@today 73, 75
\@locale@support@nr 72
\@locale@support@val 72
\@locale@supportopt 72, 77, 80, 85
\@locale@symbols 72, 78
\@locale@tag 93, 94
\@locale@time@nr 76
\@locale@time@val 76
\@locale@timedata@nr 76
\@locale@timedata@val 76
\@locale@tmp 163
\@locale@trackall 81, 85
\@locale@trackedmain 70, 71
\@locale@undef@action 98
\@locale@unknown@currency 140–142
\@locale@warn 125, 143, 153, 154
\@localenum@fmt 151
\@secondofthree 86
\@secondoftwo 80, 87
\@set@locale@poly@map@region 159, 160
\@set@locale@poly@map@script 159, 161
\@set@locale@poly@map@sublang 160, 161
\@set@locale@poly@map@variant 160, 161
\@texosquery@argquote 90, 93
\@thirdofthree 86, 87
\@tracklang@dialects 82
\@tracklang@for 93, 98

\@tracklang@ifinlist 99
\@tracklang@ifundef 88, 97–99, 154–156, 159, 160
\@tracklang@namedef 154–156, 159, 160
\@tracklang@nameuse 156, 159, 160
\@tracklang@parselangtag 114
\@tracklang@pkgwarn 89, 90
\@tracklang@sanitize 90, 91
\@use@ption 77
\^ 90

A

\AnyTrackedLanguages 70

B

\begin 87

C

\catcode 89
\csname 77, 88, 89, 94–101, 143, 155
\CurrentLocaleApplyDateTimePattern *65*, 148, 150
\CurrentLocaleCurrency *64*, 145, 147, 149
\CurrentLocaleCurrencyPattern *65*, 148, 150, 152
\CurrentLocaleDate *61*, 74, 147, 149, 150
\CurrentLocaleDateTime *62*
\CurrentLocaleDayIndexFromRegion *60*, 147, 148
\CurrentLocaleDayName *60*, 144, 147, 148
\CurrentLocaleDecimalPattern *65*, 148, 150, 152
\CurrentLocaleDecimalSep ... *64*, 145, 148, 149
\CurrentLocaleExponent *65*, 145, 148, 149
\CurrentLocaleFirstDayIndex *60*, 147, 148
\CurrentLocaleFullDate *62*, 147, 148
\CurrentLocaleFullDateTime *63*, 147, 149
\CurrentLocaleFullTime *63*, 147, 148
\CurrentLocaleIfNumericUsesGroup *64*, 148, 149
\CurrentLocaleIntegerPattern *65*, 148, 150, 151
\CurrentLocaleLanguageName *59*, 147, 148
\CurrentLocaleLanguageNativeName *59*, 147, 148
\CurrentLocaleLongDate *62*, 147, 148
\CurrentLocaleLongDateTime *63*, 147, 149
\CurrentLocaleLongTime *63*, 147, 149
\CurrentLocaleMediumDate *63*, 147, 148
\CurrentLocaleMediumDateTime ... *63*, 147, 149
\CurrentLocaleMediumTime *63*, 147, 149
\CurrentLocaleMonetarySep .. *64*, 145, 148, 149
\CurrentLocaleMonthName *61*, 144, 147, 148
\CurrentLocaleNumericGroupSep *64*, 145, 147, 149
\CurrentLocalePercent 145, 148, 150
\CurrentLocalePercentPattern *65*, 148, 150, 152
\CurrentLocalePermill 145, 148, 150
\CurrentLocaleRegionName *60*, 147, 148
\CurrentLocaleRegionNativeName . *60*, 147, 148
\CurrentLocaleShortDate *63*, 147, 148
\CurrentLocaleShortDateTime *64*, 147, 149
\CurrentLocaleShortDayName . *60*, 144, 147, 148
\CurrentLocaleShortMonthName *61*, 144, 147, 148
\CurrentLocaleShortTime *63*, 147, 149
\CurrentLocaleStandaloneDayName *61*, 147, 148
\CurrentLocaleStandaloneMonthName *61*, 144, 147, 148
\CurrentLocaleStandaloneShortDayName *61*, 147, 148
\CurrentLocaleStandaloneShortMonthName *61*, 144, 147, 148
\CurrentLocaleTime *62*, 147, 149, 150
\CurrentLocaleVariantName *60*, 147, 148
\CurrentLocaleVariantNativeName *60*, 147, 148
\CurrentOption 77
\CurrentTrackedDialect 144, 145, 148

D

\DeclareOption 71
\DeclareOptionX 71
\def 71–73, 75–77, 79–81, 89–92, 95–131, 133, 136–161, 163
\define@boolkey 72
\define@choicekey 72, 74–76
\define@key 71–73
\do 77, 78, 80, 81, 93, 98
\dtmFridayIndex *48*
\DTMlangsetup 74
\dtmMondayIndex *47*
\dtmSaturdayIndex *48*
\DTMsetstyle 75
\dtmSundayIndex *48*
\dtmThursdayIndex *48*
\dtmTuesdayIndex *48*
\dtmWednesdayIndex *48*

E

\edef 70–73, 79–83, 85, 87, 89, 92–94, 115, 116, 152–154, 157, 158, 161–163
\else 70–72, 77–83, 89, 90, 93–95, 99–101, 114–116, 123–126, 129–135, 140–143, 147, 151–154, 158, 159, 161–163
\empty 78, 79, 82, 83, 93, 94, 99, 113–116, 123, 125, 140–143, 151, 152, 154, 158, 161, 162
\end 87
\endcsname 77, 88, 89, 94–101, 143, 155

\endinput 89
\errhelp 89
\errmessage 89
\expandafter 77, 80, 82, 83, 89, 92, 95–101, 126, 143, 152–154, 159

F

\fi 70–72, 75–82, 84, 87, 89, 90, 92–95, 99–101, 114–116, 123–126, 129–136, 140–143, 147, 150–154, 158, 159, 162, 163
\ForEachTrackedDialect 70, 74, 81, 85, 87, 152, 157, 161

G

\GetTrackedDialectScript 82, 161
\GetTrackedDialectSubLang 161
\GetTrackedDialectToMapping 154, 158
\GetTrackedDialectVariant 161
\GetTrackedLanguageTag 70
\global 95

I

\if@locale@fontspec 80
\ifcase 74–76, 85, 129–135
\ifcsstring 85
\ifdefempty 74
\ifdefstring 73, 78, 80, 81, 83, 87, 158
\IfFileExists 73, 83, 86, 158
\ifin@ 77
\ifluatex 71
\ifnum 80, 89, 100, 131, 134, 151
\ifTeXOSQueryDryRun 125, 153, 159
\IfTrackedDialect 153
\IfTrackedDialectHasMapping ... 154, 157, 158
\IfTrackedDialectIsScriptCs 158
\IfTrackedLanguageHasIsoCode 115, 116
\ifx 70, 71, 77–83, 89, 90, 92–94, 99–101, 113–116, 123–125, 140–143, 147, 150, 152–154, 158, 159, 161–163
\ifxetex 70
\in@ 77
\input 77, 81, 83, 88, 89
\inputencoding 83, 84

J

\jobname 92, 159

L

\languagename 88
\let 70, 71, 77, 78, 80, 81, 83–87, 95, 97, 99, 113–116, 124, 125, 151, 158
\listadd 84
\locale@bbl@dialect 158, 159
\locale@this@dialect 70, 74, 78, 80–82, 85, 87, 152
\LocaleAddToAttributeList *18*, 102, 106, 110, 123, 125
\LocaleAddToCurrencyAttributeList *20*
\LocaleAddToDialectAttributeList *19*
\LocaleAddToRegionAttributeList *19*
\LocaleApplyDateTimePattern *39*, 150
\LocaleAppToAttribute 96, 102, 106, 110
\localecur *34*
\localecurrchoice *64*, 76, 77, 149
\LocaleCurrencyLabel *55*, 149
\LocaleCurrencyRegionalLabel *56*, 149
\LocaleCurrencySymbol *56*, 149
\LocaleCurrencyTeXSymbol *57*, 149
\localedatechoice *61*, 74–76, 149
\localedatetimefmt .. *45*, 127, 128, 136, 137, 143
\LocaleDateTimeInfo *39*, 113, 114, 126
\LocaleDayIndexFromOneMonToZeroMon *50*, 131, 144
\LocaleDayIndexFromOneSunToZeroMon . *50*, 133
\LocaleDayIndexFromRegion *51*, 148
\LocaleDayIndexFromZeroMonToOneMon . *50*, 133
\LocaleDayIndexFromZeroMonToOneSun . *50*, 135
\LocaleDayIndexToRegion *51*
\LocaleDayName *48*, 148
\localedec *34*
\LocaleFileMod *15*, 114, 115, 125
\LocaleFirstDayIndex *49*, 131, 133, 148
\LocaleForEachInAttributeList *19*, 103, 107, 111
\LocaleForEachInCurrencyAttributeList . *20*
\LocaleForEachInDialectAttributeList .. *19*
\LocaleForEachInRegionAttributeList ... *20*
\LocaleFullDate *46*, 148, 149
\LocaleFullDateTime *53*, 149
\LocaleFullTime *52*, 148, 149
\LocaleGetAttribute *16*, 103, 107, 111, 153
\LocaleGetAttributeOrDefValue 98, 103, 107, 111
\LocaleGetCurrencyAttribute *17*
\LocaleGetCurrencyAttributeOrDefValue 142
\LocaleGetDialectAttribute *16*, 126–128, 136–145, 150
\LocaleGetDialectAttributeOrDefValue 128, 140, 141
\LocaleGetRegionAttribute *17*
\LocaleIfAttributeEqCs 104, 108, 112

\LocaleIfAttributeEqCsName 104, 108, 112
\LocaleIfAttributeEqNum 105, 109, 113
\LocaleIfDateTimePatternsSupported ....
..... *38*, 76, 93, 94, 113, 124, 144
\LocaleIfDialectAttributeEqCs 140–142
\LocaleIfDialectAttributeEqNum 146
\LocaleIfHasAttribute 95–99, 103, 107, 111, 153
\LocaleIfHasDialectNonEmptyAttribute ..
..... 143, 146
\LocaleIfHasLanguageName *44*
\LocaleIfHasNonEmptyAttribute .....
..... 100, 104, 108, 112
\LocaleIfHasRegionName *44*
\LocaleIfHasVariantName *45*
\LocaleIfInAttributeList . *19*, 96, 104, 108, 112
\LocaleIfInCurrencyAttributeList *20*
\LocaleIfInDialectAttributeList *19*
\LocaleIfInRegionAttributeList *20*
\LocaleIfNumericUsesGroup *54*, 149
\LocaleIfSameAttributeValues .. 105, 109, 113
\LocaleIfSameDialectAttributeValues ...
..... 123, 141, 142
\LocaleIfXpInAttributeList . 96, 104, 108, 112
\localeint *34*
\LocaleLanguageName *42*, 148
\LocaleLanguageNativeName *43*, 148
\LocaleLanguageTag *42*
\LocaleLetAttribute 102, 106, 110, 116
\LocaleLetRegionAttribute 115
\LocaleLongDate *46*, 148, 149
\LocaleLongDateTime *53*, 149
\LocaleLongTime *52*, 149
\LocaleMain *11*, 70, 71, 77, 78, 80, 93, 115, 125
\LocaleMainDialect .....
..... *42*, 115, 125, 140, 141, 154, 158, 163
\LocaleMainFile *15*, 93, 113, 114
\LocaleMainRegion *42*, 115, 125, 140–142
\LocaleMediumDate *46*, 148, 149
\LocaleMediumDateTime *53*, 149
\LocaleMediumTime *53*, 149
\LocaleMonthName *51*, 148
\localenopolypunct 163
\LocaleNowStamp *15*, 113, 125
\LocaleNumericDecimalSep *54*, 149
\LocaleNumericExponent *55*, 149
\LocaleNumericGroupSep *54*, 149
\LocaleNumericMonetarySep *55*, 149
\LocaleNumericPercent *55*, 150
\LocaleNumericPermill *55*, 150
\localenumfmt *33*, 151, 152
\localenumfmtneg *34*, 151
\localenumfmtpos *34*, 151
\localenumfmtzero *34*, 151
\LocaleOSarch *14*, 113, 125
\LocaleOScodeset *14*, 79, 80, 83, 113, 125
\LocaleOSname *14*, 113, 125
\LocaleOStag *15*, 78–80, 114, 125
\LocaleOSversion *14*, 113, 125
\LocaleOther *11*, 70–72, 77, 78, 80, 81, 93
\localeper *34*
\localepostquery *9*, 87
\localeprequery *9*, 87
\LocaleProvideAttribute 101, 105, 109
\LocaleProvideCurrencyAttribute ... 114, 123
\LocaleProvideRegionAttribute 123
\LocaleQueryCodesetParam *14*, 92
\LocaleQueryFile *6*, 94, 95
\LocaleRegionName *44*, 148
\LocaleRegionNativeName *44*, 148
\LocaleSetAttribute ... *16*, 96, 97, 101, 105, 109
\LocaleSetCurrencyAttribute *17*
\LocaleSetDialectAttribute *16*, 116–125
\LocaleSetRegionAttribute *16*, 114
\LocaleShortDate *46*, 148, 149
\LocaleShortDateTime *54*, 149
\LocaleShortDayName *49*, 148
\LocaleShortMonthName *52*, 148
\LocaleShortTime *53*, 149
\localeshowattribute *17*, 101, 105, 109
\localeshowcurrencyattribute *17*
\localeshowdialectattribute *17*
\localeshowregionattribute *17*
\LocaleStandaloneDayName *49*, 148
\LocaleStandaloneMonthName *52*, 148
\LocaleStandaloneShortDayName *49*, 148
\LocaleStandaloneShortMonthName *52*, 148
\LocaleStyQueryFile *7*, 79
\LocaleSupportPackageCase *68*
\localetimechoice *62*, 76, 149
\LocaleVariantName *44*, 148
\LocaleVariantNativeName *44*, 148
\LocaleXpAddToAttributeList *18*, 102, 106, 110
\LocaleXpAddToCurrencyAttributeList *20*, 123
\LocaleXpAddToDialectAttributeList *19*
\LocaleXpAddToRegionAttributeList .. *20*, 116
\LocaleXpAppToAttribute 96, 102, 106, 110

M

\makeatletter 87
\message 90
\MessageBreak 90, 159

N

\NeedsTeXFormat 70
\newcommand 71, 73, 74, 76–79, 84, 85, 87–89
\newcount 72
\newlinechar 90
\newrobustcmd 88
\noexpand 89, 163
\number 89, 137–139

O

\or 75–77, 85–87, 129–135

P

\PackageError 79, 89
\PackageInfo 79, 90
\PackageWarning 78, 82, 83, 88
\PassOptionsToPackage 74, 75, 82, 83, 159
\ProcessOptionsX 77
\providecommand 84
\ProvidesPackage 70

R

\relax 72, 77, 78, 80, 84, 89, 124, 125, 151, 152
\renewcommand 74, 83, 84, 87
\RequirePackage 70, 71, 74, 75, 78, 80, 82–84, 87, 88, 159, 161

S

\selectlanguage 153, 154
\selectlocale *12*
\SetCurrentTrackedDialect .. 74, 148, 152, 154
\setmainlanguage 163
\setotherlanguage 163
\SetTrackedDialectLabelMap 157, 158
\show 97
\space 92–94, 97, 150, 153, 154
\string 79–81, 92–94, 97, 153, 154

T

\TeXOSQuery 79, 94
\texosqueryfmtdatetime 143
\texosqueryfmtnumber 151
\TeXOSQueryFromFile 79, 94, 95
\texosquerystripquotes 92
\texosquerytimezonefmt 144, 145
\this@dialect 157, 158, 161, 163
\this@region 161, 162
\this@root@lang 85–87, 157, 158, 161–163
\this@script 161
\this@sublang 161, 162
\this@subvariant 162
\this@variant 161–163
\today 74
\TrackedIsoCodeFromLanguage ... 115, 116, 161
\TrackedLanguageFromDialect 82, 85, 87, 154, 157, 161
\TrackLangGetDefaultScript 82
\TrackLangLastTrackedDialect 115, 116
\TrackLangScriptLatn 158
\TrackLanguageTag 78, 115, 116

U

\undefined 70, 71, 77, 79, 89, 90, 92, 94, 147, 150, 153

X

\xifinlist 85

Change History

1.0 (2018-08-26)
General: Initial release 70