Region (Numeric) Support for datatool v3.0+ Package

Nicola L. C. Talbot

2025-03-01 version 1.0

This is the region (numeric) support for the datatool package (version 3.0+). The supplied `ldf` files need to be installed in addition to datatool. To ensure language support, you will also need to install the applicable language module (for example, datatool-english).

Contents

1 Introduction **5**
1.1 Non-Region Specific Settings 7

2 Supported Regions **8**
2.1 Region “AU” 8
2.2 Region “BE” 11
2.3 Region “CA” 14
2.4 Region “FK” 18
2.5 Region “GB” 21
2.6 Region “GG” 24
2.7 Region “GI” 28
2.8 Region “IE” 31
2.9 Region “IM” 34
2.10 Region “JE” 37
2.11 Region “NZ” 41
2.12 Region “US” 44
2.13 Region “ZA” 47

3 The Code **51**
3.1 datatool-AU.ldf 51
  3.1.1 Numbers and Currency 51
  3.1.2 Date and Time Parsing 52
  3.1.3 Options 57
  3.1.4 Hooks 60
3.2 datatool-BE.ldf 61
  3.2.1 Numbers and Currency 61
  3.2.2 Date and Time Parsing 63
  3.2.3 Options 67
  3.2.4 Hooks 70
3.3 datatool-CA.ldf 71
  3.3.1 Numbers and Currency 71
  3.3.2 Date and Time Parsing 73
  3.3.3 Options 77
  3.3.4 Hooks 80
3.4 datatool-FK.ldf 81
  3.4.1 Numbers and Currency 81
  3.4.2 Date and Time Parsing 83

3.4.3 Options . . . 88
3.4.4 Hooks . . . 90
3.5 datatool-GB.ldf . . . 91
3.5.1 Numbers and Currency . . . 92
3.5.2 Date and Time Parsing . . . 93
3.5.3 Options . . . 98
3.5.4 Hooks . . . 101
3.6 datatool-GG.ldf . . . 102
3.6.1 Numbers and Currency . . . 102
3.6.2 Date and Time Parsing . . . 103
3.6.3 Options . . . 108
3.6.4 Hooks . . . 111
3.7 datatool-GI.ldf . . . 112
3.7.1 Numbers and Currency . . . 112
3.7.2 Date and Time Parsing . . . 114
3.7.3 Options . . . 119
3.7.4 Hooks . . . 121
3.8 datatool-IE.ldf . . . 122
3.8.1 Numbers and Currency . . . 123
3.8.2 Date and Time Parsing . . . 124
3.8.3 Options . . . 129
3.8.4 Hooks . . . 131
3.9 datatool-IM.ldf . . . 132
3.9.1 Numbers and Currency . . . 132
3.9.2 Date and Time Parsing . . . 134
3.9.3 Options . . . 139
3.9.4 Hooks . . . 141
3.10 datatool-JE.ldf . . . 142
3.10.1 Numbers and Currency . . . 143
3.10.2 Date and Time Parsing . . . 144
3.10.3 Options . . . 149
3.10.4 Hooks . . . 152
3.11 datatool-NZ.ldf . . . 153
3.11.1 Numbers and Currency . . . 153
3.11.2 Date and Time Parsing . . . 154
3.11.3 Options . . . 159
3.11.4 Hooks . . . 162
3.12 datatool-US.ldf . . . 163
3.12.1 Numbers and Currency . . . 163
3.12.2 Date and Time Parsing . . . 165
3.12.3 Options . . . 170
3.12.4 Hooks . . . 172

3.13 datatool-ZA.ldf 174
    3.13.1 Numbers and Currency 174
    3.13.2 Date and Time Parsing 175
    3.13.3 Options 179
    3.13.4 Hooks 182
Index 184

1 Introduction

This bundle provides language-independent region support files for datatool v3.0+. The files simply need to be installed on TEX's path. (They will be ignored if a pre-3.0 version of datatool is installed.) The datatool-base package (which is automatically loaded by datatool) uses tracklang's interface for detecting localisation settings and finding the appropriate files. If you use babel or polyglossia, make sure that you specify the document dialects before the first package to load tracklang.

If the chosen document language does not have an associated region, no region support will be provided.

For example:

```
\usepackage[british]{babel}
\usepackage{datatool-base}
```

In this case, british is associated with region "GB" so datatool-base will load `datatool-GB.ldf` if it's on TEX's path.

Alternatively, if you are not using a language package, simply use the locales option. For example:

```
\usepackage[locales={en-GB}]{datatool-base}
```

The tracklang interface doesn't allow the language to be omitted, but datatool-base's locales option will check for any item in the list that simply consists of two uppercase letters. If found, the region will be added to any tracked dialects that don't have a region set. If no dialects have been tracked, the region will be tracked with the language set to und (undetermined). For example:

```
\documentclass{article}
\usepackage[locales={GB}]{datatool-base}
```

This is equivalent to:

```
\usepackage[locales={und-GB}]{datatool-base}
```

(since no language has been specified). Whereas

```
\usepackage[afrikaans,english]{babel}
\usepackage[locales={ZA}]{datatool-base}
```

will add the region ZA to afrikaans and english dialects.

If the language already has a region, or if there are multiple regions for a particular language, then you will need to include the language in the tag. For example:

```
\usepackage[locales={en-GB,en-IE}]{datatool-base}
```

Bear in mind that if tracklang can't determine the applicable dialect label for the captions hook, the settings may not be applied when the language changes in multilingual documents. In this case, you can either load tracklang before datatool-base and set up the appropriate mappings or just add the applicable `\DTL`⟨*tag*⟩`LocaleHook` command to the relevant captions hook.

Any option that can be passed to datatool-base can also be passed to datatool but if datatool-base has already been loaded, it will be too late to use the locales option. For example:

```
\usepackage[locales={en-GB}]{datatool}
```

But not:

Incorrect!

```
\usepackage{datatool-base}
\usepackage[locales={en-GB}]{datatool}
```

If another package that also loads tracklang is loaded first, then datatool-base can pick up the settings from that. For example:

```
\usepackage[en-GB]{datetime2}
\usepackage{datatool}
```

Supplementary packages provided with datatool can also have the locales provided. For example:

```
\usepackage[locales={en-GB}]{datagidx}
```

As with datatool, these supplementary packages internally load datatool-base so if that has already been loaded, then the localisation support should already have been set.

1.1 Non-Region Specific Settings

Some settings are not specific to regions, such as currency-symbol-style. These should be set in `\DTLsetup`. All settings relating to numbers and currency are set within the numeric option and all settings relating to dates and times are set within the datetime option. These settings are described in the datatool user guide.

For example:

```
\DTLsetup{
 numeric = {
    region-currency-prefix = smallcaps ,
    currency-symbol-style = symbol
  },
 datetime = {
   parse = auto-reformat
 }
}
```

Note that scientific notation isn't governed by regional styles, but the value can be encapsulated with `\si` (if siunitx has been loaded) with the auto-reformat numeric option.

```
\DTLsetup{numeric={auto-reformat}}
\DTLparse{\result}{1.5e+3}\result.
(Value: \DTLdatumvalue{\result}.)
```

1.5×10^3. (Value: 1.5e+3.)

2 Supported Regions

Only a limited number of regions are currently supported.

2.1 Region “AU”

The `datatool-AU.ldf` file provides support for region “AU” (Australia). This defines the “AUD” currency and sets it as the default currency. The default number group character is a comma, and the default decimal character is a dot.

If there’s no caption hook and you have multiple locales, you will need to set both the language and region hooks. For example, this document has:

```
\DTLenLocaleHook
\DTLAULocaleHook
```

Default Numeric Styles

```
\DTLdecimaltolocale{12345}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLdecimaltolocale{12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLdecimaltocurrency{-12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLmul\result{\result}{0.5}\result.
(Value: \DTLdatumvalue{\result}.)
```

12,345. (Value: 12345.)
12,345.678. (Value: 12345.678.)
-$12,345.68. (Value: -12345.678.)
-$6,172.84. (Value: -6172.839.)

Remember that date-time parsing needs to be enabled, if required.

Default Date-Time Styles

```
\DTLsetup{datetime={parse}}
\DTLparse\result{28/2/2025 15:45 AEDT}
String: \result.
Data type: \DTLgetDataTypeName{\DTLdatumtype{\result}}.
Numeric value: \DTLdatumvalue{\result}.

\ExplSyntaxOn
\datatool_extract_timestamp:NN \result \l_tmpa_tl
Time-stamp: ~ \l_tmpa_tl
\ExplSyntaxOff
```

String: 28/2/2025 15:45 AEDT. Data type: date-time. Numeric value: 2460735.114583333.
Time-stamp: 2025-02-28T15:45:00+01:00

Options may be set with `\DTLsetLocaleOptions{`**`AU`**`}{`⟨*key*=*val list*⟩`}`. Available options are listed below.

`/AU/currency-symbol-prefix`=⟨*boolean*⟩ (initially false)

If true, shows the region prefix before the currency symbol if the numeric setting currency-symbol-style is not set to “iso”. The prefix font may be changed to smallcaps or smaller with `\DTLsetup{ numeric = { region-currency-prefix =` ⟨*value*⟩ `} }` where ⟨*value*⟩ is smallcaps or smaller.

```
\DTLsetup{numeric={region-currency-prefix=smallcaps}}
\DTLsetLocaleOptions{AU}{currency-symbol-prefix}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

AU$12,345.68

`/AU/currency-symbol-position`=⟨*value*⟩ (initially before)

Adjusts the formatting currency style. If **`before`**, the currency symbol is placed before the value. If **`after`**, the currency symbol is placed after the value.

```
\DTLsetup{numeric={currency-symbol-style=iso}}
\DTLsetLocaleOptions{AU}{currency-symbol-position=after}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

12,345.68 AUD

`/AU/currency-symbol-sep`=⟨*value*⟩ (initially none)

Sets the separator to use between the currency symbol (not code) and the value. Permitted values: **`none`** (no space), **`thin-space`** (a thin space), **`space`** (a normal space), or **`nbsp`** (a non-breaking space).

```
\DTLsetLocaleOptions{AU}{currency-symbol-sep=thin-space}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

$ 12,345.68

`/AU/number-style`=⟨*setting*⟩ (initially official)

Sets the number group and decimal characters. The value may be one of: **`official`** (comma number group and decimal point), or **`unofficial`** (thin space number group and decimal point). In the case of **`unofficial`**, a normal space may also be used when parsing.

```
\DTLsetup{numeric={auto-reformat}}
\DTLsetLocaleOptions{AU}{number-style=unofficial}
\DTLdecimaltocurrency{12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLparse{\result}{\$12 345.678}\result.
(Value: \DTLdatumvalue{\result}.)
```

$12 345.68. (Value: 12345.678.)
$12 345.68. (Value: 12345.678.)

The remaining options relate to dates and times, which are still experimental. You need to enable date and time parsing with `\DTLsetup{ datetime = {parse} }`.

`/AU/date-style`=⟨*style*⟩ (initially mdyyyy)

Sets the current date style. The value may be one of: **`dmyyyy`** (day month year), **`mdyyyy`** (month day year), **`yyyymd`** (year month day), **`dmyy`** (day month 2-digit year), **`mdyy`** (month day 2-digit year), or **`yymd`** (2-digit year month day).

`/AU/date-variant`=⟨*style*⟩ (initially slash)

Sets the current numeric date separator. Allowed values: **`slash`** (/), **`hyphen`** (-), **`dot`** (.) or **`dialect`** (if the language supports it).

`/AU/time-variant`=⟨*style*⟩ (initially colon)

Sets the current numeric time separator. Allowed values: **`colon`** (:), **`dot`** (.) or **`dialect`** (if the language supports it), **`dialect-colon`** (if the language supports it), or **`dialect-dot`** (if the language supports it).

```
\DTLsetup{datetime={parse}}
\DTLsetLocaleOptions{AU}{
 date-style=mdyyyy,
 date-variant = dialect,
 time-variant = dot
}
\DTLparse\result{February 28, 2025 3.45pm AEDT}
String: \result.
Data type: \DTLgetDataTypeName{\DTLdatumtype{\result}}.
Numeric value: \DTLdatumvalue{\result}.

\ExplSyntaxOn
\datatool_extract_timestamp:NN \result \l_tmpa_tl
Time-stamp: ~ \l_tmpa_tl
\ExplSyntaxOff
```

String: February 28, 2025 3.45pm AEDT. Data type: date-time. Numeric value: 2460735.114583333. Time-stamp: 2025-02-28T15:45:00+01:00

`\datatoolAUSetNumberChars`

Hook to switch to AU number group and decimal characters.

`\datatoolAUcurrencyfmt`

Used to format the AUD currency. This supports the currency symbol prefix.

`\DTLAULocaleHook`

Hook to switch to AU settings. This may be added to the captions hook by datatool-base, depending on the settings. Otherwise it can be explicitly used to switch to this region.

2.2 Region "BE"

The `datatool-BE.ldf` file provides support for region "BE" (Belgium). This sets "EUR" as the default currency, but number formatting depends on the language. If no command called `\datatool`⟨*lang*⟩`BESetNumberChars` is defined (the applicable language module would need to provide `datatool-`⟨*lang*⟩`-BE.ldf` that defines this), this will default to a dot number group separator and a decimal comma.

For example, a French module would need to include the file `datatool-fr-BE.ldf` which defines `\datatoolfrBESetNumberChars`. (You can use `datatool-en-ZA.ldf` provided with datatool-english as an example.)

If there's no caption hook and you have multiple locales, you will need to set the region hook. For example, this document has:

```
\DTLBELocaleHook
```

Default Styles

```
\DTLdecimaltolocale{12345}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLdecimaltolocale{12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLdecimaltocurrency{-12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLmul\result{\result}{0,5}\result.
(Value: \DTLdatumvalue{\result}.)
```

12.345. (Value: 12345.)
12.345,678. (Value: 12345.678.)
-12.345,68€. (Value: -12345.678.)
-6.172,84€. (Value: -6172.839.)

Remember that date-time parsing needs to be enabled, if required.

Default Date-Time Styles

```
\DTLsetup{datetime={parse}}
\DTLparse\result{28.02.2025 15:45 CET}
String: \result.
Data type: \DTLgetDataTypeName{\DTLdatumtype{\result}}.
Numeric value: \DTLdatumvalue{\result}.

\ExplSyntaxOn
\datatool_extract_timestamp:NN \result \l_tmpa_tl
Time-stamp: ~ \l_tmpa_tl
\ExplSyntaxOff
```

String: 28.02.2025 15:45 CET. Data type: date-time. Numeric value: 2460735.114583333.
Time-stamp: 2025-02-28T15:45:00+01:00

Options may be set with `\DTLsetLocaleOptions{`**`BE`**`}{`⟨*key*=*val list*⟩`}`. Available options are listed below.

`/BE/currency-symbol-position`=⟨*value*⟩ (initially after)

Adjusts the formatting currency style. If **`before`**, the currency symbol is placed before the value. If **`after`**, the currency symbol is placed after the value.

```
\DTLsetup{numeric={currency-symbol-style=iso}}
\DTLsetLocaleOptions{BE}{currency-symbol-position=before}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

EUR 12.345,68

/BE/currency-symbol-sep=⟨*value*⟩ (initially none)

Sets the separator to use between the currency symbol (not code) and the value. Permitted values: **none** (no space), **thin-space** (a thin space), **space** (a normal space), or **nbsp** (a non-breaking space).

```
\DTLsetLocaleOptions{BE}{currency-symbol-sep=thin-space}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

12.345,68 €

/BE/number-style=⟨*style*⟩ (initially dialect)

Sets the number group and decimal characters. The value may be one of: **default** (dot number group and decimal comma), **dialect** (attempt to use the number style for the module ⟨*lang*⟩-BE, if it's defined, where ⟨*lang*⟩ is the current language tag, or fallback on **default**), or **thinspace** (thin space number group and decimal point). In the case of **thinspace**, a normal space may also be used when parsing.

```
\DTLsetLocaleOptions{BE}{number-style=thinspace}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

12 345.68€

The remaining options relate to dates and times, which are still experimental. You need to enable date and time parsing with `\DTLsetup{ datetime = {parse} }`.

/BE/date-style=⟨*style*⟩ (initially dmyyyy)

Sets the current date style. The value may be one of: **dmyyyy** (day month year), **mdyyyy** (month day year), **yyyymd** (year month day), **dmyy** (day month 2-digit year), **mdyy** (month day 2-digit year), or **yymd** (2-digit year month day).

/BE/date-variant=⟨*style*⟩ (initially dot)

Sets the current numeric date separator. Allowed values: **slash** (/), **hyphen** (-), **dot** (.) or **dialect** (if the language supports it).

/BE/time-variant=⟨*style*⟩ (initially colon)

Sets the current numeric time separator. Allowed values: **colon** (:), **dot** (.) or **dialect** (if the language supports it), **dialect-colon** (if the language supports it), or **dialect-dot** (if the language supports it).

```
\DTLsetup{datetime={parse}}
\DTLsetLocaleOptions{BE}{
 date-style=yyyymd,
 date-variant = hyphen
}
\DTLparse\result{2025-06-01 15:45 CEST}
String: \result.
Data type: \DTLgetDataTypeName{\DTLdatumtype{\result}}.
Numeric value: \DTLdatumvalue{\result}.

\ExplSyntaxOn
\datatool_extract_timestamp:NN \result \l_tmpa_tl
Time-stamp: ~ \l_tmpa_tl
\ExplSyntaxOff
```

String: 2025-06-01 15:45 CEST. Data type: date-time. Numeric value: 2460828.072916667.
Time-stamp: 2025-06-01T15:45:00+02:00

`\datatoolBESetNumberChars`

Hook to switch to BE number group and decimal characters.

`\datatoolBESetCurrency`

Hook to switch to the EUR currency, rounding to 2 decimal places, and redefines `\DTLdefaultEURcurrencyfmt` to reflect the currency-symbol-position setting.

`\DTLBELocaleHook`

Hook to switch to BE settings. This may be added to the captions hook by datatool-base, depending on the settings. Otherwise it can be explicitly used to switch to this region.

2.3 Region “CA”

The `datatool-CA.ldf` file provides support for region “CA” (Canada). This supplies the currency (CAD) but number formatting depends on the language, so this requires specific language & region files, which should be provided by the applicable language module. For example, datatool-english provides `datatool-en-CA.ldf`.

If there’s no caption hook and you have multiple locales, you will need to set both the language and region hooks. For example, this document has:

```
\DTLenLocaleHook
\DTLenCALocaleHook
\DTLCALocaleHook
```

Default Numeric Styles (with English)

```
\DTLdecimaltolocale{12345}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLdecimaltolocale{12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLdecimaltocurrency{-12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLmul\result{\result}{0.5}\result.
(Value: \DTLdatumvalue{\result}.)
```

12 345. (Value: 12345.)
12 345.678. (Value: 12345.678.)
-$12 345.68. (Value: -12345.678.)
-$6 172.84. (Value: -6172.839.)

Remember that date-time parsing needs to be enabled, if required.

Default Date-Time Styles

```
\DTLsetup{datetime={parse}}
\DTLparse\result{28/2/2025 15:45 PST}
String: \result.
Data type: \DTLgetDataTypeName{\DTLdatumtype{\result}}.
Numeric value: \DTLdatumvalue{\result}.

\ExplSyntaxOn
\datatool_extract_timestamp:NN \result \l_tmpa_tl
Time-stamp: ~ \l_tmpa_tl
\ExplSyntaxOff
```

String: 28/2/2025 15:45 PST. Data type: string. Numeric value: .
Time-stamp:

Options may be set with `\DTLsetLocaleOptions{CA}{⟨key=val list⟩}`. Available options are listed below.

`/CA/currency-symbol-prefix`=⟨*boolean*⟩ (initially false)

If true, shows the region prefix before the currency symbol if the numeric setting currency-symbol-style is not set to “iso”. The prefix font may be changed to smallcaps or smaller with the base numeric option region-currency-prefix setting.

For example:

```
\DTLsetup{numeric={region-currency-prefix=smallcaps}}
\DTLsetLocaleOptions{CA}{currency-symbol-prefix}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

CA$12 345.68

/CA/currency-symbol-position=⟨*value*⟩ (initially before)

Adjusts the formatting currency style. If **before**, the currency symbol is placed before the value. If **after**, the currency symbol is placed after the value.

```
\DTLsetup{numeric={currency-symbol-style=iso}}
\DTLsetLocaleOptions{CA}{currency-symbol-position=after}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

12 345.68 CAD

/CA/currency-symbol-sep=⟨*value*⟩ (initially none)

Sets the separator to use between the currency symbol (not code) and the value. Permitted values: **none** (no space), **thin-space** (a thin space), **space** (a normal space), or **nbsp** (a non-breaking space).

```
\DTLsetLocaleOptions{CA}{currency-symbol-sep=thin-space}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

$ 12 345.68

/CA/number-style=⟨*style*⟩

This option will attempt to set number-style=⟨*style*⟩ for the module ⟨*lang*⟩-CA, if it's defined. Where ⟨*lang*⟩ is the current language tag. For example, if the current language tag is "en", then this will be equivalent to `\DTLsetLocaleOptions{en-CA}{number-style=`⟨*style*⟩`}`.

For example, if `datatool-en-CA.ldf` has been loaded and the current locale is en-GB:

```
\DTLsetLocaleOptions{CA}{number-style=unofficial}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

$12,345.68

The remaining options relate to dates and times, which are still experimental. You need to enable date and time parsing with `\DTLsetup{ datetime = {parse} }`.

/CA/date-style=⟨*style*⟩ (initially yyyymd)

Sets the current date style. The value may be one of: **dmyyyy** (day month year), **mdyyyy** (month day year), **yyyymd** (year month day), **dmyy** (day month 2-digit year), **mdyy** (month day 2-digit year), or **yymd** (2-digit year month day).

/CA/date-variant=⟨*style*⟩ (initially hyphen)

Sets the current numeric date separator. Allowed values: **slash** (/), **hyphen** (-), **dot** (.) or **dialect** (if the language supports it).

/CA/time-variant=⟨*style*⟩ (initially colon)

Sets the current numeric time separator. Allowed values: **colon** (:), **dot** (.) or **dialect** (if the language supports it), **dialect-colon** (if the language supports it), or **dialect-dot** (if the language supports it).

```
\DTLsetup{datetime={parse}}
\DTLsetLocaleOptions{CA}{
 date-style=dmyyyy,
 date-variant = dialect,
 time-variant = dot
}
\DTLparse\result{Fri 28th Feb 2025 3.45pm PST}
String: \result.
Data type: \DTLgetDataTypeName{\DTLdatumtype{\result}}.
Numeric value: \DTLdatumvalue{\result}.

\ExplSyntaxOn
\datatool_extract_timestamp:NN \result \l_tmpa_tl
Time-stamp: ~ \l_tmpa_tl
\ExplSyntaxOff
```

String: Fri 28th Feb 2025 3.45pm PST. Data type: date-time. Numeric value: 2460735.489583333. Time-stamp: 2025-02-28T15:45:00-08:00

\datatoolCASetNumberChars

Hook to switch to CA number group and decimal characters but requires language support to be enabled as well otherwise it will simply trigger a warning.

\datatoolCAcurrencyfmt

Used to format the CAD currency. This supports the currency symbol prefix.

\DTLCALocaleHook

Hook to switch to CA settings. This may be added to the captions hook by datatool-base, depending on the settings. Otherwise it can be explicitly used to switch to this region.

2.4 Region "FK"

The `datatool-FK.ldf` file provides support for region "FK" (Falkland Islands). This defines the "FKP" currency and sets it as the default currency. The default number group character is a comma, and the default decimal character is a dot.

If there's no caption hook and you have multiple locales, you will need to set both the language and region hooks. For example, this document has:

```
\DTLenLocaleHook
\DTLFKLocaleHook
```

Default Numeric Styles

```
\DTLdecimaltolocale{12345}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLdecimaltolocale{12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLdecimaltocurrency{-12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLmul\result{\result}{0.5}\result.
(Value: \DTLdatumvalue{\result}.)
```

12,345. (Value: 12345.)
12,345.678. (Value: 12345.678.)
-£12,345.68. (Value: -12345.678.)
-£6,172.84. (Value: -6172.839.)

Remember that date-time parsing needs to be enabled, if required.

Default Date-Time Styles

```
\DTLsetup{datetime={parse}}
\DTLparse\result{28/2/2025 15:45 PST}
String: \result.
Data type: \DTLgetDataTypeName{\DTLdatumtype{\result}}.
Numeric value: \DTLdatumvalue{\result}.

\ExplSyntaxOn
\datatool_extract_timestamp:NN \result \l_tmpa_tl
Time-stamp: ~ \l_tmpa_tl
\ExplSyntaxOff
```

String: 28/2/2025 15:45 PST. Data type: unset. Numeric value: .
Time-stamp:

Options may be set with `\DTLsetLocaleOptions{FK}{⟨key=val list⟩}`. Available options are listed below.

`/FK/currency-symbol-prefix`=⟨*boolean*⟩ (initially false)

If true, shows the region prefix before the currency symbol if the numeric setting currency-symbol-style is not set to “iso”. The prefix font may be changed to smallcaps or smaller with `\DTLsetup{ numeric = { region-currency-prefix = ⟨value⟩ } }` where ⟨*value*⟩ is smallcaps or smaller.

```
\DTLsetup{numeric={region-currency-prefix=smallcaps}}
\DTLsetLocaleOptions{FK}{currency-symbol-prefix}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

FK£12,345.68

`/FK/currency-symbol-position`=⟨*value*⟩ (initially before)

Adjusts the formatting currency style. If **`before`**, the currency symbol is placed before the value. If **`after`**, the currency symbol is placed after the value.

```
\DTLsetup{numeric={currency-symbol-style=iso}}
\DTLsetLocaleOptions{FK}{currency-symbol-position=after}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

12,345.68 FKP

`/FK/currency-symbol-sep`=⟨*value*⟩ (initially none)

Sets the separator to use between the currency symbol (not code) and the value. Permitted values: **`none`** (no space), **`thin-space`** (a thin space), **`space`** (a normal space), or **`nbsp`** (a non-breaking space).

```
\DTLsetLocaleOptions{FK}{currency-symbol-sep=thin-space}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

£ 12,345.68

`/FK/number-style`=⟨*setting*⟩ (initially official)

Sets the number group and decimal characters. The value may be one of: **`official`** (comma number group and decimal point) or **`unofficial`** (thin space number group and decimal point). In the case of **`unofficial`**, a normal space may also be used when parsing.

```
\DTLsetup{numeric={auto-reformat}}
\DTLsetLocaleOptions{FK}{number-style=unofficial}
\DTLdecimaltocurrency{12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLparse{\result}{£12 345.678}\result.
(Value: \DTLdatumvalue{\result}.)
```

£12 345.68. (Value: 12345.678.)
£12 345.68. (Value: 12345.678.)

The remaining options relate to dates and times, which are still experimental. You need to enable date and time parsing with `\DTLsetup{ datetime = {parse} }`.

`/FK/date-style`=⟨*style*⟩ (initially dmyyyy)

Sets the current date style. The value may be one of: **`dmyyyy`** (day month year), **`mdyyyy`** (month day year), **`yyyymd`** (year month day), **`dmyy`** (day month 2-digit year), **`mdyy`** (month day 2-digit year), or **`yymd`** (2-digit year month day).

`/FK/date-variant`=⟨*style*⟩ (initially slash)

Sets the current numeric date separator. Allowed values: **`slash`** (/), **`hyphen`** (-), **`dot`** (.) or **`dialect`** (if the language supports it).

`/FK/time-variant`=⟨*style*⟩ (initially colon)

Sets the current numeric time separator. Allowed values: **`colon`** (:), **`dot`** (.) or **`dialect`** (if the language supports it), **`dialect-colon`** (if the language supports it), or **`dialect-dot`** (if the language supports it).

```
\DTLsetup{datetime={parse}}
\DTLsetLocaleOptions{FK}{
 date-style=dmyyyy,
 date-variant = dialect,
 time-variant = dot
}
\DTLparse\result{Fri 28th Feb 2025 3.45pm FKST}
String: \result.
Data type: \DTLgetDataTypeName{\DTLdatumtype{\result}}.
Numeric value: \DTLdatumvalue{\result}.

\ExplSyntaxOn
\datatool_extract_timestamp:NN \result \l_tmpa_tl
Time-stamp: ~ \l_tmpa_tl
\ExplSyntaxOff
```

String: Fri 28th Feb 2025 3.45pm FKST. Data type: date-time. Numeric value: 2460735.28125.
Time-stamp: 2025-02-28T15:45:00-03:00

`\datatoolFKSetNumberChars`

Hook to switch to FK number group and decimal characters.

`\datatoolFKcurrencyfmt`

Used to format the FKP currency. This supports the currency symbol prefix.

`\DTLFKLocaleHook`

Hook to switch to FK settings. This may be added to the captions hook by datatool-base, depending on the settings. Otherwise it can be explicitly used to switch to this region.

2.5 Region "GB"

The `datatool-GB.ldf` file provides support for region "GB" (United Kingdom). This defines the "GBP" currency and sets it as the default currency. The default number group character is a comma, and the default decimal character is a dot.

If there's no caption hook and you have multiple locales, you will need to set both the language and region hooks. For example, this document has:

```
\DTLenLocaleHook
\DTLGBLocaleHook
```

Default Numeric Styles

```
\DTLdecimaltolocale{12345}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLdecimaltolocale{12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLdecimaltocurrency{-12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLmul\result{\result}{0.5}\result.
(Value: \DTLdatumvalue{\result}.)
```

12,345. (Value: 12345.)
12,345.678. (Value: 12345.678.)
-£12,345.68. (Value: -12345.678.)
-£6,172.84. (Value: -6172.839.)

Remember that date-time parsing needs to be enabled, if required.

Default Date-Time Styles

```
\DTLsetup{datetime={parse}}
\DTLparse\result{28/2/2025 15:45 GMT}
String: \result.
Data type: \DTLgetDataTypeName{\DTLdatumtype{\result}}.
Numeric value: \DTLdatumvalue{\result}.

\ExplSyntaxOn
\datatool_extract_timestamp:NN \result \l_tmpa_tl
Time-stamp: ~ \l_tmpa_tl
\ExplSyntaxOff
```

String: 28/2/2025 15:45 GMT. Data type: date-time. Numeric value: 2460735.15625.
Time-stamp: 2025-02-28T15:45:00+00:00

Options may be set with `\DTLsetLocaleOptions{`**`GB`**`}{`⟨*key*=*val list*⟩`}`. Available options are listed below.

`/GB/currency-symbol-prefix`=⟨*boolean*⟩ (initially false)

If true, shows the region prefix before the currency symbol if the numeric setting currency-symbol-style is not set to “iso”. The prefix font may be changed to smallcaps or smaller with `\DTLsetup{ numeric = { region-currency-prefix =` ⟨*value*⟩ `} }` where ⟨*value*⟩ is smallcaps or smaller.

```
\DTLsetup{numeric={region-currency-prefix=smallcaps}}
\DTLsetLocaleOptions{GB}{currency-symbol-prefix}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

GB£12,345.68

`/GB/currency-symbol-position`=⟨*value*⟩ (initially before)

Adjusts the formatting currency style. If **`before`**, the currency symbol is placed before the value. If **`after`**, the currency symbol is placed after the value.

```
\DTLsetup{numeric={currency-symbol-style=iso}}
\DTLsetLocaleOptions{GB}{currency-symbol-position=after}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

12,345.68 GBP

`/GB/currency-symbol-sep`=⟨*value*⟩ (initially none)

Sets the separator to use between the currency symbol (not code) and the value. Permitted values: **`none`** (no space), **`thin-space`** (a thin space), **`space`** (a normal space), or **`nbsp`** (a non-breaking space).

```
\DTLsetLocaleOptions{GB}{currency-symbol-sep=thin-space}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

£ 12,345.68

/GB/number-style=⟨*setting*⟩ (initially official)

Sets the number group and decimal characters. The value may be one of: **official** (comma number group and decimal point), **education** (thin space number group and decimal point), or **old** (comma number group and mid dot decimal). In the case of **education**, a normal space may also be used when parsing. In the case of **old**, a normal dot may also be used when parsing.

```
\DTLsetup{numeric={auto-reformat}}
\DTLsetLocaleOptions{GB}{number-style=education}
\DTLdecimaltocurrency{12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLparse{\result}{£12 345.678}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLsetLocaleOptions{GB}{number-style=old}
\DTLdecimaltocurrency{12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLparse{\result}{£12,345.678}\result.
(Value: \DTLdatumvalue{\result}.)
```

£12 345.68. (Value: 12345.678.)
£12 345.68. (Value: 12345.678.)
£12,345·68. (Value: 12345.678.)
£12,345·68. (Value: 12345.678.)

The remaining options relate to dates and times, which are still experimental. You need to enable date and time parsing with `\DTLsetup{ datetime = {parse} }`.

/GB/date-style=⟨*style*⟩ (initially dmyyyy)

Sets the current date style. The value may be one of: **dmyyyy** (day month year), **mdyyyy** (month day year), **yyyymd** (year month day), **dmyy** (day month 2-digit year), **mdyy** (month day 2-digit year), or **yymd** (2-digit year month day).

/GB/date-variant=⟨*style*⟩ (initially slash)

Sets the current numeric date separator. Allowed values: **slash** (/), **hyphen** (-), **dot** (.) or **dialect** (if the language supports it).

/GB/time-variant=⟨*style*⟩ (initially colon)

Sets the current numeric time separator. Allowed values: **colon** (:), **dot** (.) or **dialect** (if the language supports it), **dialect-colon** (if the language supports it), or **dialect-dot** (if the language supports it).

```
\DTLsetup{datetime={parse}}
\DTLsetLocaleOptions{GB}{
 date-style=dmyyyy,
 date-variant = dialect,
 time-variant = dot
}
\DTLparse\result{Fri 28th Feb 2025 3.45pm GMT}
String: \result.
Data type: \DTLgetDataTypeName{\DTLdatumtype{\result}}.
Numeric value: \DTLdatumvalue{\result}.

\ExplSyntaxOn
\datatool_extract_timestamp:NN \result \l_tmpa_tl
Time-stamp: ~ \l_tmpa_tl
\ExplSyntaxOff
```

String: Fri 28th Feb 2025 3.45pm GMT. Data type: date-time. Numeric value: 2460735.15625.
Time-stamp: 2025-02-28T15:45:00+00:00

`\datatoolGBSetNumberChars`

Hook to switch to GB number group and decimal characters.

`\datatoolGBcurrencyfmt`

Used to format the GBP currency. This supports the currency symbol prefix.

`\DTLGBLocaleHook`

Hook to switch to GB settings. This may be added to the captions hook by datatool-base, depending on the settings. Otherwise it can be explicitly used to switch to this region.

2.6 Region “GG”

The `datatool-GG.ldf` file provides support for region “GG” (Guernsey). This defines the “GGP” currency and sets it as the default currency. The default number group character is a comma, and the default decimal character is a dot.

If there’s no caption hook and you have multiple locales, you will need to set both the language and region hooks. For example, this document has:

```
\DTLenLocaleHook
\DTLGGLocaleHook
```

Default Numeric Styles

```
\DTLdecimaltolocale{12345}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLdecimaltolocale{12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLdecimaltocurrency{-12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLmul\result{\result}{0.5}\result.
(Value: \DTLdatumvalue{\result}.)
```

12,345. (Value: 12345.)
12,345.678. (Value: 12345.678.)
-£12,345.68. (Value: -12345.678.)
-£6,172.84. (Value: -6172.839.)

Remember that date-time parsing needs to be enabled, if required.

Default Date-Time Styles

```
\DTLsetup{datetime={parse}}
\DTLparse\result{28/2/2025 15:45 GMT}
String: \result.
Data type: \DTLgetDataTypeName{\DTLdatumtype{\result}}.
Numeric value: \DTLdatumvalue{\result}.

\ExplSyntaxOn
\datatool_extract_timestamp:NN \result \l_tmpa_tl
Time-stamp: ~ \l_tmpa_tl
\ExplSyntaxOff
```

String: 28/2/2025 15:45 GMT. Data type: date-time. Numeric value: 2460735.15625.
Time-stamp: 2025-02-28T15:45:00+00:00

Options may be set with `\DTLsetLocaleOptions{GG}{`⟨*key=val list*⟩`}`. Available options are listed below.

`/GG/currency-symbol-prefix`=⟨*boolean*⟩ (initially false)

If true, shows the region prefix before the currency symbol if the numeric setting currency-symbol-style is not set to “iso”. The prefix font may be changed to smallcaps or smaller with `\DTLsetup{ numeric = { region-currency-prefix = `⟨*value*⟩` } }` where ⟨*value*⟩ is smallcaps or smaller.

```
\DTLsetup{numeric={region-currency-prefix=smallcaps}}
\DTLsetLocaleOptions{GG}{currency-symbol-prefix}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

GG£12,345.68

/GG/currency-symbol-position=⟨*value*⟩ (initially before)

Adjusts the formatting currency style. If **before**, the currency symbol is placed before the value. If **after**, the currency symbol is placed after the value.

```
\DTLsetup{numeric={currency-symbol-style=iso}}
\DTLsetLocaleOptions{GG}{currency-symbol-position=after}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

12,345.68 GGP

/GG/currency-symbol-sep=⟨*value*⟩ (initially none)

Sets the separator to use between the currency symbol (not code) and the value. Permitted values: **none** (no space), **thin-space** (a thin space), **space** (a normal space), or **nbsp** (a non-breaking space).

```
\DTLsetLocaleOptions{GG}{currency-symbol-sep=thin-space}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

£ 12,345.68

/GG/number-style=⟨*setting*⟩ (initially official)

Sets the number group and decimal characters. The value may be one of: **official** (comma number group and decimal point) or **unofficial** (thin space number group and decimal point). In the case of **unofficial**, a normal space may also be used when parsing.

```
\DTLsetup{numeric={auto-reformat}}
\DTLsetLocaleOptions{GG}{number-style=unofficial}
\DTLdecimaltocurrency{12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLparse{\result}{£12 345.678}\result.
(Value: \DTLdatumvalue{\result}.)
```

£12 345.68. (Value: 12345.678.)
£12 345.68. (Value: 12345.678.)

The remaining options relate to dates and times, which are still experimental. You need to enable date and time parsing with `\DTLsetup{ datetime = {parse} }`.

`/GG/date-style`=⟨*style*⟩ (initially dmyyyy)

Sets the current date style. The value may be one of: **`dmyyyy`** (day month year), **`mdyyyy`** (month day year), **`yyyymd`** (year month day), **`dmyy`** (day month 2-digit year), **`mdyy`** (month day 2-digit year), or **`yymd`** (2-digit year month day).

`/GG/date-variant`=⟨*style*⟩ (initially slash)

Sets the current numeric date separator. Allowed values: **`slash`** (/), **`hyphen`** (-), **`dot`** (.) or **`dialect`** (if the language supports it).

`/GG/time-variant`=⟨*style*⟩ (initially colon)

Sets the current numeric time separator. Allowed values: **`colon`** (:), **`dot`** (.) or **`dialect`** (if the language supports it), **`dialect-colon`** (if the language supports it), or **`dialect-dot`** (if the language supports it).

```
\DTLsetup{datetime={parse}}
\DTLsetLocaleOptions{GG}{
 date-style=dmyyyy,
 date-variant = dialect,
 time-variant = dot
}
\DTLparse\result{Fri 28th Feb 2025 3.45pm GMT}
String: \result.
Data type: \DTLgetDataTypeName{\DTLdatumtype{\result}}.
Numeric value: \DTLdatumvalue{\result}.

\ExplSyntaxOn
\datatool_extract_timestamp:NN \result \l_tmpa_tl
Time-stamp: ~ \l_tmpa_tl
\ExplSyntaxOff
```

String: Fri 28th Feb 2025 3.45pm GMT. Data type: date-time. Numeric value: 2460735.15625.
Time-stamp: 2025-02-28T15:45:00+00:00

`\datatoolGGSetNumberChars`

Hook to switch to GG number group and decimal characters.

`\datatoolGGcurrencyfmt`

Used to format the GGP currency. This supports the currency symbol prefix.

`\DTLGGLocaleHook`

Hook to switch to GG settings. This may be added to the captions hook by datatool-base, depending on the settings. Otherwise it can be explicitly used to switch to this region.

2.7 Region “GI”

The `datatool-GI.ldf` file provides support for region “GI” (Gibraltar). This defines the “GIP” currency and sets it as the default currency. The default number group character is a comma, and the default decimal character is a dot.

If there’s no caption hook and you have multiple locales, you will need to set both the language and region hooks. For example, this document has:

```
\DTLenLocaleHook
\DTLGILocaleHook
```

Default Numeric Styles

```
\DTLdecimaltolocale{12345}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLdecimaltolocale{12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLdecimaltocurrency{-12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLmul\result{\result}{0.5}\result.
(Value: \DTLdatumvalue{\result}.)
```

12,345. (Value: 12345.)
12,345.678. (Value: 12345.678.)
-£12,345.68. (Value: -12345.678.)
-£6,172.84. (Value: -6172.839.)

Remember that date-time parsing needs to be enabled, if required.

Default Date-Time Styles

```
\DTLsetup{datetime={parse}}
\DTLparse\result{28/2/2025 15:45 GMT}
String: \result.
Data type: \DTLgetDataTypeName{\DTLdatumtype{\result}}.
Numeric value: \DTLdatumvalue{\result}.

\ExplSyntaxOn
\datatool_extract_timestamp:NN \result \l_tmpa_tl
Time-stamp: ~ \l_tmpa_tl
\ExplSyntaxOff
```

String: 28/2/2025 15:45 GMT. Data type: date-time. Numeric value: 2460735.15625.
Time-stamp: 2025-02-28T15:45:00+00:00

Options may be set with `\DTLsetLocaleOptions{GI}{⟨key=val list⟩}`. Available options are listed below.

`/GI/currency-symbol-prefix`=⟨*boolean*⟩ (initially false)

If true, shows the region prefix before the currency symbol if the numeric setting currency-symbol-style is not set to “iso”. The prefix font may be changed to smallcaps or smaller with `\DTLsetup{ numeric = { region-currency-prefix = ⟨value⟩ } }` where ⟨*value*⟩ is smallcaps or smaller.

```
\DTLsetup{numeric={region-currency-prefix=smallcaps}}
\DTLsetLocaleOptions{GI}{currency-symbol-prefix}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

GI£12,345.68

`/GI/currency-symbol-position`=⟨*value*⟩ (initially before)

Adjusts the formatting currency style. If **`before`**, the currency symbol is placed before the value. If **`after`**, the currency symbol is placed after the value.

```
\DTLsetup{numeric={currency-symbol-style=iso}}
\DTLsetLocaleOptions{GI}{currency-symbol-position=after}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

12,345.68 GIP

`/GI/currency-symbol-sep`=⟨*value*⟩ (initially none)

Sets the separator to use between the currency symbol (not code) and the value. Permitted values: **`none`** (no space), **`thin-space`** (a thin space), **`space`** (a normal space), or **`nbsp`** (a non-breaking space).

```
\DTLsetLocaleOptions{GI}{currency-symbol-sep=thin-space}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

£ 12,345.68

`/GI/number-style`=⟨*setting*⟩ (initially official)

Sets the number group and decimal characters. The value may be one of: **`official`** (comma number group and decimal point) or **`unofficial`** (thin space number group and decimal point). In the case of **`unofficial`**, a normal space may also be used when parsing.

```
\DTLsetup{numeric={auto-reformat}}
\DTLsetLocaleOptions{GI}{number-style=unofficial}
\DTLdecimaltocurrency{12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLparse{\result}{£12 345.678}\result.
(Value: \DTLdatumvalue{\result}.)
```

£12 345.68. (Value: 12345.678.)
£12 345.68. (Value: 12345.678.)

The remaining options relate to dates and times, which are still experimental. You need to enable date and time parsing with `\DTLsetup{ datetime = {parse} }`.

/GI/date-style=⟨*style*⟩ (initially dmyyyy)

Sets the current date style. The value may be one of: **dmyyyy** (day month year), **mdyyyy** (month day year), **yyyymd** (year month day), **dmyy** (day month 2-digit year), **mdyy** (month day 2-digit year), or **yymd** (2-digit year month day).

/GI/date-variant=⟨*style*⟩ (initially slash)

Sets the current numeric date separator. Allowed values: **slash** (/), **hyphen** (-), **dot** (.) or **dialect** (if the language supports it).

/GI/time-variant=⟨*style*⟩ (initially colon)

Sets the current numeric time separator. Allowed values: **colon** (:), **dot** (.) or **dialect** (if the language supports it), **dialect-colon** (if the language supports it), or **dialect-dot** (if the language supports it).

```
\DTLsetup{datetime={parse}}
\DTLsetLocaleOptions{GI}{
 date-style=dmyyyy,
 date-variant = dialect,
 time-variant = dot
}
\DTLparse\result{Fri 28th Feb 2025 3.45pm GMT}
String: \result.
Data type: \DTLgetDataTypeName{\DTLdatumtype{\result}}.
Numeric value: \DTLdatumvalue{\result}.

\ExplSyntaxOn
\datatool_extract_timestamp:NN \result \l_tmpa_tl
Time-stamp: ~ \l_tmpa_tl
\ExplSyntaxOff
```

String: Fri 28th Feb 2025 3.45pm GMT. Data type: date-time. Numeric value: 2460735.15625.
Time-stamp: 2025-02-28T15:45:00+00:00

`\datatoolGISetNumberChars`

Hook to switch to GI number group and decimal characters.

`\datatoolGIcurrencyfmt`

Used to format the GIP currency. This supports the currency symbol prefix.

`\DTLGILocaleHook`

Hook to switch to GI settings. This may be added to the captions hook by datatool-base, depending on the settings. Otherwise it can be explicitly used to switch to this region.

2.8 Region “IE”

The `datatool-IE.ldf` file provides support for region “IE” (Republic of Ireland). This sets “EUR” as the default currency.

If there’s no caption hook and you have multiple locales, you will need to set both the language and region hooks. For example, this document has:

```
\DTLenLocaleHook
\DTLIELocaleHook
```

Default Numeric Styles

```
\DTLdecimaltolocale{12345}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLdecimaltolocale{12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLdecimaltocurrency{-12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLmul\result{\result}{0.5}\result.
(Value: \DTLdatumvalue{\result}.)
```

12,345. (Value: 12345.)
12,345.678. (Value: 12345.678.)
-€12,345.68. (Value: -12345.678.)
-€6,172.84. (Value: -6172.839.)

Remember that date-time parsing needs to be enabled, if required.

Default Date-Time Styles

```
\DTLsetup{datetime={parse}}
\DTLparse\result{1/6/2025 15:45 IST}
String: \result.
Data type: \DTLgetDataTypeName{\DTLdatumtype{\result}}.
Numeric value: \DTLdatumvalue{\result}.

\ExplSyntaxOn
\datatool_extract_timestamp:NN \result \l_tmpa_tl
Time-stamp: ~ \l_tmpa_tl
\ExplSyntaxOff
```

String: 1/6/2025 15:45 IST. Data type: date-time. Numeric value: 2460828.114583333.
Time-stamp: 2025-06-01T15:45:00+01:00

Options may be set with `\DTLsetLocaleOptions{IE}{`⟨*key*=*val list*⟩`}`. Available options are listed below.

`/IE/currency-symbol-position`=⟨*value*⟩ (initially before)

Adjusts the formatting currency style. If **`before`**, the currency symbol is placed before the value. If **`after`**, the currency symbol is placed after the value.

```
\DTLsetup{numeric={currency-symbol-style=iso}}
\DTLsetLocaleOptions{IE}{currency-symbol-position=after}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

12,345.68 EUR

`/IE/currency-symbol-sep`=⟨*value*⟩ (initially none)

Sets the separator to use between the currency symbol (not code) and the value. Permitted values: **`none`** (no space), **`thin-space`** (a thin space), **`space`** (a normal space), or **`nbsp`** (a non-breaking space).

```
\DTLsetLocaleOptions{IE}{currency-symbol-sep=thin-space}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

€ 12,345.68

`/IE/number-style`=⟨*setting*⟩ (initially official)

Sets the number group and decimal characters. The value may be one of: **`official`** (comma number group and decimal point) or **`unofficial`** (thin space number group and decimal point). In the case of **`unofficial`**, a normal space may also be used when parsing.

```
\DTLsetup{numeric={auto-reformat}}
\DTLsetLocaleOptions{IE}{number-style=unofficial}
\DTLdecimaltocurrency{12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLparse{\result}{\texteuro 12 345.678}\result.
(Value: \DTLdatumvalue{\result}.)
```

€12 345.68. (Value: 12345.678.)
€12 345.68. (Value: 12345.678.)

The remaining options relate to dates and times, which are still experimental. You need to enable date and time parsing with `\DTLsetup{ datetime = { parse } }`.

`/IE/date-style`=⟨*style*⟩ (initially dmyyyy)

Sets the current date style. The value may be one of: **`dmyyyy`** (day month year), **`mdyyyy`** (month day year), **`yyyymd`** (year month day), **`dmyy`** (day month 2-digit year), **`mdyy`** (month day 2-digit year), or **`yymd`** (2-digit year month day).

`/IE/date-variant`=⟨*style*⟩ (initially slash)

Sets the current numeric date separator. Allowed values: **`slash`** (/), **`hyphen`** (-), **`dot`** (.) or **`dialect`** (if the language supports it).

`/IE/time-variant`=⟨*style*⟩ (initially colon)

Sets the current numeric time separator. Allowed values: **`colon`** (:), **`dot`** (.) or **`dialect`** (if the language supports it), **`dialect-colon`** (if the language supports it), or **`dialect-dot`** (if the language supports it).

```
\DTLsetup{datetime={parse}}
\DTLsetLocaleOptions{IE}{
 date-style=dmyyyy,
 date-variant = dialect,
 time-variant = dot
}
\DTLparse\result{1st June 2025 3.45pm IST}
String: \result.
Data type: \DTLgetDataTypeName{\DTLdatumtype{\result}}.
Numeric value: \DTLdatumvalue{\result}.

\ExplSyntaxOn
\datatool_extract_timestamp:NN \result \l_tmpa_tl
Time-stamp: ~ \l_tmpa_tl
\ExplSyntaxOff
```

String: 1st June 2025 3.45pm IST. Data type: date-time. Numeric value: 2460828.114583333.
Time-stamp: 2025-06-01T15:45:00+01:00

`\datatoolIESetNumberChars`

Hook to switch to IE number group and decimal characters.

`\datatoolIESetCurrency`

Hook to switch to the EUR currency, rounding to 2 decimal places, and redefines `\DTLdefaultEURcurrencyfmt` to reflect the currency-symbol-position setting.

`\DTLIELocaleHook`

Hook to switch to IE settings. This may be added to the captions hook by datatool-base, depending on the settings. Otherwise it can be explicitly used to switch to this region.

2.9 Region "IM"

The `datatool-IM.ldf` file provides support for region "IM" (Isle of Man). This defines the "IMP" currency and sets it as the default currency. The default number group character is a comma, and the default decimal character is a dot.

If there's no caption hook and you have multiple locales, you will need to set both the language and region hooks. For example, this document has:

```
\DTLenLocaleHook
\DTLIMLocaleHook
```

Default Numeric Styles

```
\DTLdecimaltolocale{12345}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLdecimaltolocale{12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLdecimaltocurrency{-12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLmul\result{\result}{0.5}\result.
(Value: \DTLdatumvalue{\result}.)
```

12,345. (Value: 12345.)
12,345.678. (Value: 12345.678.)
-£12,345.68. (Value: -12345.678.)
-£6,172.84. (Value: -6172.839.)

Remember that date-time parsing needs to be enabled, if required.

Default Date-Time Styles

```
\DTLsetup{datetime={parse}}
\DTLparse\result{28/2/2025 15:45 GMT}
String: \result.
Data type: \DTLgetDataTypeName{\DTLdatumtype{\result}}.
Numeric value: \DTLdatumvalue{\result}.

\ExplSyntaxOn
\datatool_extract_timestamp:NN \result \l_tmpa_tl
Time-stamp: ~ \l_tmpa_tl
\ExplSyntaxOff
```

String: 28/2/2025 15:45 GMT. Data type: date-time. Numeric value: 2460735.15625.
Time-stamp: 2025-02-28T15:45:00+00:00

Options may be set with `\DTLsetLocaleOptions{IM}{`⟨*key*=*val list*⟩`}`. Available options are listed below.

`/IM/currency-symbol-prefix`=⟨*boolean*⟩ (initially false)

If true, shows the region prefix before the currency symbol if the numeric setting currency-symbol-style is not set to “iso”. The prefix font may be changed to smallcaps or smaller with `\DTLsetup{ numeric = { region-currency-prefix =` ⟨*value*⟩ `} }` where ⟨*value*⟩ is smallcaps or smaller.

```
\DTLsetup{numeric={region-currency-prefix=smallcaps}}
\DTLsetLocaleOptions{IM}{currency-symbol-prefix}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

IM£12,345.68

`/IM/currency-symbol-position`=⟨*value*⟩ (initially before)

Adjusts the formatting currency style. If **`before`**, the currency symbol is placed before the value. If **`after`**, the currency symbol is placed after the value.

```
\DTLsetup{numeric={currency-symbol-style=iso}}
\DTLsetLocaleOptions{IM}{currency-symbol-position=after}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

12,345.68 IMP

`/IM/currency-symbol-sep`=⟨*value*⟩ (initially none)

Sets the separator to use between the currency symbol (not code) and the value. Permitted values: **`none`** (no space), **`thin-space`** (a thin space), **`space`** (a normal space), or **`nbsp`** (a non-breaking space).

```
\DTLsetLocaleOptions{IM}{currency-symbol-sep=thin-space}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

£ 12,345.68

`/IM/number-style`=⟨*setting*⟩ (initially official)

Sets the number group and decimal characters. The value may be one of: **`official`** (comma number group and decimal point) or **`unofficial`** (thin space number group and decimal point). In the case of **`unofficial`**, a normal space may also be used when parsing.

```
\DTLsetup{numeric={auto-reformat}}
\DTLsetLocaleOptions{IM}{number-style=unofficial}
\DTLdecimaltocurrency{12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLparse{\result}{£12 345.678}\result.
(Value: \DTLdatumvalue{\result}.)
```

£12 345.68. (Value: 12345.678.)
£12 345.68. (Value: 12345.678.)

The remaining options relate to dates and times, which are still experimental. You need to enable date and time parsing with `\DTLsetup{ datetime = {parse} }`.

`/IM/date-style`=⟨*style*⟩ (initially dmyyyy)

Sets the current date style. The value may be one of: **`dmyyyy`** (day month year), **`mdyyyy`** (month day year), **`yyyymd`** (year month day), **`dmyy`** (day month 2-digit year), **`mdyy`** (month day 2-digit year), or **`yymd`** (2-digit year month day).

`/IM/date-variant`=⟨*style*⟩ (initially slash)

Sets the current numeric date separator. Allowed values: **`slash`** (/), **`hyphen`** (-), **`dot`** (.) or **`dialect`** (if the language supports it).

`/IM/time-variant`=⟨*style*⟩ (initially colon)

Sets the current numeric time separator. Allowed values: **`colon`** (:), **`dot`** (.) or **`dialect`** (if the language supports it), **`dialect-colon`** (if the language supports it), or **`dialect-dot`** (if the language supports it).

```
\DTLsetup{datetime={parse}}
\DTLsetLocaleOptions{IM}{
 date-style=dmyyyy,
 date-variant = dialect,
 time-variant = dot
}
\DTLparse\result{Fri 28th Feb 2025 3.45pm GMT}
String: \result.
Data type: \DTLgetDataTypeName{\DTLdatumtype{\result}}.
Numeric value: \DTLdatumvalue{\result}.

\ExplSyntaxOn
\datatool_extract_timestamp:NN \result \l_tmpa_tl
Time-stamp: ~ \l_tmpa_tl
\ExplSyntaxOff
```

String: Fri 28th Feb 2025 3.45pm GMT. Data type: date-time. Numeric value: 2460735.15625.
Time-stamp: 2025-02-28T15:45:00+00:00

`\datatoolIMSetNumberChars`

Hook to switch to IM number group and decimal characters.

`\datatoolIMcurrencyfmt`

Used to format the IMP currency. This supports the currency symbol prefix.

`\DTLIMLocaleHook`

Hook to switch to IM settings. This may be added to the captions hook by datatool-base, depending on the settings. Otherwise it can be explicitly used to switch to this region.

2.10 Region “JE”

The `datatool-JE.ldf` file provides support for region “JE” (Jersey). This defines the “JEP” currency and sets it as the default currency. The default number group character is a comma, and the default decimal character is a dot.

If there’s no caption hook and you have multiple locales, you will need to set both the language and region hooks. For example, this document has:

```
\DTLenLocaleHook
\DTLJELocaleHook
```

Default Numeric Styles

```
\DTLdecimaltolocale{12345}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLdecimaltolocale{12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLdecimaltocurrency{-12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLmul\result{\result}{0.5}\result.
(Value: \DTLdatumvalue{\result}.)
```

12,345. (Value: 12345.)
12,345.678. (Value: 12345.678.)
-£12,345.68. (Value: -12345.678.)
-£6,172.84. (Value: -6172.839.)

Remember that date-time parsing needs to be enabled, if required.

Default Date-Time Styles

```
\DTLsetup{datetime={parse}}
\DTLparse\result{28/2/2025 15:45 GMT}
String: \result.
Data type: \DTLgetDataTypeName{\DTLdatumtype{\result}}.
Numeric value: \DTLdatumvalue{\result}.

\ExplSyntaxOn
\datatool_extract_timestamp:NN \result \l_tmpa_tl
Time-stamp: ~ \l_tmpa_tl
\ExplSyntaxOff
```

String: 28/2/2025 15:45 GMT. Data type: date-time. Numeric value: 2460735.15625.
Time-stamp: 2025-02-28T15:45:00+00:00

Options may be set with `\DTLsetLocaleOptions{`**`JE`**`}{`⟨*key*=*val list*⟩`}`. Available options are listed below.

`/JE/currency-symbol-prefix`=⟨*boolean*⟩ (initially false)

If true, shows the region prefix before the currency symbol if the numeric setting currency-symbol-style is not set to “iso”. The prefix font may be changed to smallcaps or smaller with `\DTLsetup{ numeric = { region-currency-prefix =` ⟨*value*⟩ `} }` where ⟨*value*⟩ is smallcaps or smaller.

```
\DTLsetup{numeric={region-currency-prefix=smallcaps}}
\DTLsetLocaleOptions{JE}{currency-symbol-prefix}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

JE£12,345.68

/JE/currency-symbol-position=⟨*value*⟩ (initially before)

Adjusts the formatting currency style. If **before**, the currency symbol is placed before the value. If **after**, the currency symbol is placed after the value.

```
\DTLsetup{numeric={currency-symbol-style=iso}}
\DTLsetLocaleOptions{JE}{currency-symbol-position=after}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

12,345.68 JEP

/JE/currency-symbol-sep=⟨*value*⟩ (initially none)

Sets the separator to use between the currency symbol (not code) and the value. Permitted values: **none** (no space), **thin-space** (a thin space), **space** (a normal space), or **nbsp** (a non-breaking space).

```
\DTLsetLocaleOptions{JE}{currency-symbol-sep=thin-space}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

£ 12,345.68

/JE/number-style=⟨*setting*⟩ (initially official)

Sets the number group and decimal characters. The value may be one of: **official** (comma number group and decimal point) or **unofficial** (thin space number group and decimal point). In the case of **unofficial**, a normal space may also be used when parsing.

```
\DTLsetup{numeric={auto-reformat}}
\DTLsetLocaleOptions{JE}{number-style=unofficial}
\DTLdecimaltocurrency{12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLparse{\result}{£12 345.678}\result.
(Value: \DTLdatumvalue{\result}.)
```

£12 345.68. (Value: 12345.678.)
£12 345.68. (Value: 12345.678.)

The remaining options relate to dates and times, which are still experimental. You need to enable date and time parsing with `\DTLsetup{ datetime = {parse} }`.

/JE/date-style=⟨*style*⟩ (initially dmyyyy)

Sets the current date style. The value may be one of: **dmyyyy** (day month year), **mdyyyy** (month day year), **yyyymd** (year month day), **dmyy** (day month 2-digit year), **mdyy** (month day 2-digit year), or **yymd** (2-digit year month day).

/JE/date-variant=⟨*style*⟩ (initially slash)

Sets the current numeric date separator. Allowed values: **slash** (/), **hyphen** (-), **dot** (.) or **dialect** (if the language supports it).

/JE/time-variant=⟨*style*⟩ (initially colon)

Sets the current numeric time separator. Allowed values: **colon** (:), **dot** (.) or **dialect** (if the language supports it), **dialect-colon** (if the language supports it), or **dialect-dot** (if the language supports it).

```
\DTLsetup{datetime={parse}}
\DTLsetLocaleOptions{JE}{
 date-style=dmyyyy,
 date-variant = dialect,
 time-variant = dot
}
\DTLparse\result{Fri 28th Feb 2025 3.45pm GMT}
String: \result.
Data type: \DTLgetDataTypeName{\DTLdatumtype{\result}}.
Numeric value: \DTLdatumvalue{\result}.

\ExplSyntaxOn
\datatool_extract_timestamp:NN \result \l_tmpa_tl
Time-stamp: ~ \l_tmpa_tl
\ExplSyntaxOff
```

String: Fri 28th Feb 2025 3.45pm GMT. Data type: date-time. Numeric value: 2460735.15625.
Time-stamp: 2025-02-28T15:45:00+00:00

\datatoolJESetNumberChars

Hook to switch to JE number group and decimal characters.

\datatoolJEcurrencyfmt

Used to format the JEP currency. This supports the currency symbol prefix.

\DTLJELocaleHook

Hook to switch to JE settings. This may be added to the captions hook by datatool-base, depending on the settings. Otherwise it can be explicitly used to switch to this region.

2.11 Region "NZ"

The `datatool-NZ.ldf` file provides support for region "NZ" (New Zealand). This defines the "NZD" currency and sets it as the default currency. The default number group character is a comma, and the default decimal character is a dot.

If there's no caption hook and you have multiple locales, you will need to set both the language and region hooks. For example, this document has:

```
\DTLenLocaleHook
\DTLNZLocaleHook
```

Default Numeric Styles

```
\DTLdecimaltolocale{12345}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLdecimaltolocale{12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLdecimaltocurrency{-12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLmul\result{\result}{0.5}\result.
(Value: \DTLdatumvalue{\result}.)
```

12,345. (Value: 12345.)
12,345.678. (Value: 12345.678.)
-$12,345.68. (Value: -12345.678.)
-$6,172.84. (Value: -6172.839.)

Remember that date-time parsing needs to be enabled, if required.

Default Date-Time Styles

```
\DTLsetup{datetime={parse}}
\DTLparse\result{28/2/2025 15:45 NZDT}
String: \result.
Data type: \DTLgetDataTypeName{\DTLdatumtype{\result}}.
Numeric value: \DTLdatumvalue{\result}.

\ExplSyntaxOn
\datatool_extract_timestamp:NN \result \l_tmpa_tl
Time-stamp: ~ \l_tmpa_tl
\ExplSyntaxOff
```

String: 28/2/2025 15:45 NZDT. Data type: date-time. Numeric value: 2460735.114583333.
Time-stamp: 2025-02-28T15:45:00+01:00

Options may be set with `\DTLsetLocaleOptions{`**`NZ`**`}{`⟨*key=val list*⟩`}`. Available options are listed below.

`/NZ/currency-symbol-prefix`=⟨*boolean*⟩ (initially false)

If true, shows the region prefix before the currency symbol if the numeric setting currency-symbol-style is not set to “iso”. The prefix font may be changed to smallcaps or smaller with `\DTLsetup{ numeric = { region-currency-prefix =` ⟨*value*⟩ `} }` where ⟨*value*⟩ is smallcaps or smaller.

```
\DTLsetup{numeric={region-currency-prefix=smallcaps}}
\DTLsetLocaleOptions{NZ}{currency-symbol-prefix}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

NZ$12,345.68

`/NZ/currency-symbol-position`=⟨*value*⟩ (initially before)

Adjusts the formatting currency style. If **`before`**, the currency symbol is placed before the value. If **`after`**, the currency symbol is placed after the value.

```
\DTLsetup{numeric={currency-symbol-style=iso}}
\DTLsetLocaleOptions{NZ}{currency-symbol-position=after}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

12,345.68 NZD

`/NZ/currency-symbol-sep`=⟨*value*⟩ (initially none)

Sets the separator to use between the currency symbol (not code) and the value. Permitted values: **`none`** (no space), **`thin-space`** (a thin space), **`space`** (a normal space), or **`nbsp`** (a non-breaking space).

```
\DTLsetLocaleOptions{NZ}{currency-symbol-sep=thin-space}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

$ 12,345.68

`/NZ/number-style`=⟨*setting*⟩ (initially official)

Sets the number group and decimal characters. The value may be one of: **`official`** (comma number group and decimal point), or **`unofficial`** (thin space number group and decimal point). In the case of **`unofficial`**, a normal space may also be used when parsing.

```
\DTLsetup{numeric={auto-reformat}}
\DTLsetLocaleOptions{NZ}{number-style=unofficial}
\DTLdecimaltocurrency{12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLparse{\result}{\$12 345.678}\result.
(Value: \DTLdatumvalue{\result}.)
```

$12 345.68. (Value: 12345.678.)
$12 345.68. (Value: 12345.678.)

The remaining options relate to dates and times, which are still experimental. You need to enable date and time parsing with `\DTLsetup{ datetime = {parse} }`.

`/NZ/date-style`=⟨*style*⟩ (initially mdyyyy)

Sets the current date style. The value may be one of: **`dmyyyy`** (day month year), **`mdyyyy`** (month day year), **`yyyymd`** (year month day), **`dmyy`** (day month 2-digit year), **`mdyy`** (month day 2-digit year), or **`yymd`** (2-digit year month day).

`/NZ/date-variant`=⟨*style*⟩ (initially slash)

Sets the current numeric date separator. Allowed values: **`slash`** (/), **`hyphen`** (-), **`dot`** (.) or **`dialect`** (if the language supports it).

`/NZ/time-variant`=⟨*style*⟩ (initially colon)

Sets the current numeric time separator. Allowed values: **`colon`** (:), **`dot`** (.) or **`dialect`** (if the language supports it), **`dialect-colon`** (if the language supports it), or **`dialect-dot`** (if the language supports it).

```
\DTLsetup{datetime={parse}}
\DTLsetLocaleOptions{NZ}{
 date-style=mdyyyy,
 date-variant = dialect,
 time-variant = dot
}
\DTLparse\result{February 28, 2025 3.45pm AEDT}
String: \result.
Data type: \DTLgetDataTypeName{\DTLdatumtype{\result}}.
Numeric value: \DTLdatumvalue{\result}.

\ExplSyntaxOn
\datatool_extract_timestamp:NN \result \l_tmpa_tl
Time-stamp: ~ \l_tmpa_tl
\ExplSyntaxOff
```

String: February 28, 2025 3.45pm AEDT. Data type: unset. Numeric value: .
Time-stamp:

\datatoolNZSetNumberChars

Hook to switch to NZ number group and decimal characters.

\datatoolNZcurrencyfmt

Used to format the NZD currency. This supports the currency symbol prefix.

\DTLNZLocaleHook

Hook to switch to NZ settings. This may be added to the captions hook by datatool-base, depending on the settings. Otherwise it can be explicitly used to switch to this region.

2.12 Region "US"

The `datatool-US.ldf` file provides support for region "US" (United States). This defines the "USD" currency and sets it as the default currency. The default number group character is a comma, and the default decimal character is a dot.

If there's no caption hook and you have multiple locales, you will need to set both the language and region hooks. For example, this document has:

```
\DTLenLocaleHook
\DTLUSLocaleHook
```

Default Styles

```
\DTLdecimaltolocale{12345}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLdecimaltolocale{12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLdecimaltocurrency{-12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLmul\result{\result}{0.5}\result.
(Value: \DTLdatumvalue{\result}.)
```

12,345. (Value: 12345.)
12,345.678. (Value: 12345.678.)
-$12,345.68. (Value: -12345.678.)
-$6,172.84. (Value: -6172.839.)

Remember that date-time parsing needs to be enabled, if required.

Default Date-Time Styles

```
\DTLsetup{datetime={parse}}
\DTLparse\result{2/28/2025 15:45 PST}
String: \result.
Data type: \DTLgetDataTypeName{\DTLdatumtype{\result}}.
Numeric value: \DTLdatumvalue{\result}.

\ExplSyntaxOn
\datatool_extract_timestamp:NN \result \l_tmpa_tl
Time-stamp: ~ \l_tmpa_tl
\ExplSyntaxOff
```

String: 2/28/2025 15:45 PST. Data type: date-time. Numeric value: 2460735.489583333.
Time-stamp: 2025-02-28T15:45:00-08:00

Options may be set with `\DTLsetLocaleOptions{`**`US`**`}{`⟨*key*=*val list*⟩`}`. Available options are listed below.

`/US/currency-symbol-prefix`=⟨*boolean*⟩ (initially false)

If true, shows the region prefix before the currency symbol if the numeric setting currency-symbol-style is not set to “iso”. The prefix font may be changed to smallcaps or smaller with `\DTLsetup{ numeric = { region-currency-prefix =` ⟨*value*⟩ `} }` where ⟨*value*⟩ is smallcaps or smaller.

```
\DTLsetup{numeric={region-currency-prefix=smallcaps}}
\DTLsetLocaleOptions{US}{currency-symbol-prefix}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

US$12,345.68

`/US/currency-symbol-position`=⟨*value*⟩ (initially before)

Adjusts the formatting currency style. If **`before`**, the currency symbol is placed before the value. If **`after`**, the currency symbol is placed after the value.

```
\DTLsetup{numeric={currency-symbol-style=iso}}
\DTLsetLocaleOptions{US}{currency-symbol-position=after}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

12,345.68 USD

`/US/currency-symbol-sep`=⟨*value*⟩ (initially none)

Sets the separator to use between the currency symbol (not code) and the value. Permitted values: **`none`** (no space), **`thin-space`** (a thin space), **`space`** (a normal space), or **`nbsp`** (a non-breaking space).

```
\DTLsetLocaleOptions{US}{currency-symbol-sep=thin-space}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

$ 12,345.68

`/US/number-style`=⟨*setting*⟩ (initially official)

Sets the number group and decimal characters. The value may be one of: **`official`** (comma number group and decimal point), or **`unofficial`** (thin space number group and decimal point). In the case of **`unofficial`**, a normal space may also be used when parsing.

```
\DTLsetup{numeric={auto-reformat}}
\DTLsetLocaleOptions{US}{number-style=unofficial}
\DTLdecimaltocurrency{12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLparse{\result}{\$12 345.678}\result.
(Value: \DTLdatumvalue{\result}.)
```

$12 345.68. (Value: 12345.678.)
$12 345.68. (Value: 12345.678.)

The remaining options relate to dates and times, which are still experimental. You need to enable date and time parsing with `\DTLsetup{ datetime = {parse} }`.

`/US/date-style`=⟨*style*⟩ (initially mdyyyy)

Sets the current date style. The value may be one of: **`dmyyyy`** (day month year), **`mdyyyy`** (month day year), **`yyyymd`** (year month day), **`dmyy`** (day month 2-digit year), **`mdyy`** (month day 2-digit year), or **`yymd`** (2-digit year month day).

`/US/date-variant`=⟨*style*⟩ (initially slash)

Sets the current numeric date separator. Allowed values: **`slash`** (/), **`hyphen`** (-), **`dot`** (.) or **`dialect`** (if the language supports it).

`/US/time-variant`=⟨*style*⟩ (initially colon)

Sets the current numeric time separator. Allowed values: **`colon`** (:), **`dot`** (.) or **`dialect`** (if the language supports it), **`dialect-colon`** (if the language supports it), or **`dialect-dot`** (if the language supports it).

```
\DTLsetup{datetime={parse}}
\DTLsetLocaleOptions{US}{
 date-style=mdyyyy,
 date-variant = dialect,
 time-variant = dot
}
\DTLparse\result{February 28, 2025 3.45pm EST}
String: \result.
Data type: \DTLgetDataTypeName{\DTLdatumtype{\result}}.
Numeric value: \DTLdatumvalue{\result}.

\ExplSyntaxOn
\datatool_extract_timestamp:NN \result \l_tmpa_tl
Time-stamp: ~ \l_tmpa_tl
\ExplSyntaxOff
```

String: February 28, 2025 3.45pm EST. Data type: date-time. Numeric value: 2460735.364583333.
Time-stamp: 2025-02-28T15:45:00-05:00

`\datatoolUSSetNumberChars`

Hook to switch to US number group and decimal characters.

`\datatoolUScurrencyfmt`

Used to format the USD currency. This supports the currency symbol prefix.

`\DTLUSLocaleHook`

Hook to switch to US settings. This may be added to the captions hook by datatool-base, depending on the settings. Otherwise it can be explicitly used to switch to this region.

2.13 Region “ZA”

The `datatool-ZA.ldf` file provides support for region “ZA” (South Africa). This supplies the currency (ZAR) but number formatting depends on the language, so this requires specific language & region files, which should be provided by the applicable language module. For example, datatool-english provides `datatool-en-ZA.ldf`.

If there’s no caption hook and you have multiple locales, you will need to set both the language and region hooks. For example, this document has:

```
\DTLenLocaleHook
\DTLenZALocaleHook
\DTLZALocaleHook
```

Default Styles (with English)

```
\DTLdecimaltolocale{12345}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLdecimaltolocale{12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLdecimaltocurrency{-12345.678}{\result}\result.
(Value: \DTLdatumvalue{\result}.)

\DTLmul\result{\result}{0,5}\result.
(Value: \DTLdatumvalue{\result}.)
```

12.345. (Value: 12345.)
12.345,678. (Value: 12345.678.)
-R12.345,68. (Value: -12345.678.)
-R6.172,84. (Value: -6172.839.)

Remember that date-time parsing needs to be enabled, if required.

Default Date-Time Styles

```
\DTLsetup{datetime={parse}}
\DTLparse\result{2025-2-28 15:45 SAST}
String: \result.
Data type: \DTLgetDataTypeName{\DTLdatumtype{\result}}.
Numeric value: \DTLdatumvalue{\result}.

\ExplSyntaxOn
\datatool_extract_timestamp:NN \result \l_tmpa_tl
Time-stamp: ~ \l_tmpa_tl
\ExplSyntaxOff
```

String: 2025-2-28 15:45 SAST. Data type: date-time. Numeric value: 2460709.072916667.
Time-stamp: 2025-02-02T15:45:00+02:00

Options may be set with `\DTLsetLocaleOptions{`**`ZA`**`}{`⟨*key*=*val list*⟩`}`. Available options are listed below.

`/ZA/currency-symbol-position`=⟨*value*⟩ (initially before)

Adjusts the formatting currency style. If **`before`**, the currency symbol is placed before the value. If **`after`**, the currency symbol is placed after the value.

```
\DTLsetup{numeric={currency-symbol-style=iso}}
\DTLsetLocaleOptions{ZA}{currency-symbol-position=after}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

12.345,68 ZAR

/ZA/currency-symbol-sep=⟨*value*⟩ (initially none)

Sets the separator to use between the currency symbol (not code) and the value. Permitted values: **none** (no space), **thin-space** (a thin space), **space** (a normal space), or **nbsp** (a non-breaking space).

```
\DTLsetLocaleOptions{ZA}{currency-symbol-sep=thin-space}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

R 12.345,68

/ZA/number-style=⟨*style*⟩ (initially official)

Sets the number group and decimal characters. The value may be one of: **official** (dot number group and decimal comma) or **dialect** (attempt to use the number style for the module ⟨*lang*⟩-ZA, if it's defined, where ⟨*lang*⟩ is the current language tag).

```
\DTLsetLocaleOptions{ZA}{number-style=official}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

R12.345,68

Or if `datatool-en-ZA.ldf` has been loaded and the current language is English:

```
\DTLsetLocaleOptions{ZA}{number-style=dialect}
\DTLdecimaltocurrency{12345.678}{\result}\result
```

R12 345.68

The remaining options relate to dates and times, which are still experimental. You need to enable date and time parsing with `\DTLsetup{ datetime = {parse} }`.

/ZA/date-style=⟨*style*⟩ (initially yyyymd)

Sets the current date style. The value may be one of: **dmyyyy** (day month year), **mdyyyy** (month day year), **yyyymd** (year month day), **dmyy** (day month 2-digit year), **mdyy** (month day 2-digit year), or **yymd** (2-digit year month day).

/ZA/date-variant=⟨*style*⟩ (initially hyphen)

Sets the current numeric date separator. Allowed values: **slash** (/), **hyphen** (-), **dot** (.) or **dialect** (if the language supports it).

/ZA/time-variant=⟨*style*⟩ (initially colon)

Sets the current numeric time separator. Allowed values: **colon** (:), **dot** (.) or **dialect** (if the language supports it), **dialect-colon** (if the language supports it), or **dialect-dot** (if the language supports it).

```
\DTLsetup{datetime={parse}}
\DTLsetLocaleOptions{ZA}{
 date-style=mdyyyy,
 date-variant = dialect,
 time-variant = dot
}
\DTLparse\result{June 1, 2025 3.45pm SAST}
String: \result.
Data type: \DTLgetDataTypeName{\DTLdatumtype{\result}}.
Numeric value: \DTLdatumvalue{\result}.

\ExplSyntaxOn
\datatool_extract_timestamp:NN \result \l_tmpa_tl
Time-stamp: ~ \l_tmpa_tl
\ExplSyntaxOff
```

String: June 1, 2025 3.45pm SAST. Data type: date-time. Numeric value: 2460828.072916667. Time-stamp: 2025-06-01T15:45:00+02:00

\datatoolZASetNumberChars

Hook to switch to ZA number group and decimal characters.

\datatoolZAcurrencyfmt

Used to format the ZAR currency.

\DTLZALocaleHook

Hook to switch to ZA settings. This may be added to the captions hook by datatool-base, depending on the settings. Otherwise it can be explicitly used to switch to this region.

3 The Code

3.1 datatool-AU.ldf

Support for region AU.

```
\TrackLangProvidesResource{AU}[2025/03/01 v1.0]
```

Switch on LaTeX3 syntax:

```
\ExplSyntaxOn
```

3.1.1 Numbers and Currency

Set the number group and decimal symbols for this region.

```
\cs_new:Nn \datatool_AU_set_numberchars_official:
 {
  \datatool_set_numberchars:nn { , } { . }
 }
```

Unofficial style.

```
\cs_new:Nn \datatool_AU_set_numberchars_unofficial:
 {
  \datatool_set_thinspace_group_decimal_char:n { . }
 }
```

Hook to set the number group and decimal characters for the region:

```
\newcommand \datatoolAUSetNumberChars
 {
   \bool_if:NT \l_datatool_region_set_numberchars_bool
    {
      \datatool_AU_set_numberchars_official:
    }
 }
```

How to format the position of the currency symbol in relation to the value.

```
\cs_new:Nn \datatool_AU_currency_position:nn
 {
   \dtlcurrprefixfmt { #1 } { #2 }
 }
```

Separator between currency symbol and value.

```
\tl_new:N \l_datatool_AU_sym_sep_tl
```

Set the currency format for this region.

```
\newcommand \datatoolAUcurrencyfmt [ 2 ]
 {
   \datatool_AU_currency_position:nn
```

```
    {
      \datatoolAUsymbolprefix { AU }
      #1
    }
    { #2 }
 }
```

Prefix for symbol, if required.

```
\newcommand \datatoolAUsymbolprefix [ 1 ] { }
```

Define the currency symbols for this region.

```
\datatool_def_currency:nnnV
 { \datatoolAUcurrencyfmt }
 { AUD }
 { \$ }
 \c_dollar_str
```

Register the currency code with this region:

```
\datatool_register_regional_currency_code:nn { AU } { AUD }
```

Provide a command to set the currency for this region (for use with any hook used when the locale changes). NB this should do nothing with `\DTLsetup{region-currency=false}`

```
\newcommand \datatoolAUSetCurrency
 {
   \bool_if:NT \l_datatool_region_set_currency_bool
    {
      \DTLsetdefaultcurrency { AUD }
```

Number of digits that `\DTLdecimaltocurrency` should round to:

```
      \renewcommand \DTLCurrentLocaleCurrencyDP { 2 }
```

Separator between symbol and value:

```
      \renewcommand \dtlcurrfmtsymsep { \l_datatool_AU_sym_sep_tl }
    }
 }
```

3.1.2 Date and Time Parsing

This defaults to day/month/year but may be changed with `\DTLsetLocaleOptions`. For example:

```
\DTLsetLocaleOptions{AU}{date-style=dmyyyy, date-variant=slash}
```

An appropriate language file will need to also be installed to parse dates containing month names or day of week names.

Provide a way to configure parsing style.

```
\tl_new:N \l__datatool_AU_datevariant_tl
\tl_set:Nn \l__datatool_AU_datevariant_tl { slash }
```

NB These token lists are used to form command names. The following is not a format string.

```
\tl_new:N \l__datatool_AU_datestyle_tl
```

```
\tl_set:Nn \l__datatool_AU_datestyle_tl { ddmmyyyy }
\tl_new:N \l__datatool_AU_timevariant_tl
\tl_set:Nn \l__datatool_AU_timevariant_tl { colon }
```

Each parsing command defined below has final {⟨*true*⟩} and {⟨*false*⟩} arguments (TF). These are used if parsing was successful (true) or if parsing failed (false). The internal commands used by datatool-base have no need for solo branches (only T or only F) so these commands are simply defined with `\cs_new:Nn` not as conditionals.

Time Stamp Parsing

Use command

`\datatool_`⟨*date-style*⟩`_hhmmss_tz_parse_timestamp:nnNnTF`

with date regular expression

`\c_datatool_`⟨*date-variant*⟩`_`⟨*date-style*⟩`_date_regex`

and time regular expression

`\c_datatool_`⟨*time-variant*⟩`_hhmmss_time_regex`

```
\cs_new:Nn \datatool_AU_parse_timestamp:NnTF
 {
   \cs_if_exist:cTF
     {
       datatool_
       \l__datatool_AU_datestyle_tl
       _hhmmss_tz_parse_timestamp:ccNnTF
     }
    {
      \cs_if_exist:cTF
       {
          c_datatool_
          \l__datatool_AU_datevariant_tl
          _
          \l__datatool_AU_datestyle_tl
          _date_regex
       }
       {
         \cs_if_exist:cTF
          {
            c_datatool_
            \l__datatool_AU_timevariant_tl
            _hhmmss_time_regex
          }
          {
            \use:c
         {
           datatool_
          \l__datatool_AU_datestyle_tl
          _hhmmss_tz_parse_timestamp:ccNnTF
         }
         {
           c_datatool_
           \l__datatool_AU_datevariant_tl
           _
           \l__datatool_AU_datestyle_tl
           _date_regex
         }
         {
           c_datatool_
           \l__datatool_AU_timevariant_tl
           _hhmmss_time_regex
         }
          #1 { #2 } { #3 } { #4 }
      }
      {
        \datatool_warn_check_head_language_empty:Vnnn
         \l__datatool_AU_timevariant_tl
         { datatool-AU }
         {
           No ~ language ~ support ~ for ~ time ~ variant ~
           ` \exp_args:Ne \tl_tail:n { \l__datatool_AU_timevariant_tl } '
         }
         {
          No ~ support ~ for ~ time ~ variant ~
          ` \l__datatool_AU_timevariant_tl '
         }
        #4
      }
   }
   {
     \datatool_warn_check_head_language_empty:Vnnn
       \l__datatool_AU_datevariant_tl
       { datatool-AU }
       {
         No ~ language ~ support ~ for ~ date ~ style ~
          ` \l__datatool_AU_datestyle_tl '
       }
       {
         No ~ support ~ for ~ date ~ style ~
         ` \l__datatool_AU_datestyle_tl ' ~ with ~
         variant ~
         ` \l__datatool_AU_datevariant_tl '
       }
     #4
   }
}
{
  \datatool_locale_warn:nn { datatool-AU }
```

```
        {
          No ~ support ~ for ~ date ~ style ~
          ` \l__datatool_AU_datestyle_tl '
        }
       #4
     }
  }
```

Date Parsing

Use command `\datatool_`⟨*style*⟩`_parse_date:NNnTF` with regular expression

`\c_datatool_`⟨*variant*⟩`_anchored_`⟨*style*⟩`_date_regex`

```
 \cs_new:Nn \datatool_AU_parse_date:NnTF
  {
    \cs_if_exist:cTF
      {
        datatool_
        \l__datatool_AU_datestyle_tl
        _parse_date:NNnTF
      }
     {
       \cs_if_exist:cTF
        {
          c_datatool_
          \l__datatool_AU_datevariant_tl
          _anchored_
          \l__datatool_AU_datestyle_tl
          _date_regex
        }
        {
          \exp_args:cc
          {
            datatool_
            \l__datatool_AU_datestyle_tl
            _parse_date:NNnTF
          }
          {
            c_datatool_
            \l__datatool_AU_datevariant_tl
            _anchored_
            \l__datatool_AU_datestyle_tl
            _date_regex
          }
           #1 { #2 } { #3 } { #4 }
        }
        {
          \datatool_warn_check_head_language_empty:Vnnn
           \l__datatool_AU_datevariant_tl
           { datatool-AU }
           {
             No ~ language ~ support ~ for ~ date ~ style ~
              ` \l__datatool_AU_datestyle_tl '
           }
           {
              No ~ support ~ for ~ date ~ style ~
              ` \l__datatool_AU_datestyle_tl ' ~ with ~
              variant ~
              ` \l__datatool_AU_datevariant_tl '
           }
          #4
        }
     }
     {
       \datatool_locale_warn:nn { datatool-AU }
        {
           No ~ support ~ for ~ date ~ style ~
           ` \l__datatool_AU_datestyle_tl '
        }
       #4
     }
  }
```

Time Parsing

Use command `\datatool_`⟨*style*⟩`_parse_time:NNnTF` with regular expression

`\c_datatool_`⟨*variant*⟩`_anchored_`⟨*style*⟩`_time_regex`

```
\cs_new:Nn \datatool_AU_parse_time:NnTF
 {
   \cs_if_exist:cTF
    {
      c_datatool_
      \l__datatool_AU_timevariant_tl
      _anchored_hhmmss_time_regex
    }
    {
      \datatool_hhmmss_parse_time:cNnTF
       {
        c_datatool_
        \l__datatool_AU_timevariant_tl
        _anchored_hhmmss_time_regex
       }
        #1 { #2 } { #3 } { #4 }
    }
    {
      \datatool_warn_check_head_language_empty:Vnnn
        \l__datatool_AU_timevariant_tl
        { datatool-AU }
        {
           No ~ language ~ support ~ for ~ time ~ variant ~
           ` \exp_args:Ne \tl_tail:n { \l__datatool_AU_timevariant_tl } '
        }
        {
           No ~ support ~ for ~ time ~ variant ~
           ` \l__datatool_AU_timevariant_tl '
        }
      #4
    }
 }
```

Time Zone Mappings

Define time zone mapping command for this region:

```
\cs_new:Nn \datatool_AU_get_timezone_map:n
  {
    \datatool_region_get_timezone_map:n { AU / #1 }
  }
```

Define time zone IDs for this region (one-way map from ID to offset):

```
\datatool_region_set_timezone_map:nn { AU / ACST } { +09:30 }
\datatool_region_set_timezone_map:nn { AU / ACDT } { +10:30 }
\datatool_region_set_timezone_map:nn { AU / ACWST } { +08:45 }
\datatool_region_set_timezone_map:nn { AU / ACWDT } { +09:00 }
\datatool_region_set_timezone_map:nn { AU / AEST } { +10:00 }
\datatool_region_set_timezone_map:nn { AU / AEDT } { +11:00 }
\datatool_region_set_timezone_map:nn { AU / CXT } { +07:00 }
\datatool_region_set_timezone_map:nn { AU / NFT } { +11:00 }
\datatool_region_set_timezone_map:nn { AU / NFDT } { +12:00 }
\datatool_region_set_timezone_map:nn { AU / LHST } { +10:30 }
\datatool_region_set_timezone_map:nn { AU / LHDT } { +11:00 }
\datatool_region_set_timezone_map:nn { AU / CCT } { +06:30 }
```

3.1.3 Options

Define options for this region:

```
\datatool_locale_define_keys:nn { AU }
  {
    number-style .choices:nn =
     { official, unofficial }
     {
       \exp_args:NNe \renewcommand
         \datatoolAUSetNumberChars
          {
            \exp_not:N \bool_if:NT
             \exp_not:N \l_datatool_region_set_numberchars_bool
              {
                \exp_not:c { datatool_AU_set_numberchars_ \l_keys_choice_tl : }
              }
          }
       \tl_if_eq:NnT \l_datatool_current_region_tl { AU }
```

```
        {
          \datatoolAUSetNumberChars
        }
   } ,
```

Currency symbol before or after value:

```
currency-symbol-position .choice: ,
currency-symbol-position / before .code:n =
 {
   \cs_set_eq:NN \datatool_AU_currency_position:nn \dtlcurrprefixfmt
 } ,
currency-symbol-position / after .code:n =
 {
   \cs_set_eq:NN \datatool_AU_currency_position:nn \dtlcurrsuffixfmt
 } ,
```

Should the currency symbol be prefixed with the region code:

```
currency-symbol-prefix .choice: ,
currency-symbol-prefix / false .code:n =
 {
   \cs_set_eq:NN \datatoolAUsymbolprefix \use_none:n
 } ,
currency-symbol-prefix / true .code:n =
 {
   \cs_set_eq:NN
    \datatoolAUsymbolprefix
    \datatool_currency_symbol_region_prefix:n
 } ,
currency-symbol-prefix .default:n = { true } ,
```

Separator between currency symbol (not code) and value:

```
currency-symbol-sep .choice: ,
currency-symbol-sep / none .code:n =
 {
   \tl_clear:N \l_datatool_AU_sym_sep_tl
 } ,
currency-symbol-sep / thin-space .code:n =
 {
   \tl_set:Nn \l_datatool_AU_sym_sep_tl { \, }
 } ,
currency-symbol-sep / space .code:n =
 {
   \tl_set:Nn \l_datatool_AU_sym_sep_tl { ~ }
 } ,
currency-symbol-sep / nbsp .code:n =
 {
   \tl_set:Nn \l_datatool_AU_sym_sep_tl { \nobreakspace }
 } ,
currency-symbol-sep . initial:n = { none } ,
```

Date and time styles:

```
date-style .choice: ,
date-style / dmyyyy .code: n =
 {
   \tl_set:Nn \l__datatool_AU_datestyle_tl { ddmmyyyy }
 } ,
date-style / mdyyyy .code: n =
 {
   \tl_set:Nn \l__datatool_AU_datestyle_tl { mmddyyyy }
 } ,
date-style / yyyymd .code: n =
 {
   \tl_set:Nn \l__datatool_AU_datestyle_tl { yyyymmdd }
 } ,
date-style / dmyy .code: n =
 {
   \tl_set:Nn \l__datatool_AU_datestyle_tl { ddmmyy }
 } ,
date-style / mdyy .code: n =
 {
   \tl_set:Nn \l__datatool_AU_datestyle_tl { mmddyy }
 } ,
date-style / yymd .code: n =
 {
   \tl_set:Nn \l__datatool_AU_datestyle_tl { yymmdd }
 } ,
date-variant .choice: ,
date-variant / slash .code:n =
 {
   \tl_set:Nn \l__datatool_AU_datevariant_tl { slash }
 } ,
date-variant / hyphen .code:n =
 {
   \tl_set:Nn \l__datatool_AU_datevariant_tl { hyphen }
 } ,
date-variant / dot .code:n =
 {
   \tl_set:Nn \l__datatool_AU_datevariant_tl { dot }
 } ,
date-variant / dialect .code:n =
 {
   \tl_set:Nn \l__datatool_AU_datevariant_tl
     { \l_datatool_current_language_tl }
 } ,
time-variant .choice: ,
time-variant / colon .code:n =
 {
   \tl_set:Nn \l__datatool_AU_timevariant_tl { colon }
 } ,
time-variant / dot .code:n =
 {
   \tl_set:Nn \l__datatool_AU_timevariant_tl { dot }
 } ,
time-variant / dialect .code:n =
 {
   \tl_if_eq:NnTF \l__datatool_AU_timevariant_tl { dot }
    {
         \tl_set:Nn \l__datatool_AU_timevariant_tl
          { \l_datatool_current_language_tl _dot }
       }
       {
         \tl_if_in:NnF
           \l__datatool_AU_timevariant_tl
            { \l_datatool_current_language_tl }
          {
            \tl_set:Nn \l__datatool_AU_timevariant_tl
             { \l_datatool_current_language_tl _colon }
          }
       }
    } ,
   time-variant / dialect-colon .code:n =
    {
      \tl_set:Nn \l__datatool_AU_timevariant_tl
        { \l_datatool_current_language_tl _colon }
    } ,
   time-variant / dialect-dot .code:n =
    {
      \tl_set:Nn \l__datatool_AU_timevariant_tl
        { \l_datatool_current_language_tl _dot }
    } ,
 }
```

3.1.4 Hooks

Command to update temporal parsing commands for this region:

```
\newcommand \datatoolAUSetTemporalParsers
 {
   \renewcommand \DTLCurrentLocaleParseTimeStamp
    { \datatool_AU_parse_timestamp:NnTF }
   \renewcommand \DTLCurrentLocaleParseDate
    { \datatool_AU_parse_date:NnTF }
   \renewcommand \DTLCurrentLocaleParseTime
    { \datatool_AU_parse_time:NnTF }
   \let
    \DTLCurrentLocaleGetTimeZoneMap
    \datatool_AU_get_timezone_map:n
 }
```

Set temporal formatting commands for this region. Currently this just resets to the default, but may change in future. Note that the defaults test if the applicable datetime2 command is available and will fallback on ISO if not defined. Bear in mind that the default style for datetime2 is `iso` so there won't be a noticeable difference unless the datetime2 regional setting is on.

```
\newcommand \datatoolAUSetTemporalFormatters
 {
   \let
    \DTLCurrentLocaleFormatDate
    \datatool_default_date_fmt:nnnn
   \let
    \DTLCurrentLocaleFormatTime
    \datatool_default_time_fmt:nnn
   \let
    \DTLCurrentLocaleFormatTimeZone
    \datatool_default_timezone_fmt:nn
   \let
    \DTLCurrentLocaleFormatTimeStampNoZone
    \datatool_default_timestamp_fmt:nnnnnnn
   \let
    \DTLCurrentLocaleFormatTimeStampWithZone
    \datatool_default_timestamp_fmt:nnnnnnnnn
   \renewcommand \DTLCurrentLocaleTimeStampFmtSep { ~ }
 }
```

Command to update currency and temporal parsing commands for this region:

```
\newcommand \DTLAULocaleHook
 {
  \datatoolAUSetNumberChars
  \datatoolAUSetCurrency
  \datatoolAUSetTemporalParsers
  \datatoolAUSetTemporalFormatters
```

Allow language files to reference the region:

```
  \tl_set:Nn \l_datatool_current_region_tl { AU }
 }
```

Finished with LaTeX3 syntax.

```
\ExplSyntaxOff
```

Note that the hook is added to the captions by datatool-base not by the region file.

3.2 datatool-BE.ldf

Support for region BE.

```
\TrackLangProvidesResource{BE}[2025/03/01 v1.0]
```

Switch on LaTeX3 syntax:

```
\ExplSyntaxOn
```

3.2.1 Numbers and Currency

Default style is to use dot separator and decimal dot. If a French language module is developed, it will need to provide a file called `datatool-fr-BE.ldf` which should define a command called `\datatoolfrBESetNumberChars` to set the number group separator to a space. (See `datatool-en-ZA.ldf` provided with datatool-english as an example.)

```
\cs_new:Nn \datatool_BE_set_numberchars_default:
 {
   \DTLsetnumberchars { . } { , }
 }
```

Thin-space style.

```
\cs_new:Nn \datatool_BE_set_numberchars_thinspace:
 {
  \datatool_set_thinspace_group_decimal_char:n { . }
 }
```

Check if the dialect has provided a style. If so, use that, otherwise use the default.

```
\cs_new:Nn \datatool_BE_set_numberchars_dialect:
 {
   \tl_if_empty:NTF \l_datatool_current_language_tl
    {
      \datatool_BE_set_numberchars_default:
    }
    {
      \cs_if_exist_use:cF
       { datatool \l_datatool_current_language_tl BESetNumberChars }
       {
         \datatool_BE_set_numberchars_default:
       }
    }
 }
```

Hook to set the number group and decimal characters for the region:

```
\newcommand \datatoolBESetNumberChars
 {
   \bool_if:NT \l_datatool_region_set_numberchars_bool
    {
      \datatool_BE_set_numberchars_dialect:
    }
 }
```

The EUR currency is already defined by datatool-base, but we need to register the currency code with this region:

```
\datatool_register_regional_currency_code:nn { BE } { EUR }
```

How to format the position of the currency symbol in relation to the value.

```
\cs_new:Nn \datatool_BE_currency_position:nn
 {
   \dtlcurrsuffixfmt { #1 } { #2 }
 }
```

Separator between currency symbol and value.

```
\tl_new:N \l_datatool_BE_sym_sep_tl
```

Provide a command to set the currency for this region (for use with any hook used when the locale changes).

```
\newcommand \datatoolBESetCurrency
 {
   \bool_if:NT \l_datatool_region_set_currency_bool
    {
     \DTLsetdefaultcurrency { EUR }
```

Number of digits that `\DTLdecimaltocurrency` should round to:

```
     \renewcommand \DTLCurrentLocaleCurrencyDP { 2 }
```

Update EUR format to reflect current region style:

```
      \renewcommand \DTLdefaultEURcurrencyfmt
        { \datatool_BE_currency_position:nn }
```

Separator between symbol and value:

```
      \renewcommand \dtlcurrfmtsymsep { \l_datatool_BE_sym_sep_tl }
     }
  }
```

3.2.2 Date and Time Parsing

Date parsing is more complicated as there seems to be a confusing mixture of ISO, day/-month/year and month/day/year formats. The default is year-month-day but may be changed with `\DTLsetLocaleOptions`. For example:

```
\DTLsetLocaleOptions{BE}{date-style=dmyyyy, date-variant=slash}
```

An appropriate language file will need to also be installed to parse dates containing month names or day of week names.

```
\tl_new:N \l__datatool_BE_datevariant_tl
\tl_set:Nn \l__datatool_BE_datevariant_tl { dot }
```

NB These token lists are used to form command names. The following is not a format string.

```
\tl_new:N \l__datatool_BE_datestyle_tl
\tl_set:Nn \l__datatool_BE_datestyle_tl { ddmmyyyy }
\tl_new:N \l__datatool_BE_timevariant_tl
\tl_set:Nn \l__datatool_BE_timevariant_tl { colon }
```

Each parsing command defined below has final {⟨*true*⟩} and {⟨*false*⟩} arguments (TF). These are used if parsing was successful (true) or if parsing failed (false). The internal commands used by datatool-base have no need for solo branches (only T or only F) so these commands are simply defined with `\cs_new:Nn` not as conditionals.

Time Stamp Parsing

Use command

`\datatool_`⟨*date-style*⟩`_hhmmss_tz_parse_timestamp:nnNnTF`

with date regular expression

`\c_datatool_`⟨*date-variant*⟩`_`⟨*date-style*⟩`_date_regex`

and time regular expression

`\c_datatool_⟨time-variant⟩_hhmmss_time_regex`

```
\cs_new:Nn \datatool_BE_parse_timestamp:NnTF
 {
   \cs_if_exist:cTF
     {
       datatool_
       \l__datatool_BE_datestyle_tl
       _hhmmss_tz_parse_timestamp:ccNnTF
     }
    {
      \cs_if_exist:cTF
       {
          c_datatool_
          \l__datatool_BE_datevariant_tl
          _
          \l__datatool_BE_datestyle_tl
          _date_regex
       }
       {
         \cs_if_exist:cTF
          {
            c_datatool_
            \l__datatool_BE_timevariant_tl
            _hhmmss_time_regex
          }
          {
            \use:c
             {
               datatool_
              \l__datatool_BE_datestyle_tl
              _hhmmss_tz_parse_timestamp:ccNnTF
             }
             {
               c_datatool_
               \l__datatool_BE_datevariant_tl
               _
               \l__datatool_BE_datestyle_tl
               _date_regex
             }
             {
               c_datatool_
               \l__datatool_BE_timevariant_tl
               _hhmmss_time_regex
             }
              #1 { #2 } { #3 } { #4 }
          }
          {
            \datatool_locale_warn:nn { datatool-BE }
             {
              No ~ support ~ for ~ time ~ variant ~
              ` \l__datatool_BE_timevariant_tl '
             }
            #4
          }
       }
       {
         \datatool_locale_warn:nn { datatool-BE }
          {
           No ~ support ~ for ~ date ~ style ~
           ` \l__datatool_BE_datestyle_tl ' ~ with ~
           variant ~
           ` \l__datatool_BE_datevariant_tl '
          }
         #4
       }
    }
    {
      \datatool_locale_warn:nn { datatool-BE }
       {
          No ~ support ~ for ~ date ~ style ~
          ` \l__datatool_BE_datestyle_tl '
       }
      #4
    }
 }
```

Date Parsing

Use command \datatool_⟨*style*⟩_parse_date:NNnTF with regular expression

\c_datatool_⟨*variant*⟩_anchored_⟨*style*⟩_date_regex

```
\cs_new:Nn \datatool_BE_parse_date:NnTF
 {
   \cs_if_exist:cTF
     {
       datatool_
       \l__datatool_BE_datestyle_tl
       _parse_date:NNnTF
     }
    {
      \cs_if_exist:cTF
       {
         c_datatool_
         \l__datatool_BE_datevariant_tl
         _anchored_
         \l__datatool_BE_datestyle_tl
         _date_regex
       }
       {
         \exp_args:cc
         {
           datatool_
           \l__datatool_BE_datestyle_tl
           _parse_date:NNnTF
         }
         {
           c_datatool_
           \l__datatool_BE_datevariant_tl
           _anchored_
           \l__datatool_BE_datestyle_tl
           _date_regex
         }
          #1 { #2 } { #3 } { #4 }
       }
       {
         \datatool_locale_warn:nn { datatool-BE }
          {
           No ~ support ~ for ~ date ~ style ~
           ` \l__datatool_BE_datestyle_tl ' ~ with ~
           variant ~
           ` \l__datatool_BE_datevariant_tl '
          }
         #4
       }
    }
    {
      \datatool_locale_warn:nn { datatool-BE }
       {
          No ~ support ~ for ~ date ~ style ~
          ` \l__datatool_BE_datestyle_tl '
       }
      #4
    }
 }
```

Time Parsing

Use command `\datatool_⟨style⟩_parse_time:NNnTF` with regular expression

`\c_datatool_⟨variant⟩_anchored_⟨style⟩_time_regex`

```
\cs_new:Nn \datatool_BE_parse_time:NnTF
 {
   \cs_if_exist:cTF
    {
      c_datatool_
      \l__datatool_BE_timevariant_tl
      _anchored_hhmmss_time_regex
    }
    {
      \datatool_hhmmss_parse_time:cNnTF
       {
        c_datatool_
        \l__datatool_BE_timevariant_tl
        _anchored_hhmmss_time_regex
       }
        #1 { #2 } { #3 } { #4 }
    }
    {
      \datatool_locale_warn:nn { datatool-BE }
       {
        No ~ support ~ for ~ time ~ variant ~
        ` \l__datatool_BE_timevariant_tl '
       }
      #4
    }
 }
```

Time Zone Mapping

Define time zone mapping command for this region:

```
\cs_new:Nn \datatool_BE_get_timezone_map:n
 {
   \datatool_region_get_timezone_map:n { BE / #1 }
 }
```

Define time zone IDs for this region (one-way map from ID to offset):

```
\datatool_region_set_timezone_map:nn { BE / WEST } { +01:00 }
\datatool_region_set_timezone_map:nn { BE / WET } { +00:00 }
\datatool_region_set_timezone_map:nn { BE / EEST } { +03:00 }
\datatool_region_set_timezone_map:nn { BE / EET } { +02:00 }
\datatool_region_set_timezone_map:nn { BE / CEST } { +02:00 }
\datatool_region_set_timezone_map:nn { BE / CET } { +01:00 }
\datatool_region_set_timezone_map:nn { BE / GMT } { +00:00 }
```

3.2.3 Options

Define options for this region:

```
\datatool_locale_define_keys:nn { BE }
 {
   number-style .choices:nn =
    { default , dialect , thinspace }
    {
      \exp_args:NNe \renewcommand
        \datatoolBESetNumberChars
         {
           \exp_not:N \bool_if:NT
            \exp_not:N \l_datatool_region_set_numberchars_bool
             {
               \exp_not:c { datatool_BE_set_numberchars_ \l_keys_choice_tl : }
             }
         }
      \tl_if_eq:NnT \l_datatool_current_region_tl { BE }
    {
      \datatoolBESetNumberChars
    }
 } ,
```

Currency symbol before or after value:

```
currency-symbol-position .choice: ,
currency-symbol-position / before .code:n =
 {
   \cs_set_eq:NN \datatool_BE_currency_position:nn \dtlcurrprefixfmt
 } ,
currency-symbol-position / after .code:n =
 {
   \cs_set_eq:NN \datatool_BE_currency_position:nn \dtlcurrsuffixfmt
 } ,
```

Separator between currency symbol (not code) and value:

```
currency-symbol-sep .choice: ,
currency-symbol-sep / none .code:n =
 {
   \tl_clear:N \l_datatool_BE_sym_sep_tl
 } ,
currency-symbol-sep / thin-space .code:n =
 {
   \tl_set:Nn \l_datatool_BE_sym_sep_tl { \, }
 } ,
currency-symbol-sep / space .code:n =
 {
   \tl_set:Nn \l_datatool_BE_sym_sep_tl { ~ }
 } ,
currency-symbol-sep / nbsp .code:n =
 {
   \tl_set:Nn \l_datatool_BE_sym_sep_tl { \nobreakspace }
 } ,
currency-symbol-sep .initial:n = { none } ,
```

Date and time styles:

```
date-style .choice: ,
date-style / dmyyyy .code: n =
 {
   \tl_set:Nn \l__datatool_BE_datestyle_tl { ddmmyyyy }
 } ,
date-style / mdyyyy .code: n =
 {
   \tl_set:Nn \l__datatool_BE_datestyle_tl { mmddyyyy }
 } ,
date-style / yyyymd .code: n =
 {
   \tl_set:Nn \l__datatool_BE_datestyle_tl { yyyymmdd }
 } ,
date-style / dmyy .code: n =
 {
   \tl_set:Nn \l__datatool_BE_datestyle_tl { ddmmyy }
 } ,
date-style / mdyy .code: n =
 {
   \tl_set:Nn \l__datatool_BE_datestyle_tl { mmddyy }
 } ,
date-style / yymd .code: n =
 {
   \tl_set:Nn \l__datatool_BE_datestyle_tl { yymmdd }
 } ,
date-variant .choice: ,
date-variant / slash .code:n =
 {
   \tl_set:Nn \l__datatool_BE_datevariant_tl { slash }
 } ,
date-variant / hyphen .code:n =
 {
   \tl_set:Nn \l__datatool_BE_datevariant_tl { hyphen }
 } ,
date-variant / dot .code:n =
 {
   \tl_set:Nn \l__datatool_BE_datevariant_tl { dot }
 } ,
date-variant / dialect .code:n =
 {
   \tl_set:Nn \l__datatool_BE_datevariant_tl
     { \l_datatool_current_language_tl }
 } ,
time-variant .choice: ,
time-variant / colon .code:n =
 {
   \tl_set:Nn \l__datatool_BE_timevariant_tl { colon }
 } ,
time-variant / dot .code:n =
 {
   \tl_set:Nn \l__datatool_BE_timevariant_tl { dot }
 } ,
time-variant / dialect .code:n =
 {
   \tl_if_eq:NnTF \l__datatool_BE_timevariant_tl { dot }
    {
      \tl_set:Nn \l__datatool_BE_timevariant_tl
       { \l_datatool_current_language_tl _dot }
    }
    {
      \tl_if_in:NnF
        \l__datatool_BE_timevariant_tl
        { \l_datatool_current_language_tl }
       {
         \tl_set:Nn \l__datatool_BE_timevariant_tl
          { \l_datatool_current_language_tl _colon }
       }
    }
 } ,
time-variant / dialect-colon .code:n =
    {
      \tl_set:Nn \l__datatool_BE_timevariant_tl
        { \l_datatool_current_language_tl _colon }
    } ,
   time-variant / dialect-dot .code:n =
    {
      \tl_set:Nn \l__datatool_BE_timevariant_tl
        { \l_datatool_current_language_tl _dot }
    } ,
  }
```

3.2.4 Hooks

Command to update temporal parsing commands for this region:

```
\newcommand \datatoolBESetTemporalParsers
 {
   \renewcommand \DTLCurrentLocaleParseTimeStamp
    { \datatool_BE_parse_timestamp:NnTF }
   \renewcommand \DTLCurrentLocaleParseDate
    { \datatool_BE_parse_date:NnTF }
   \renewcommand \DTLCurrentLocaleParseTime
    { \datatool_BE_parse_time:NnTF }
   \let
    \DTLCurrentLocaleGetTimeZoneMap
    \datatool_BE_get_timezone_map:n
 }
```

Set temporal formatting commands for this region. Currently this just resets to the default, but may change in future. Note that the defaults test if the applicable datetime2 command is available and will fallback on ISO if not defined. Bear in mind that the default style for datetime2 is `iso` so there won't be a noticeable difference unless the datetime2 regional setting is on.

```
\newcommand \datatoolBESetTemporalFormatters
 {
   \let
    \DTLCurrentLocaleFormatDate
    \datatool_default_date_fmt:nnnn
   \let
    \DTLCurrentLocaleFormatTime
    \datatool_default_time_fmt:nnn
   \let
    \DTLCurrentLocaleFormatTimeZone
    \datatool_default_timezone_fmt:nn
   \let
    \DTLCurrentLocaleFormatTimeStampNoZone
    \datatool_default_timestamp_fmt:nnnnnnn
   \let
    \DTLCurrentLocaleFormatTimeStampWithZone
    \datatool_default_timestamp_fmt:nnnnnnnnn
   \renewcommand \DTLCurrentLocaleTimeStampFmtSep { ~ }
 }
```

Command to update currency and temporal parsing commands for this region:

```
\newcommand \DTLBELocaleHook
 {
  \datatoolBESetNumberChars
  \datatoolBESetCurrency
  \datatoolBESetTemporalParsers
  \datatoolBESetTemporalFormatters
```

Allow language files to reference the region:

```
  \tl_set:Nn \l_datatool_current_region_tl { BE }
 }
```

Finished with LaTeX3 syntax.

```
\ExplSyntaxOff
```

Note that the hook is added to the captions by datatool-base not by the region file.

3.3 datatool-CA.ldf

Support for region CA.

```
\TrackLangProvidesResource{CA}[2025/03/01 v1.0]
```

Switch on LaTeX3 syntax:

```
\ExplSyntaxOn
```

3.3.1 Numbers and Currency

NB the number group and decimal symbols for this region should be set in the files `datatool-en-CA.ldf` and `datatool-fr-CA.ldf` as they depend on both the language and region. Those files should be included in the applicable datatool language module. For example, `datatool-en-CA.ldf` is provided with `datatool-english`. However, a region hook is still needed to support `\DTLsetup{numeric={region-number-chars}}` although it's not added to the language hook:

```
\newcommand \datatoolCASetNumberChars
 {
   \bool_if:NT \l_datatool_region_set_numberchars_bool
    {
      \tl_if_empty:NTF \l_datatool_current_language_tl
       {
          \datatool_locale_warn:nn { datatool-CA }
           {
              No ~ current ~ language: ~ can't ~ set ~
              number ~ group ~ and ~ decimal ~ characters
           }
       }
       {
         \cs_if_exist_use:cF
          { datatool \l_datatool_current_language_tl CASetNumberChars }
           {
             \datatool_locale_warn:nn { datatool-CA }
```

```
              {
                No ~ support ~ for ~ locale ~
                ` \l_datatool_current_language_tl - CA': ~ can't ~ set ~
                number ~ group ~ and ~ decimal ~ characters
              }
            }
        }
     }
  }
```

How to format the position of the currency symbol in relation to the value.

```
\cs_new:Nn \datatool_CA_currency_position:nn
 {
   \dtlcurrprefixfmt { #1 } { #2 }
 }
```

Separator between currency symbol and value.

```
\tl_new:N \l_datatool_CA_sym_sep_tl
```

Set the currency format for this region.

```
\newcommand \datatoolCAcurrencyfmt [ 2 ]
 {
   \datatool_CA_currency_position:nn
    {
      \datatoolCAsymbolprefix { CA }
      #1
    }
    { #2 }
 }
```

Prefix for symbol, if required.

```
\newcommand \datatoolCAsymbolprefix [ 1 ] { }
```

Define the currency symbols for this region.

```
\datatool_def_currency:nnnV
 { \datatoolCAcurrencyfmt }
 { CAD }
 { \$ }
 \c_dollar_str
```

Register the currency code with this region:

```
\datatool_register_regional_currency_code:nn { CA } { CAD }
```

Provide a command to set the currency for this region (for use with any hook used when the locale changes). NB this should do nothing with `\DTLsetup{region-currency=false}`

```
\newcommand \datatoolCASetCurrency
 {
   \bool_if:NT \l_datatool_region_set_currency_bool
    {
      \DTLsetdefaultcurrency { CAD }
```

Number of digits that `\DTLdecimaltocurrency` should round to:

```
      \renewcommand \DTLCurrentLocaleCurrencyDP { 2 }
```

Separator between symbol and value:

```
      \renewcommand \dtlcurrfmtsymsep { \l_datatool_CA_sym_sep_tl }
    }
  }
```

3.3.2 Date and Time Parsing

Date parsing is more complicated as there seems to be a confusing mixture of ISO, day/-month/year and month/day/year formats. This defaults to year-month-day and should be changed as appropriate using `\DTLsetLocaleOptions`. For example:

```
\DTLsetLocaleOptions{CA}{date-style=dmyyyy, date-variant=slash}
```

An appropriate language file will need to also be installed to parse dates containing month names or day of week names.

Provide a way to configure parsing style.

```
\tl_new:N \l__datatool_CA_datevariant_tl
\tl_set:Nn \l__datatool_CA_datevariant_tl { hyphen }
```

NB These token lists are used to form command names. The following is not a format string.

```
\tl_new:N \l__datatool_CA_datestyle_tl
\tl_set:Nn \l__datatool_CA_datestyle_tl { yyyymmdd }
\tl_new:N \l__datatool_CA_timevariant_tl
\tl_set:Nn \l__datatool_CA_timevariant_tl { colon }
```

Each parsing command defined below has final {⟨*true*⟩} and {⟨*false*⟩} arguments (TF). These are used if parsing was successful (true) or if parsing failed (false). The internal commands used by datatool-base have no need for solo branches (only T or only F) so these commands are simply defined with `\cs_new:Nn` not as conditionals.

Time Stamp Parsing

Use command

`\datatool_`⟨*date-style*⟩`_hhmmss_tz_parse_timestamp:nnNnTF`

with date regular expression

`\c_datatool_`⟨*date-variant*⟩`_`⟨*date-style*⟩`_date_regex`

and time regular expression

`\c_datatool_`⟨*time-variant*⟩`_hhmmss_time_regex`

```
\cs_new:Nn \datatool_CA_parse_timestamp:NnTF
 {
   \cs_if_exist:cTF
     {
       datatool_
       \l__datatool_CA_datestyle_tl
       _hhmmss_tz_parse_timestamp:ccNnTF
     }
    {
      \cs_if_exist:cTF
       {
          c_datatool_
          \l__datatool_CA_datevariant_tl
          _
          \l__datatool_CA_datestyle_tl
          _date_regex
       }
       {
         \cs_if_exist:cTF
          {
            c_datatool_
            \l__datatool_CA_timevariant_tl
            _hhmmss_time_regex
          }
          {
            \use:c
             {
               datatool_
              \l__datatool_CA_datestyle_tl
              _hhmmss_tz_parse_timestamp:ccNnTF
             }
             {
               c_datatool_
               \l__datatool_CA_datevariant_tl
               _
               \l__datatool_CA_datestyle_tl
               _date_regex
             }
             {
               c_datatool_
               \l__datatool_CA_timevariant_tl
               _hhmmss_time_regex
             }
              #1 { #2 } { #3 } { #4 }
          }
          {
            \datatool_locale_warn:nn { datatool-CA }
             {
              No ~ support ~ for ~ time ~ variant ~
              ` \l__datatool_CA_timevariant_tl '
             }
            #4
          }
       }
       {
         \datatool_locale_warn:nn { datatool-CA }
          {
           No ~ support ~ for ~ date ~ style ~
           ` \l__datatool_CA_datestyle_tl ' ~ with ~
           variant ~
           ` \l__datatool_CA_datevariant_tl '
          }
         #4
       }
    }
    {
      \datatool_locale_warn:nn { datatool-CA }
       {
          No ~ support ~ for ~ date ~ style ~
          ` \l__datatool_CA_datestyle_tl '
       }
      #4
    }
 }
```

Date Parsing

Use command `\datatool_`⟨*style*⟩`_parse_date:NNnTF` with regular expression

`\c_datatool_`⟨*variant*⟩`_anchored_`⟨*style*⟩`_date_regex`

```
\cs_new:Nn \datatool_CA_parse_date:NnTF
 {
   \cs_if_exist:cTF
     {
       datatool_
       \l__datatool_CA_datestyle_tl
       _parse_date:NNnTF
     }
    {
      \cs_if_exist:cTF
       {
         c_datatool_
         \l__datatool_CA_datevariant_tl
         _anchored_
         \l__datatool_CA_datestyle_tl
         _date_regex
       }
       {
         \exp_args:cc
         {
           datatool_
           \l__datatool_CA_datestyle_tl
           _parse_date:NNnTF
         }
         {
           c_datatool_
           \l__datatool_CA_datevariant_tl
           _anchored_
           \l__datatool_CA_datestyle_tl
           _date_regex
         }
          #1 { #2 } { #3 } { #4 }
       }
       {
         \datatool_locale_warn:nn { datatool-CA }
          {
           No ~ support ~ for ~ date ~ style ~
           ` \l__datatool_CA_datestyle_tl ' ~ with ~
           variant ~
           ` \l__datatool_CA_datevariant_tl '
          }
         #4
       }
    }
    {
      \datatool_locale_warn:nn { datatool-CA }
       {
          No ~ support ~ for ~ date ~ style ~
          ` \l__datatool_CA_datestyle_tl '
       }
      #4
    }
 }
```

Time Parsing

Use command `\datatool_⟨style⟩_parse_time:NNnTF` with regular expression

`\c_datatool_⟨variant⟩_anchored_⟨style⟩_time_regex`

```
\cs_new:Nn \datatool_CA_parse_time:NnTF
 {
   \cs_if_exist:cTF
    {
      c_datatool_
      \l__datatool_CA_timevariant_tl
      _anchored_hhmmss_time_regex
    }
    {
      \datatool_hhmmss_parse_time:cNnTF
       {
        c_datatool_
        \l__datatool_CA_timevariant_tl
        _anchored_hhmmss_time_regex
       }
        #1 { #2 } { #3 } { #4 }
    }
    {
      \datatool_locale_warn:nn { datatool-CA }
       {
        No ~ support ~ for ~ time ~ variant ~
        ` \l__datatool_CA_timevariant_tl '
       }
      #4
    }
 }
```

Time Zone Mappings

Define time zone mapping command for this region:

```
\cs_new:Nn \datatool_CA_get_timezone_map:n
 {
   \datatool_region_get_timezone_map:n { CA / #1 }
 }
```

Define time zone IDs for this region (one-way map from ID to offset):

```
\datatool_region_set_timezone_map:nn { CA / NDT } { -02:30 }
\datatool_region_set_timezone_map:nn { CA / NST } { -03:30 }
\datatool_region_set_timezone_map:nn { CA / AST } { -04:00 }
\datatool_region_set_timezone_map:nn { CA / ADT } { -03:00 }
\datatool_region_set_timezone_map:nn { CA / EST } { -05:00 }
\datatool_region_set_timezone_map:nn { CA / EDT } { -04:00 }
\datatool_region_set_timezone_map:nn { CA / CST } { -06:00 }
\datatool_region_set_timezone_map:nn { CA / CDT } { -05:00 }
\datatool_region_set_timezone_map:nn { CA / MST } { -07:00 }
\datatool_region_set_timezone_map:nn { CA / MDT } { -06:00 }
\datatool_region_set_timezone_map:nn { CA / PST } { -08:00 }
\datatool_region_set_timezone_map:nn { CA / PDT } { -07:00 }
```

3.3.3 Options

Define options for this region:

```
\datatool_locale_define_keys:nn { CA }
 {
   number-style .code:n =
    {
      \exp_args:Ne \keys_if_exist:nnTF
         { datatool / locale / \l_datatool_current_language_tl - CA }
         { number-style }
       {
         \exp_args:Ne \keys_set:nn
          { datatool / locale / \l_datatool_current_language_tl - CA }
          { number-style = { #1 } }
       }
       {
         \datatool_locale_warn:nn { datatool-CA }
          {
            No ~ number-style ~ available ~ for ~ current ~ language ~
            ` \l_datatool_current_language_tl ' ~
            (additional ~ language ~ module ~ may ~ need ~ installing)
          }
       }
    } ,
```

Currency symbol before or after value:

```
    currency-symbol-position .choice: ,
    currency-symbol-position / before .code:n =
     {
       \cs_set_eq:NN \datatool_CA_currency_position:nn \dtlcurrprefixfmt
     } ,
    currency-symbol-position / after .code:n =
     {
       \cs_set_eq:NN \datatool_CA_currency_position:nn \dtlcurrsuffixfmt
     } ,
```

Should the currency symbol be prefixed with the region code:

```
    currency-symbol-prefix .choice: ,
    currency-symbol-prefix / false .code:n =
     {
       \cs_set_eq:NN \datatoolCAsymbolprefix \use_none:n
     } ,
    currency-symbol-prefix / true .code:n =
     {
       \cs_set_eq:NN
         \datatoolCAsymbolprefix
         \datatool_currency_symbol_region_prefix:n
     } ,
    currency-symbol-prefix .default:n = { true } ,
```

Separator between currency symbol (not code) and value:

```
    currency-symbol-sep .choice: ,
    currency-symbol-sep / none .code:n =
     {
       \tl_clear:N \l_datatool_CA_sym_sep_tl
     } ,
    currency-symbol-sep / thin-space .code:n =
     {
       \tl_set:Nn \l_datatool_CA_sym_sep_tl { \, }
     } ,
    currency-symbol-sep / space .code:n =
     {
       \tl_set:Nn \l_datatool_CA_sym_sep_tl { ~ }
     } ,
    currency-symbol-sep / nbsp .code:n =
     {
       \tl_set:Nn \l_datatool_CA_sym_sep_tl { \nobreakspace }
     } ,
    currency-symbol-sep .initial:n = { none } ,
```

Date and time styles:

```
date-style .choice: ,
date-style / dmyyyy .code: n =
 {
   \tl_set:Nn \l__datatool_CA_datestyle_tl { ddmmyyyy }
 } ,
date-style / mdyyyy .code: n =
 {
   \tl_set:Nn \l__datatool_CA_datestyle_tl { mmddyyyy }
 } ,
date-style / yyyymd .code: n =
 {
   \tl_set:Nn \l__datatool_CA_datestyle_tl { yyyymmdd }
 } ,
date-style / dmyy .code: n =
 {
   \tl_set:Nn \l__datatool_CA_datestyle_tl { ddmmyy }
 } ,
date-style / mdyy .code: n =
 {
   \tl_set:Nn \l__datatool_CA_datestyle_tl { mmddyy }
 } ,
date-style / yymd .code: n =
 {
   \tl_set:Nn \l__datatool_CA_datestyle_tl { yymmdd }
 } ,
date-variant .choice: ,
date-variant / slash .code:n =
 {
   \tl_set:Nn \l__datatool_CA_datevariant_tl { slash }
 } ,
date-variant / hyphen .code:n =
 {
   \tl_set:Nn \l__datatool_CA_datevariant_tl { hyphen }
 } ,
date-variant / dot .code:n =
 {
   \tl_set:Nn \l__datatool_CA_datevariant_tl { dot }
 } ,
date-variant / dialect .code:n =
 {
   \tl_set:Nn \l__datatool_CA_datevariant_tl
     { \l_datatool_current_language_tl }
 } ,
time-variant .choice: ,
time-variant / colon .code:n =
 {
   \tl_set:Nn \l__datatool_CA_timevariant_tl { colon }
 } ,
time-variant / dot .code:n =
 {
   \tl_set:Nn \l__datatool_CA_timevariant_tl { dot }
 } ,
time-variant / dialect .code:n =
    {
      \tl_if_eq:NnTF \l__datatool_CA_timevariant_tl { dot }
       {
         \tl_set:Nn \l__datatool_CA_timevariant_tl
          { \l_datatool_current_language_tl _dot }
       }
       {
         \tl_if_in:NnF
           \l__datatool_CA_timevariant_tl
            { \l_datatool_current_language_tl }
          {
            \tl_set:Nn \l__datatool_CA_timevariant_tl
             { \l_datatool_current_language_tl _colon }
          }
       }
    } ,
   time-variant / dialect-colon .code:n =
    {
      \tl_set:Nn \l__datatool_CA_timevariant_tl
        { \l_datatool_current_language_tl _colon }
    } ,
   time-variant / dialect-dot .code:n =
    {
      \tl_set:Nn \l__datatool_CA_timevariant_tl
        { \l_datatool_current_language_tl _dot }
    } ,
 }
```

3.3.4 Hooks

Command to update temporal parsing commands for this region:

```
\newcommand \datatoolCASetTemporalParsers
 {
   \renewcommand \DTLCurrentLocaleParseTimeStamp
    { \datatool_CA_parse_timestamp:NnTF }
   \renewcommand \DTLCurrentLocaleParseDate
    { \datatool_CA_parse_date:NnTF }
   \renewcommand \DTLCurrentLocaleParseTime
    { \datatool_CA_parse_time:NnTF }
   \let
    \DTLCurrentLocaleGetTimeZoneMap
    \datatool_CA_get_timezone_map:n
 }
```

Set temporal formatting commands for this region. Currently this just resets to the default, but may change in future. Note that the defaults test if the applicable datetime2 command is available and will fallback on ISO if not defined. Bear in mind that the default style for datetime2 is `iso` so there won't be a noticeable difference unless the datetime2 regional setting is on.

```
\newcommand \datatoolCASetTemporalFormatters
 {
   \let
    \DTLCurrentLocaleFormatDate
    \datatool_default_date_fmt:nnnn
   \let
    \DTLCurrentLocaleFormatTime
    \datatool_default_time_fmt:nnn
   \let
    \DTLCurrentLocaleFormatTimeZone
    \datatool_default_timezone_fmt:nn
   \let
    \DTLCurrentLocaleFormatTimeStampNoZone
    \datatool_default_timestamp_fmt:nnnnnnn
   \let
    \DTLCurrentLocaleFormatTimeStampWithZone
    \datatool_default_timestamp_fmt:nnnnnnnnn
   \renewcommand \DTLCurrentLocaleTimeStampFmtSep { ~ }
 }
```

Command to update currency and temporal parsing commands for this region:

```
\newcommand \DTLCALocaleHook
 {
  \datatoolCASetCurrency
  \datatoolCASetTemporalParsers
  \datatoolCASetTemporalFormatters
```

Allow language files to reference the region:

```
  \tl_set:Nn \l_datatool_current_region_tl { CA }
 }
```

Finished with LaTeX3 syntax.

```
\ExplSyntaxOff
```

Note that the hook is added to the captions by datatool-base not by the region file.

3.4 datatool-FK.ldf

Support for region FK.

```
\TrackLangProvidesResource{FK}[2025/03/01 v1.0]
```

Switch on LaTeX3 syntax:

```
\ExplSyntaxOn
```

3.4.1 Numbers and Currency

Set the number group and decimal symbols for this region.

```
\cs_new:Nn \datatool_FK_set_numberchars_official:
 {
  \datatool_set_numberchars:nn { , } { . }
 }
```

Unofficial style.

```
\cs_new:Nn \datatool_FK_set_numberchars_unofficial:
 {
  \datatool_set_thinspace_group_decimal_char:n { . }
 }
```

Hook to set the number group and decimal characters for the region:

```
\newcommand \datatoolFKSetNumberChars
 {
   \bool_if:NT \l_datatool_region_set_numberchars_bool
    {
      \datatool_FK_set_numberchars_official:
    }
 }
```

How to format the position of the currency symbol in relation to the value.

```
\cs_new:Nn \datatool_FK_currency_position:nn
 {
   \dtlcurrprefixfmt { #1 } { #2 }
 }
```

Separator between currency symbol and value.

```
\tl_new:N \l_datatool_FK_sym_sep_tl
```

Set the currency format for this region.

```
\newcommand \datatoolFKcurrencyfmt [ 2 ]
 {
   \datatool_FK_currency_position:nn
    {
      \datatoolFKsymbolprefix { FK }
      #1
    }
    { #2 }
 }
```

Prefix for symbol, if required.

```
\newcommand \datatoolFKsymbolprefix [ 1 ] { }
```

Define the currency symbols for this region.

```
\datatool_def_currency:nnnV
 { \datatoolFKcurrencyfmt }
 { FKP }
 { \pounds }
 \l_datatool_pound_tl
```

Register the currency code with this region:

```
\datatool_register_regional_currency_code:nn { FK } { FKP }
```

Provide a command to set the currency for this region (for use with any hook used when the locale changes). NB this should do nothing with `\DTLsetup{region-currency=false}`

```
\newcommand \datatoolFKSetCurrency
 {
   \bool_if:NT \l_datatool_region_set_currency_bool
    {
      \DTLsetdefaultcurrency { FKP }
```

Number of digits that `\DTLdecimaltocurrency` should round to:

```
        \renewcommand \DTLCurrentLocaleCurrencyDP { 2 }
```

Separator between symbol and value:

```
        \renewcommand \dtlcurrfmtsymsep { \l_datatool_FK_sym_sep_tl }
      }
   }
```

3.4.2 Date and Time Parsing

This defaults to day/month/year but may be changed with `\DTLsetLocaleOptions`. For example:

```
\DTLsetLocaleOptions{FK}{date-style=dmyyyy, date-variant=slash}
```

An appropriate language file will need to also be installed to parse dates containing month names or day of week names.

Provide a way to configure parsing style.

```
\tl_new:N \l__datatool_FK_datevariant_tl
\tl_set:Nn \l__datatool_FK_datevariant_tl { slash }
```

NB These token lists are used to form command names. The following is not a format string.

```
\tl_new:N \l__datatool_FK_datestyle_tl
\tl_set:Nn \l__datatool_FK_datestyle_tl { ddmmyyyy }
\tl_new:N \l__datatool_FK_timevariant_tl
\tl_set:Nn \l__datatool_FK_timevariant_tl { colon }
```

Each parsing command defined below has final {⟨*true*⟩} and {⟨*false*⟩} arguments (TF). These are used if parsing was successful (true) or if parsing failed (false). The internal commands used by datatool-base have no need for solo branches (only T or only F) so these commands are simply defined with `\cs_new:Nn` not as conditionals.

Time Stamp Parsing

Use command

```
\datatool_⟨date-style⟩_hhmmss_tz_parse_timestamp:nnNnTF
```

with date regular expression

```
\c_datatool_⟨date-variant⟩_⟨date-style⟩_date_regex
```

and time regular expression

```
\c_datatool_⟨time-variant⟩_hhmmss_time_regex
```

```
\cs_new:Nn \datatool_FK_parse_timestamp:NnTF
 {
   \cs_if_exist:cTF
     {
       datatool_
       \l__datatool_FK_datestyle_tl
       _hhmmss_tz_parse_timestamp:ccNnTF
     }
    {
      \cs_if_exist:cTF
       {
          c_datatool_
          \l__datatool_FK_datevariant_tl
          _
          \l__datatool_FK_datestyle_tl
          _date_regex
       }
       {
         \cs_if_exist:cTF
          {
            c_datatool_
            \l__datatool_FK_timevariant_tl
            _hhmmss_time_regex
          }
          {
            \use:c
             {
               datatool_
              \l__datatool_FK_datestyle_tl
              _hhmmss_tz_parse_timestamp:ccNnTF
             }
             {
               c_datatool_
               \l__datatool_FK_datevariant_tl
               _
               \l__datatool_FK_datestyle_tl
               _date_regex
             }
             {
               c_datatool_
               \l__datatool_FK_timevariant_tl
               _hhmmss_time_regex
             }
              #1 { #2 } { #3 } { #4 }
          }
          {
            \datatool_warn_check_head_language_empty:Vnnn
             \l__datatool_FK_timevariant_tl
             { datatool-FK }
             {
               No ~ language ~ support ~ for ~ time ~ variant ~
               ` \exp_args:Ne \tl_tail:n { \l__datatool_FK_timevariant_tl } '
             }
             {
              No ~ support ~ for ~ time ~ variant ~
              ` \l__datatool_FK_timevariant_tl '
             }
            #4
          }
       }
       {
         \datatool_warn_check_head_language_empty:Vnnn
           \l__datatool_FK_datevariant_tl
           { datatool-FK }
           {
             No ~ language ~ support ~ for ~ date ~ style ~
              ` \l__datatool_FK_datestyle_tl '
           }
           {
             No ~ support ~ for ~ date ~ style ~
             ` \l__datatool_FK_datestyle_tl ' ~ with ~
             variant ~
             ` \l__datatool_FK_datevariant_tl '
           }
         #4
       }
    }
    {
      \datatool_locale_warn:nn { datatool-FK }
       {
          No ~ support ~ for ~ date ~ style ~
          ` \l__datatool_FK_datestyle_tl '
       }
      #4
    }
 }
```

Date Parsing

Use command `\datatool_`⟨*style*⟩`_parse_date:NNnTF` with regular expression

`\c_datatool_`⟨*variant*⟩`_anchored_`⟨*style*⟩`_date_regex`

```
\cs_new:Nn \datatool_FK_parse_date:NnTF
 {
   \cs_if_exist:cTF
     {
       datatool_
       \l__datatool_FK_datestyle_tl
       _parse_date:NNnTF
     }
    {
      \cs_if_exist:cTF
       {
         c_datatool_
         \l__datatool_FK_datevariant_tl
         _anchored_
         \l__datatool_FK_datestyle_tl
         _date_regex
       }
       {
         \exp_args:cc
         {
           datatool_
           \l__datatool_FK_datestyle_tl
           _parse_date:NNnTF
         }
         {
           c_datatool_
           \l__datatool_FK_datevariant_tl
           _anchored_
           \l__datatool_FK_datestyle_tl
           _date_regex
         }
          #1 { #2 } { #3 } { #4 }
       }
       {
         \datatool_warn_check_head_language_empty:Vnnn
          \l__datatool_FK_datevariant_tl
          { datatool-FK }
          {
            No ~ language ~ support ~ for ~ date ~ style ~
             ` \l__datatool_FK_datestyle_tl '
          }
          {
             No ~ support ~ for ~ date ~ style ~
             ` \l__datatool_FK_datestyle_tl ' ~ with ~
             variant ~
             ` \l__datatool_FK_datevariant_tl '
          }
         #4
       }
    }
    {
      \datatool_locale_warn:nn { datatool-FK }
       {
          No ~ support ~ for ~ date ~ style ~
          ` \l__datatool_FK_datestyle_tl '
       }
      #4
    }
 }
```

Time Parsing

Use command `\datatool_`⟨*style*⟩`_parse_time:NNnTF` with regular expression

\c_datatool_⟨*variant*⟩_anchored_⟨*style*⟩_time_regex

```
\cs_new:Nn \datatool_FK_parse_time:NnTF
 {
   \cs_if_exist:cTF
    {
      c_datatool_
      \l__datatool_FK_timevariant_tl
      _anchored_hhmmss_time_regex
    }
    {
      \datatool_hhmmss_parse_time:cNnTF
       {
        c_datatool_
        \l__datatool_FK_timevariant_tl
        _anchored_hhmmss_time_regex
       }
        #1 { #2 } { #3 } { #4 }
    }
    {
      \datatool_warn_check_head_language_empty:Vnnn
        \l__datatool_FK_timevariant_tl
        { datatool-FK }
        {
           No ~ language ~ support ~ for ~ time ~ variant ~
           ` \exp_args:Ne \tl_tail:n { \l__datatool_FK_timevariant_tl } '
        }
        {
           No ~ support ~ for ~ time ~ variant ~
           ` \l__datatool_FK_timevariant_tl '
        }
      #4
    }
 }
```

Time Zone Mappings

Define time zone mapping command for this region:

```
\cs_new:Nn \datatool_FK_get_timezone_map:n
 {
   \datatool_region_get_timezone_map:n { FK / #1 }
 }
```

Define time zone IDs for this region (one-way map from ID to offset):

```
\datatool_region_set_timezone_map:nn { FK / BST } { +01:00 }
\datatool_region_set_timezone_map:nn { FK / FKST } { -03:00 }
\datatool_region_set_timezone_map:nn { FK / GMT } { +00:00 }
```

3.4.3 Options

Define options for this region:

```
\datatool_locale_define_keys:nn { FK }
 {
   number-style .choices:nn =
    { official, unofficial }
    {
      \exp_args:NNe \renewcommand
        \datatoolFKSetNumberChars
         {
           \exp_not:N \bool_if:NT
            \exp_not:N \l_datatool_region_set_numberchars_bool
             {
               \exp_not:c { datatool_FK_set_numberchars_ \l_keys_choice_tl : }
             }
         }
      \tl_if_eq:NnT \l_datatool_current_region_tl { FK }
       {
         \datatoolFKSetNumberChars
       }
    } ,
```

Currency symbol before or after value:

```
   currency-symbol-position .choice: ,
   currency-symbol-position / before .code:n =
    {
      \cs_set_eq:NN \datatool_FK_currency_position:nn \dtlcurrprefixfmt
    } ,
   currency-symbol-position / after .code:n =
    {
      \cs_set_eq:NN \datatool_FK_currency_position:nn \dtlcurrsuffixfmt
    } ,
```

Should the currency symbol be prefixed with the region code:

```
   currency-symbol-prefix .choice: ,
   currency-symbol-prefix / false .code:n =
    {
      \cs_set_eq:NN \datatoolFKsymbolprefix \use_none:n
    } ,
   currency-symbol-prefix / true .code:n =
    {
      \cs_set_eq:NN
       \datatoolFKsymbolprefix
       \datatool_currency_symbol_region_prefix:n
    } ,
   currency-symbol-prefix .default:n = { true } ,
```

Separator between currency symbol (not code) and value:

```
   currency-symbol-sep .choice: ,
   currency-symbol-sep / none .code:n =
    {
      \tl_clear:N \l_datatool_FK_sym_sep_tl
    } ,
   currency-symbol-sep / thin-space .code:n =
    {
      \tl_set:Nn \l_datatool_FK_sym_sep_tl { \, }
    } ,
   currency-symbol-sep / space .code:n =
    {
      \tl_set:Nn \l_datatool_FK_sym_sep_tl { ~ }
    } ,
   currency-symbol-sep / nbsp .code:n =
    {
      \tl_set:Nn \l_datatool_FK_sym_sep_tl { \nobreakspace }
    } ,
   currency-symbol-sep . initial:n = { none } ,
```

Date and time styles:

```
date-style .choice: ,
date-style / dmyyyy .code: n =
 {
   \tl_set:Nn \l__datatool_FK_datestyle_tl { ddmmyyyy }
 } ,
date-style / mdyyyy .code: n =
 {
   \tl_set:Nn \l__datatool_FK_datestyle_tl { mmddyyyy }
 } ,
date-style / yyyymd .code: n =
 {
   \tl_set:Nn \l__datatool_FK_datestyle_tl { yyyymmdd }
 } ,
date-style / dmyy .code: n =
 {
   \tl_set:Nn \l__datatool_FK_datestyle_tl { ddmmyy }
 } ,
date-style / mdyy .code: n =
 {
   \tl_set:Nn \l__datatool_FK_datestyle_tl { mmddyy }
 } ,
date-style / yymd .code: n =
 {
   \tl_set:Nn \l__datatool_FK_datestyle_tl { yymmdd }
 } ,
date-variant .choice: ,
date-variant / slash .code:n =
 {
   \tl_set:Nn \l__datatool_FK_datevariant_tl { slash }
 } ,
date-variant / hyphen .code:n =
 {
   \tl_set:Nn \l__datatool_FK_datevariant_tl { hyphen }
 } ,
date-variant / dot .code:n =
 {
   \tl_set:Nn \l__datatool_FK_datevariant_tl { dot }
 } ,
date-variant / dialect .code:n =
     {
       \tl_set:Nn \l__datatool_FK_datevariant_tl
         { \l_datatool_current_language_tl }
     } ,
    time-variant .choice: ,
    time-variant / colon .code:n =
     {
       \tl_set:Nn \l__datatool_FK_timevariant_tl { colon }
     } ,
    time-variant / dot .code:n =
     {
       \tl_set:Nn \l__datatool_FK_timevariant_tl { dot }
     } ,
    time-variant / dialect .code:n =
     {
       \tl_if_eq:NnTF \l__datatool_FK_timevariant_tl { dot }
        {
          \tl_set:Nn \l__datatool_FK_timevariant_tl
           { \l_datatool_current_language_tl _dot }
        }
        {
          \tl_if_in:NnF
            \l__datatool_FK_timevariant_tl
             { \l_datatool_current_language_tl }
           {
             \tl_set:Nn \l__datatool_FK_timevariant_tl
              { \l_datatool_current_language_tl _colon }
           }
        }
     } ,
    time-variant / dialect-colon .code:n =
     {
       \tl_set:Nn \l__datatool_FK_timevariant_tl
         { \l_datatool_current_language_tl _colon }
     } ,
    time-variant / dialect-dot .code:n =
     {
       \tl_set:Nn \l__datatool_FK_timevariant_tl
         { \l_datatool_current_language_tl _dot }
     } ,
  }
```

3.4.4 Hooks

Command to update temporal parsing commands for this region:

```
\newcommand \datatoolFKSetTemporalParsers
 {
   \renewcommand \DTLCurrentLocaleParseTimeStamp
    { \datatool_FK_parse_timestamp:NnTF }
   \renewcommand \DTLCurrentLocaleParseDate
    { \datatool_FK_parse_date:NnTF }
   \renewcommand \DTLCurrentLocaleParseTime
```

```
     { \datatool_FK_parse_time:NnTF }
    \let
     \DTLCurrentLocaleGetTimeZoneMap
     \datatool_FK_get_timezone_map:n
  }
```

Set temporal formatting commands for this region. Currently this just resets to the default, but may change in future. Note that the defaults test if the applicable datetime2 command is available and will fallback on ISO if not defined. Bear in mind that the default style for datetime2 is `iso` so there won't be a noticeable difference unless the datetime2 regional setting is on.

```
 \newcommand \datatoolFKSetTemporalFormatters
  {
    \let
     \DTLCurrentLocaleFormatDate
     \datatool_default_date_fmt:nnnn
    \let
     \DTLCurrentLocaleFormatTime
     \datatool_default_time_fmt:nnn
    \let
     \DTLCurrentLocaleFormatTimeZone
     \datatool_default_timezone_fmt:nn
    \let
     \DTLCurrentLocaleFormatTimeStampNoZone
     \datatool_default_timestamp_fmt:nnnnnnn
    \let
     \DTLCurrentLocaleFormatTimeStampWithZone
     \datatool_default_timestamp_fmt:nnnnnnnnn
    \renewcommand \DTLCurrentLocaleTimeStampFmtSep { ~ }
  }
```

Command to update currency and temporal parsing commands for this region:

```
 \newcommand \DTLFKLocaleHook
  {
   \datatoolFKSetNumberChars
   \datatoolFKSetCurrency
   \datatoolFKSetTemporalParsers
   \datatoolFKSetTemporalFormatters
```

Allow language files to reference the region:

```
   \tl_set:Nn \l_datatool_current_region_tl { FK }
  }
```

Finished with LaTeX3 syntax.

```
 \ExplSyntaxOff
```

Note that the hook is added to the captions by datatool-base not by the region file.

3.5 datatool-GB.ldf

Support for region GB.

```
 \TrackLangProvidesResource{GB}[2025/03/01 v1.0]
```

Switch on LaTeX3 syntax:

```
\ExplSyntaxOn
```

3.5.1 Numbers and Currency

Set the number group and decimal symbols for this region. Official style has comma number group and decimal dot.

```
\cs_new:Nn \datatool_GB_set_numberchars_official:
 {
   \DTLsetnumberchars { , } { . }
 }
```

This will allow a normal space, `\,` (thin space command) or the thin space Unicode character U+2009 as the number group separator when parsing and will use `\,` when formatting.

```
\cs_new:Nn \datatool_GB_set_numberchars_education:
 {
   \datatool_set_thinspace_group_decimal_char:n { . }
 }
```

Old style uses mid-dot as the decimal point but allows both middle dot and low dot when parsing.

```
\cs_new:Nn \datatool_GB_set_numberchars_old:
 {
   \datatool_set_numberchars_tl_regex:nnnn
     { , } { \textperiodcentered }
     { , } { \x{2E} | \u{l_datatool_middot_tl} | \c{textperiodcentered} }
 }
```

Hook to set the number group and decimal characters for the region:

```
\newcommand \datatoolGBSetNumberChars
 {
   \bool_if:NT \l_datatool_region_set_numberchars_bool
    {
      \datatool_GB_set_numberchars_official:
    }
 }
```

How to format the position of the currency symbol in relation to the value.

```
\cs_new:Nn \datatool_GB_currency_position:nn
 {
   \dtlcurrprefixfmt { #1 } { #2 }
 }
```

Separator between currency symbol and value.

```
\tl_new:N \l_datatool_GB_sym_sep_tl
```

Set the currency format for this region.

```
\newcommand \datatoolGBcurrencyfmt [ 2 ]
 {
   \datatool_GB_currency_position:nn
    {
      \datatoolGBsymbolprefix { GB }
      #1
    }
    { #2 }
 }
```

Prefix for symbol, if required.

```
 \newcommand \datatoolGBsymbolprefix [ 1 ] { }
```

Define the currency symbols for this region.

```
 \datatool_def_currency:nnnV
  { \datatoolGBcurrencyfmt }
  { GBP }
  { \pounds }
  \l_datatool_pound_tl
```

Register the currency code with this region:

```
 \datatool_register_regional_currency_code:nn { GB } { GBP }
```

Provide a command to set the currency for this region (for use with any hook used when the locale changes). NB this should do nothing with `\DTLsetup{region-currency=false}`

```
 \newcommand \datatoolGBSetCurrency
  {
    \bool_if:NT \l_datatool_region_set_currency_bool
     {
       \DTLsetdefaultcurrency { GBP }
```

Number of digits that `\DTLdecimaltocurrency` should round to:

```
       \renewcommand \DTLCurrentLocaleCurrencyDP { 2 }
```

Separator between symbol and value:

```
       \renewcommand \dtlcurrfmtsymsep { \l_datatool_GB_sym_sep_tl }
     }
  }
```

3.5.2 Date and Time Parsing

Provide a way to configure temporal parsing style. The default is day/month/year but may be changed with `\DTLsetLocaleOptions`. For example:

```
\DTLsetLocaleOptions{GB}{date-style=dmyy, date-variant=dot}
```

An appropriate language file will need to also be installed to parse dates containing month names or day of week names.

```
 \tl_new:N \l__datatool_GB_datevariant_tl
 \tl_set:Nn \l__datatool_GB_datevariant_tl { slash }
```

NB These token lists are used to form command names. The following is not a format string.

```
 \tl_new:N \l__datatool_GB_datestyle_tl
 \tl_set:Nn \l__datatool_GB_datestyle_tl { ddmmyyyy }
 \tl_new:N \l__datatool_GB_timevariant_tl
 \tl_set:Nn \l__datatool_GB_timevariant_tl { colon }
```

Each parsing command defined below has final {⟨*true*⟩} and {⟨*false*⟩} arguments (TF). These are used if parsing was successful (true) or if parsing failed (false). The internal commands used by datatool-base have no need for solo branches (only T or only F) so these commands are simply defined with `\cs_new:Nn` not as conditionals.

Time Stamp Parsing

Use command

`\datatool_`⟨*date-style*⟩`_hhmmss_tz_parse_timestamp:nnNnTF`

with date regular expression

`\c_datatool_`⟨*date-variant*⟩`_`⟨*date-style*⟩`_date_regex`

and time regular expression

`\c_datatool_`⟨*time-variant*⟩`_hhmmss_time_regex`

```
\cs_new:Nn \datatool_GB_parse_timestamp:NnTF
 {
   \cs_if_exist:cTF
     {
       datatool_
       \l__datatool_GB_datestyle_tl
       _hhmmss_tz_parse_timestamp:ccNnTF
     }
    {
      \cs_if_exist:cTF
       {
          c_datatool_
          \l__datatool_GB_datevariant_tl
          _
          \l__datatool_GB_datestyle_tl
          _date_regex
       }
       {
         \cs_if_exist:cTF
          {
            c_datatool_
            \l__datatool_GB_timevariant_tl
            _hhmmss_time_regex
          }
          {
            \use:c
             {
               datatool_
              \l__datatool_GB_datestyle_tl
              _hhmmss_tz_parse_timestamp:ccNnTF
             }
             {
               c_datatool_
               \l__datatool_GB_datevariant_tl
               _
               \l__datatool_GB_datestyle_tl
               _date_regex
             }
             {
               c_datatool_
               \l__datatool_GB_timevariant_tl
               _hhmmss_time_regex
             }
              #1 { #2 } { #3 } { #4 }
          }
          {
            \datatool_warn_check_head_language_empty:Vnnn
             \l__datatool_GB_timevariant_tl
             { datatool-GB }
             {
               No ~ language ~ support ~ for ~ time ~ variant ~
               ` \exp_args:Ne \tl_tail:n { \l__datatool_GB_timevariant_tl } '
             }
             {
              No ~ support ~ for ~ time ~ variant ~
              ` \l__datatool_GB_timevariant_tl '
             }
            #4
          }
       }
       {
         \datatool_warn_check_head_language_empty:Vnnn
           \l__datatool_GB_datevariant_tl
           { datatool-GB }
           {
             No ~ language ~ support ~ for ~ date ~ style ~
              ` \l__datatool_GB_datestyle_tl '
           }
           {
             No ~ support ~ for ~ date ~ style ~
             ` \l__datatool_GB_datestyle_tl ' ~ with ~
             variant ~
             ` \l__datatool_GB_datevariant_tl '
           }
         #4
       }
    }
    {
      \datatool_locale_warn:nn { datatool-GB }
       {
          No ~ support ~ for ~ date ~ style ~
          ` \l__datatool_GB_datestyle_tl '
```

```
        }
      #4
    }
  }
```

Date Parsing

Use command `\datatool_⟨style⟩_parse_date:NNnTF` with regular expression

`\c_datatool_⟨variant⟩_anchored_⟨style⟩_date_regex`

```
\cs_new:Nn \datatool_GB_parse_date:NnTF
 {
   \cs_if_exist:cTF
     {
       datatool_
       \l__datatool_GB_datestyle_tl
       _parse_date:NNnTF
     }
    {
      \cs_if_exist:cTF
       {
         c_datatool_
         \l__datatool_GB_datevariant_tl
         _anchored_
         \l__datatool_GB_datestyle_tl
         _date_regex
       }
       {
         \exp_args:cc
         {
           datatool_
           \l__datatool_GB_datestyle_tl
           _parse_date:NNnTF
         }
         {
           c_datatool_
           \l__datatool_GB_datevariant_tl
           _anchored_
           \l__datatool_GB_datestyle_tl
           _date_regex
         }
          #1 { #2 } { #3 } { #4 }
       }
       {
         \datatool_warn_check_head_language_empty:Vnnn
          \l__datatool_GB_datevariant_tl
          { datatool-GB }
          {
            No ~ language ~ support ~ for ~ date ~ style ~
             ` \l__datatool_GB_datestyle_tl '
          }
          {
             No ~ support ~ for ~ date ~ style ~
             ` \l__datatool_GB_datestyle_tl ' ~ with ~
             variant ~
             ` \l__datatool_GB_datevariant_tl '
          }
         #4
       }
    }
    {
      \datatool_locale_warn:nn { datatool-GB }
       {
          No ~ support ~ for ~ date ~ style ~
          ` \l__datatool_GB_datestyle_tl '
       }
      #4
    }
 }
```

Time Parsing

Use command `\datatool_`⟨*style*⟩`_parse_time:NNnTF` with regular expression

`\c_datatool_`⟨*variant*⟩`_anchored_`⟨*style*⟩`_time_regex`

```
\cs_new:Nn \datatool_GB_parse_time:NnTF
 {
   \cs_if_exist:cTF
    {
      c_datatool_
      \l__datatool_GB_timevariant_tl
      _anchored_hhmmss_time_regex
    }
    {
      \datatool_hhmmss_parse_time:cNnTF
       {
        c_datatool_
        \l__datatool_GB_timevariant_tl
        _anchored_hhmmss_time_regex
       }
        #1 { #2 } { #3 } { #4 }
    }
    {
      \datatool_warn_check_head_language_empty:Vnnn
        \l__datatool_GB_timevariant_tl
        { datatool-GB }
        {
           No ~ language ~ support ~ for ~ time ~ variant ~
           ` \exp_args:Ne \tl_tail:n { \l__datatool_GB_timevariant_tl } '
        }
        {
           No ~ support ~ for ~ time ~ variant ~
           ` \l__datatool_GB_timevariant_tl '
        }
      #4
    }
 }
```

Time Zones

Define time zone mapping command for this region:

```
\cs_new:Nn \datatool_GB_get_timezone_map:n
 {
   \datatool_region_get_timezone_map:n { GB / #1 }
 }
```

Define time zone IDs for this region (one-way map from ID to offset):

```
\datatool_region_set_timezone_map:nn { GB / GMT } { +00:00 }
\datatool_region_set_timezone_map:nn { GB / BST } { +01:00 }
\datatool_region_set_timezone_map:nn { GB / CET } { +01:00 }
\datatool_region_set_timezone_map:nn { GB / CEST } { +02:00 }
```

3.5.3 Options

Define options for this region:

```
\datatool_locale_define_keys:nn { GB }
 {
   number-style .choices:nn =
    { official , education , old }
    {
      \exp_args:NNe \renewcommand
        \datatoolGBSetNumberChars
         {
           \exp_not:N \bool_if:NT
            \exp_not:N \l_datatool_region_set_numberchars_bool
             {
               \exp_not:c { datatool_GB_set_numberchars_ \l_keys_choice_tl : }
             }
         }
      \tl_if_eq:NnT \l_datatool_current_region_tl { GB }
       {
         \datatoolGBSetNumberChars
       }
    } ,
```

Currency symbol before or after value:

```
   currency-symbol-position .choice: ,
   currency-symbol-position / before .code:n =
    {
      \cs_set_eq:NN \datatool_GB_currency_position:nn \dtlcurrprefixfmt
    } ,
   currency-symbol-position / after .code:n =
    {
      \cs_set_eq:NN \datatool_GB_currency_position:nn \dtlcurrsuffixfmt
    } ,
```

Should the currency symbol be prefixed with the region code:

```
currency-symbol-prefix .choice: ,
currency-symbol-prefix / false .code:n =
 {
   \cs_set_eq:NN \datatoolGBsymbolprefix \use_none:n
 } ,
currency-symbol-prefix / true .code:n =
 {
   \cs_set_eq:NN
    \datatoolGBsymbolprefix
    \datatool_currency_symbol_region_prefix:n
 } ,
currency-symbol-prefix .default:n = { true } ,
```

Separator between currency symbol (not code) and value:

```
currency-symbol-sep .choice: ,
currency-symbol-sep / none .code:n =
 {
   \tl_clear:N \l_datatool_GB_sym_sep_tl
 } ,
currency-symbol-sep / thin-space .code:n =
 {
   \tl_set:Nn \l_datatool_GB_sym_sep_tl { \, }
 } ,
currency-symbol-sep / space .code:n =
 {
   \tl_set:Nn \l_datatool_GB_sym_sep_tl { ~ }
 } ,
currency-symbol-sep / nbsp .code:n =
 {
   \tl_set:Nn \l_datatool_GB_sym_sep_tl { \nobreakspace }
 } ,
currency-symbol-sep .initial:n = { none } ,
```

Date and time styles:

```
date-style .choice: ,
date-style / dmyyyy .code: n =
 {
   \tl_set:Nn \l__datatool_GB_datestyle_tl { ddmmyyyy }
 } ,
date-style / mdyyyy .code: n =
 {
   \tl_set:Nn \l__datatool_GB_datestyle_tl { mmddyyyy }
 } ,
date-style / yyyymd .code: n =
 {
   \tl_set:Nn \l__datatool_GB_datestyle_tl { yyyymmdd }
 } ,
date-style / dmyy .code: n =
 {
   \tl_set:Nn \l__datatool_GB_datestyle_tl { ddmmyy }
 } ,
date-style / mdyy .code: n =
 {
   \tl_set:Nn \l__datatool_GB_datestyle_tl { mmddyy }
 } ,
date-style / yymd .code: n =
 {
   \tl_set:Nn \l__datatool_GB_datestyle_tl { yymmdd }
 } ,
date-variant .choice: ,
date-variant / slash .code:n =
 {
   \tl_set:Nn \l__datatool_GB_datevariant_tl { slash }
 } ,
date-variant / hyphen .code:n =
 {
   \tl_set:Nn \l__datatool_GB_datevariant_tl { hyphen }
 } ,
date-variant / dot .code:n =
 {
   \tl_set:Nn \l__datatool_GB_datevariant_tl { dot }
 } ,
date-variant / dialect .code:n =
 {
   \tl_set:Nn \l__datatool_GB_datevariant_tl
     { \l_datatool_current_language_tl }
 } ,
time-variant .choice: ,
time-variant / colon .code:n =
 {
   \tl_set:Nn \l__datatool_GB_timevariant_tl { colon }
 } ,
time-variant / dot .code:n =
 {
   \tl_set:Nn \l__datatool_GB_timevariant_tl { dot }
 } ,
time-variant / dialect .code:n =
 {
   \tl_if_eq:NnTF \l__datatool_GB_timevariant_tl { dot }
    {
      \tl_set:Nn \l__datatool_GB_timevariant_tl
       { \l_datatool_current_language_tl _dot }
    }
    {
      \tl_if_in:NnF
        \l__datatool_GB_timevariant_tl
         { \l_datatool_current_language_tl }
       {
         \tl_set:Nn \l__datatool_GB_timevariant_tl
          { \l_datatool_current_language_tl _colon }
       }
      }
    } ,
   time-variant / dialect-colon .code:n =
    {
      \tl_set:Nn \l__datatool_GB_timevariant_tl
        { \l_datatool_current_language_tl _colon }
    } ,
   time-variant / dialect-dot .code:n =
    {
      \tl_set:Nn \l__datatool_GB_timevariant_tl
        { \l_datatool_current_language_tl _dot }
    } ,
 }
```

3.5.4 Hooks

Command to update temporal parsing commands for this region:

```
\newcommand \datatoolGBSetTemporalParsers
 {
   \renewcommand \DTLCurrentLocaleParseTimeStamp
    { \datatool_GB_parse_timestamp:NnTF }
   \renewcommand \DTLCurrentLocaleParseDate
    { \datatool_GB_parse_date:NnTF }
   \renewcommand \DTLCurrentLocaleParseTime
    { \datatool_GB_parse_time:NnTF }
   \let
    \DTLCurrentLocaleGetTimeZoneMap
    \datatool_GB_get_timezone_map:n
 }
```

Set temporal formatting commands for this region. Currently this just resets to the default, but may change in future. Note that the defaults test if the applicable datetime2 command is available and will fallback on ISO if not defined. Bear in mind that the default style for datetime2 is `iso` so there won't be a noticeable difference unless the datetime2 regional setting is on.

```
\newcommand \datatoolGBSetTemporalFormatters
 {
   \let
    \DTLCurrentLocaleFormatDate
    \datatool_default_date_fmt:nnnn
   \let
    \DTLCurrentLocaleFormatTime
    \datatool_default_time_fmt:nnn
   \let
    \DTLCurrentLocaleFormatTimeZone
    \datatool_default_timezone_fmt:nn
   \let
    \DTLCurrentLocaleFormatTimeStampNoZone
    \datatool_default_timestamp_fmt:nnnnnnn
   \let
    \DTLCurrentLocaleFormatTimeStampWithZone
    \datatool_default_timestamp_fmt:nnnnnnnnn
   \renewcommand \DTLCurrentLocaleTimeStampFmtSep { ~ }
 }
```

Command to update number symbols, currency and temporal parsing commands for this region:

```
\newcommand \DTLGBLocaleHook
 {
  \datatoolGBSetNumberChars
  \datatoolGBSetCurrency
  \datatoolGBSetTemporalParsers
  \datatoolGBSetTemporalFormatters
```

Allow language files to reference the region:

```
  \tl_set:Nn \l_datatool_current_region_tl { GB }
 }
```

Finished with LaTeX3 syntax.

```
\ExplSyntaxOff
```

Note that the hook is added to the captions by datatool-base not by the region file.

3.6 datatool-GG.ldf

Support for region GG.

```
\TrackLangProvidesResource{GG}[2025/03/01 v1.0]
```

Switch on LaTeX3 syntax:

```
\ExplSyntaxOn
```

3.6.1 Numbers and Currency

Set the number group and decimal symbols for this region.

```
\cs_new:Nn \datatool_GG_set_numberchars_official:
 {
  \datatool_set_numberchars:nn { , } { . }
 }
```

Unofficial style.

```
\cs_new:Nn \datatool_GG_set_numberchars_unofficial:
 {
  \datatool_set_thinspace_group_decimal_char:n { . }
 }
```

Hook to set the number group and decimal characters for the region:

```
\newcommand \datatoolGGSetNumberChars
 {
   \bool_if:NT \l_datatool_region_set_numberchars_bool
    {
      \datatool_GG_set_numberchars_official:
    }
 }
```

How to format the position of the currency symbol in relation to the value.

```
\cs_new:Nn \datatool_GG_currency_position:nn
 {
   \dtlcurrprefixfmt { #1 } { #2 }
 }
```

Separator between currency symbol and value.

```
\tl_new:N \l_datatool_GG_sym_sep_tl
```

Set the currency format for this region.

```
\newcommand \datatoolGGcurrencyfmt [ 2 ]
 {
   \datatool_GG_currency_position:nn
    {
      \datatoolGGsymbolprefix { GG }
      #1
    }
    { #2 }
 }
```

Prefix for symbol, if required.

```
\newcommand \datatoolGGsymbolprefix [ 1 ] { }
```

Define the currency symbols for this region.

```
\datatool_def_currency:nnnV
 { \datatoolGGcurrencyfmt }
 { GGP }
 { \pounds }
 \l_datatool_pound_tl
```

Register the currency code with this region:

```
\datatool_register_regional_currency_code:nn { GG } { GGP }
```

Provide a command to set the currency for this region (for use with any hook used when the locale changes). NB this should do nothing with `\DTLsetup{region-currency=false}`

```
\newcommand \datatoolGGSetCurrency
 {
   \bool_if:NT \l_datatool_region_set_currency_bool
    {
      \DTLsetdefaultcurrency { GGP }
```

Number of digits that `\DTLdecimaltocurrency` should round to:

```
      \renewcommand \DTLCurrentLocaleCurrencyDP { 2 }
```

Separator between symbol and value:

```
      \renewcommand \dtlcurrfmtsymsep { \l_datatool_GG_sym_sep_tl }
    }
 }
```

3.6.2 Date and Time Parsing

This defaults to day/month/year but may be changed with `\DTLsetLocaleOptions`. For example:

```
\DTLsetLocaleOptions{GG}{date-style=dmyyyy, date-variant=slash}
```

An appropriate language file will need to also be installed to parse dates containing month names or day of week names.

Provide a way to configure parsing style.

```
\tl_new:N \l__datatool_GG_datevariant_tl
\tl_set:Nn \l__datatool_GG_datevariant_tl { slash }
```

NB These token lists are used to form command names. The following is not a format string.

```
\tl_new:N \l__datatool_GG_datestyle_tl
\tl_set:Nn \l__datatool_GG_datestyle_tl { ddmmyyyy }
\tl_new:N \l__datatool_GG_timevariant_tl
\tl_set:Nn \l__datatool_GG_timevariant_tl { colon }
```

Each parsing command defined below has final {⟨*true*⟩} and {⟨*false*⟩} arguments (TF). These are used if parsing was successful (true) or if parsing failed (false). The internal commands used by datatool-base have no need for solo branches (only T or only F) so these commands are simply defined with `\cs_new:Nn` not as conditionals.

Time Stamp Parsing

Use command

`\datatool_`⟨*date-style*⟩`_hhmmss_tz_parse_timestamp:nnNnTF`

with date regular expression

`\c_datatool_`⟨*date-variant*⟩`_`⟨*date-style*⟩`_date_regex`

and time regular expression

`\c_datatool_`⟨*time-variant*⟩`_hhmmss_time_regex`

```
\cs_new:Nn \datatool_GG_parse_timestamp:NnTF
 {
   \cs_if_exist:cTF
     {
       datatool_
       \l__datatool_GG_datestyle_tl
       _hhmmss_tz_parse_timestamp:ccNnTF
     }
    {
      \cs_if_exist:cTF
       {
          c_datatool_
          \l__datatool_GG_datevariant_tl
          _
          \l__datatool_GG_datestyle_tl
          _date_regex
       }
       {
         \cs_if_exist:cTF
          {
            c_datatool_
            \l__datatool_GG_timevariant_tl
            _hhmmss_time_regex
          }
          {
            \use:c
             {
               datatool_
              \l__datatool_GG_datestyle_tl
              _hhmmss_tz_parse_timestamp:ccNnTF
             }
             {
               c_datatool_
               \l__datatool_GG_datevariant_tl
               _
               \l__datatool_GG_datestyle_tl
               _date_regex
             }
             {
               c_datatool_
               \l__datatool_GG_timevariant_tl
               _hhmmss_time_regex
             }
              #1 { #2 } { #3 } { #4 }
          }
          {
            \datatool_warn_check_head_language_empty:Vnnn
             \l__datatool_GG_timevariant_tl
             { datatool-GG }
             {
               No ~ language ~ support ~ for ~ time ~ variant ~
               ` \exp_args:Ne \tl_tail:n { \l__datatool_GG_timevariant_tl } '
             }
             {
              No ~ support ~ for ~ time ~ variant ~
              ` \l__datatool_GG_timevariant_tl '
             }
            #4
          }
       }
       {
         \datatool_warn_check_head_language_empty:Vnnn
           \l__datatool_GG_datevariant_tl
           { datatool-GG }
           {
             No ~ language ~ support ~ for ~ date ~ style ~
```

```
              ` \l__datatool_GG_datestyle_tl '
           }
           {
             No ~ support ~ for ~ date ~ style ~
             ` \l__datatool_GG_datestyle_tl ' ~ with ~
             variant ~
             ` \l__datatool_GG_datevariant_tl '
           }
         #4
       }
    }
    {
      \datatool_locale_warn:nn { datatool-GG }
       {
          No ~ support ~ for ~ date ~ style ~
          ` \l__datatool_GG_datestyle_tl '
       }
      #4
    }
 }
```

Date Parsing

Use command `\datatool_`⟨*style*⟩`_parse_date:NNnTF` with regular expression

`\c_datatool_`⟨*variant*⟩`_anchored_`⟨*style*⟩`_date_regex`

```
\cs_new:Nn \datatool_GG_parse_date:NnTF
 {
   \cs_if_exist:cTF
     {
       datatool_
       \l__datatool_GG_datestyle_tl
       _parse_date:NNnTF
     }
    {
      \cs_if_exist:cTF
       {
         c_datatool_
         \l__datatool_GG_datevariant_tl
         _anchored_
         \l__datatool_GG_datestyle_tl
         _date_regex
       }
       {
         \exp_args:cc
         {
           datatool_
           \l__datatool_GG_datestyle_tl
           _parse_date:NNnTF
         }
         {
           c_datatool_
           \l__datatool_GG_datevariant_tl
           _anchored_
           \l__datatool_GG_datestyle_tl
           _date_regex
         }
          #1 { #2 } { #3 } { #4 }
       }
       {
         \datatool_warn_check_head_language_empty:Vnnn
          \l__datatool_GG_datevariant_tl
          { datatool-GG }
          {
            No ~ language ~ support ~ for ~ date ~ style ~
             ` \l__datatool_GG_datestyle_tl '
          }
          {
             No ~ support ~ for ~ date ~ style ~
             ` \l__datatool_GG_datestyle_tl ' ~ with ~
             variant ~
             ` \l__datatool_GG_datevariant_tl '
          }
         #4
       }
    }
    {
      \datatool_locale_warn:nn { datatool-GG }
       {
          No ~ support ~ for ~ date ~ style ~
          ` \l__datatool_GG_datestyle_tl '
       }
      #4
    }
 }
```

Time Parsing

Use command `\datatool_⟨style⟩_parse_time`:NNnTF with regular expression

`\c_datatool_⟨variant⟩_anchored_⟨style⟩_time_regex`

```
\cs_new:Nn \datatool_GG_parse_time:NnTF
 {
   \cs_if_exist:cTF
    {
      c_datatool_
      \l__datatool_GG_timevariant_tl
      _anchored_hhmmss_time_regex
    }
    {
      \datatool_hhmmss_parse_time:cNnTF
       {
        c_datatool_
        \l__datatool_GG_timevariant_tl
        _anchored_hhmmss_time_regex
       }
        #1 { #2 } { #3 } { #4 }
    }
    {
      \datatool_warn_check_head_language_empty:Vnnn
        \l__datatool_GG_timevariant_tl
        { datatool-GG }
        {
           No ~ language ~ support ~ for ~ time ~ variant ~
           ` \exp_args:Ne \tl_tail:n { \l__datatool_GG_timevariant_tl } '
        }
        {
           No ~ support ~ for ~ time ~ variant ~
           ` \l__datatool_GG_timevariant_tl '
        }
      #4
    }
 }
```

Time Zone Mappings

Define time zone mapping command for this region:

```
\cs_new:Nn \datatool_GG_get_timezone_map:n
 {
   \datatool_region_get_timezone_map:n { GG / #1 }
 }
```

Define time zone IDs for this region (one-way map from ID to offset):

```
\datatool_region_set_timezone_map:nn { GG / BST } { +01:00 }
\datatool_region_set_timezone_map:nn { GG / GMT } { +00:00 }
\datatool_region_set_timezone_map:nn { GG / CET } { +01:00 }
\datatool_region_set_timezone_map:nn { GG / CEST } { +02:00 }
```

3.6.3 Options

Define options for this region:

```
\datatool_locale_define_keys:nn { GG }
 {
   number-style .choices:nn =
    { official, unofficial }
    {
      \exp_args:NNe \renewcommand
        \datatoolGGSetNumberChars
         {
           \exp_not:N \bool_if:NT
            \exp_not:N \l_datatool_region_set_numberchars_bool
```

```
          {
            \exp_not:c { datatool_GG_set_numberchars_ \l_keys_choice_tl : }
          }
      }
   \tl_if_eq:NnT \l_datatool_current_region_tl { GG }
    {
      \datatoolGGSetNumberChars
    }
 } ,
```

Currency symbol before or after value:

```
currency-symbol-position .choice: ,
currency-symbol-position / before .code:n =
 {
   \cs_set_eq:NN \datatool_GG_currency_position:nn \dtlcurrprefixfmt
 } ,
currency-symbol-position / after .code:n =
 {
   \cs_set_eq:NN \datatool_GG_currency_position:nn \dtlcurrsuffixfmt
 } ,
```

Should the currency symbol be prefixed with the region code:

```
currency-symbol-prefix .choice: ,
currency-symbol-prefix / false .code:n =
 {
   \cs_set_eq:NN \datatoolGGsymbolprefix \use_none:n
 } ,
currency-symbol-prefix / true .code:n =
 {
   \cs_set_eq:NN
    \datatoolGGsymbolprefix
    \datatool_currency_symbol_region_prefix:n
 } ,
currency-symbol-prefix .default:n = { true } ,
```

Separator between currency symbol (not code) and value:

```
currency-symbol-sep .choice: ,
currency-symbol-sep / none .code:n =
 {
   \tl_clear:N \l_datatool_GG_sym_sep_tl
 } ,
currency-symbol-sep / thin-space .code:n =
 {
   \tl_set:Nn \l_datatool_GG_sym_sep_tl { \, }
 } ,
currency-symbol-sep / space .code:n =
 {
   \tl_set:Nn \l_datatool_GG_sym_sep_tl { ~ }
 } ,
currency-symbol-sep / nbsp .code:n =
 {
   \tl_set:Nn \l_datatool_GG_sym_sep_tl { \nobreakspace }
 } ,
currency-symbol-sep . initial:n = { none } ,
```

Date and time styles:

```
date-style .choice: ,
date-style / dmyyyy .code: n =
 {
   \tl_set:Nn \l__datatool_GG_datestyle_tl { ddmmyyyy }
 } ,
date-style / mdyyyy .code: n =
 {
   \tl_set:Nn \l__datatool_GG_datestyle_tl { mmddyyyy }
 } ,
date-style / yyyymd .code: n =
 {
   \tl_set:Nn \l__datatool_GG_datestyle_tl { yyyymmdd }
 } ,
date-style / dmyy .code: n =
 {
   \tl_set:Nn \l__datatool_GG_datestyle_tl { ddmmyy }
 } ,
date-style / mdyy .code: n =
 {
   \tl_set:Nn \l__datatool_GG_datestyle_tl { mmddyy }
 } ,
date-style / yymd .code: n =
 {
   \tl_set:Nn \l__datatool_GG_datestyle_tl { yymmdd }
 } ,
date-variant .choice: ,
date-variant / slash .code:n =
 {
   \tl_set:Nn \l__datatool_GG_datevariant_tl { slash }
 } ,
date-variant / hyphen .code:n =
 {
   \tl_set:Nn \l__datatool_GG_datevariant_tl { hyphen }
 } ,
date-variant / dot .code:n =
 {
   \tl_set:Nn \l__datatool_GG_datevariant_tl { dot }
 } ,
date-variant / dialect .code:n =
 {
   \tl_set:Nn \l__datatool_GG_datevariant_tl
     { \l_datatool_current_language_tl }
 } ,
time-variant .choice: ,
time-variant / colon .code:n =
 {
   \tl_set:Nn \l__datatool_GG_timevariant_tl { colon }
 } ,
time-variant / dot .code:n =
 {
   \tl_set:Nn \l__datatool_GG_timevariant_tl { dot }
 } ,
   time-variant / dialect .code:n =
    {
      \tl_if_eq:NnTF \l__datatool_GG_timevariant_tl { dot }
       {
         \tl_set:Nn \l__datatool_GG_timevariant_tl
          { \l_datatool_current_language_tl _dot }
       }
       {
         \tl_if_in:NnF
           \l__datatool_GG_timevariant_tl
            { \l_datatool_current_language_tl }
          {
            \tl_set:Nn \l__datatool_GG_timevariant_tl
             { \l_datatool_current_language_tl _colon }
          }
       }
    } ,
   time-variant / dialect-colon .code:n =
    {
      \tl_set:Nn \l__datatool_GG_timevariant_tl
        { \l_datatool_current_language_tl _colon }
    } ,
   time-variant / dialect-dot .code:n =
    {
      \tl_set:Nn \l__datatool_GG_timevariant_tl
        { \l_datatool_current_language_tl _dot }
    } ,
 }
```

3.6.4 Hooks

Command to update temporal parsing commands for this region:

```
\newcommand \datatoolGGSetTemporalParsers
 {
   \renewcommand \DTLCurrentLocaleParseTimeStamp
    { \datatool_GG_parse_timestamp:NnTF }
   \renewcommand \DTLCurrentLocaleParseDate
    { \datatool_GG_parse_date:NnTF }
   \renewcommand \DTLCurrentLocaleParseTime
    { \datatool_GG_parse_time:NnTF }
   \let
    \DTLCurrentLocaleGetTimeZoneMap
    \datatool_GG_get_timezone_map:n
 }
```

Set temporal formatting commands for this region. Currently this just resets to the default, but may change in future. Note that the defaults test if the applicable datetime2 command is available and will fallback on ISO if not defined. Bear in mind that the default style for datetime2 is `iso` so there won't be a noticeable difference unless the datetime2 regional setting is on.

```
\newcommand \datatoolGGSetTemporalFormatters
```

```
 {
   \let
    \DTLCurrentLocaleFormatDate
    \datatool_default_date_fmt:nnnn
   \let
    \DTLCurrentLocaleFormatTime
    \datatool_default_time_fmt:nnn
   \let
    \DTLCurrentLocaleFormatTimeZone
    \datatool_default_timezone_fmt:nn
   \let
    \DTLCurrentLocaleFormatTimeStampNoZone
    \datatool_default_timestamp_fmt:nnnnnnn
   \let
    \DTLCurrentLocaleFormatTimeStampWithZone
    \datatool_default_timestamp_fmt:nnnnnnnnn
   \renewcommand \DTLCurrentLocaleTimeStampFmtSep { ~ }
 }
```

Command to update currency and temporal parsing commands for this region:

```
\newcommand \DTLGGLocaleHook
 {
  \datatoolGGSetNumberChars
  \datatoolGGSetCurrency
  \datatoolGGSetTemporalParsers
  \datatoolGGSetTemporalFormatters
```

Allow language files to reference the region:

```
  \tl_set:Nn \l_datatool_current_region_tl { GG }
 }
```

Finished with LaTeX3 syntax.

```
\ExplSyntaxOff
```

Note that the hook is added to the captions by datatool-base not by the region file.

3.7 datatool-GI.ldf

Support for region GI.

```
\TrackLangProvidesResource{GI}[2025/03/01 v1.0]
```

Switch on LaTeX3 syntax:

```
\ExplSyntaxOn
```

3.7.1 Numbers and Currency

Set the number group and decimal symbols for this region.

```
\cs_new:Nn \datatool_GI_set_numberchars_official:
 {
  \datatool_set_numberchars:nn { , } { . }
 }
```

Unofficial style.

```
\cs_new:Nn \datatool_GI_set_numberchars_unofficial:
 {
  \datatool_set_thinspace_group_decimal_char:n { . }
 }
```

Hook to set the number group and decimal characters for the region:

```
\newcommand \datatoolGISetNumberChars
 {
   \bool_if:NT \l_datatool_region_set_numberchars_bool
    {
      \datatool_GI_set_numberchars_official:
    }
 }
```

How to format the position of the currency symbol in relation to the value.

```
\cs_new:Nn \datatool_GI_currency_position:nn
 {
   \dtlcurrprefixfmt { #1 } { #2 }
 }
```

Separator between currency symbol and value.

```
\tl_new:N \l_datatool_GI_sym_sep_tl
```

Set the currency format for this region.

```
\newcommand \datatoolGIcurrencyfmt [ 2 ]
 {
   \datatool_GI_currency_position:nn
    {
      \datatoolGIsymbolprefix { GI }
      #1
    }
    { #2 }
 }
```

Prefix for symbol, if required.

```
\newcommand \datatoolGIsymbolprefix [ 1 ] { }
```

Define the currency symbols for this region.

```
\datatool_def_currency:nnnV
 { \datatoolGIcurrencyfmt }
 { GIP }
 { \pounds }
 \l_datatool_pound_tl
```

Register the currency code with this region:

```
\datatool_register_regional_currency_code:nn { GI } { GIP }
```

Provide a command to set the currency for this region (for use with any hook used when the locale changes). NB this should do nothing with `\DTLsetup{region-currency=false}`

```
\newcommand \datatoolGISetCurrency
 {
   \bool_if:NT \l_datatool_region_set_currency_bool
    {
      \DTLsetdefaultcurrency { GIP }
```

Number of digits that `\DTLdecimaltocurrency` should round to:

```
        \renewcommand \DTLCurrentLocaleCurrencyDP { 2 }
```

Separator between symbol and value:

```
        \renewcommand \dtlcurrfmtsymsep { \l_datatool_GI_sym_sep_tl }
      }
   }
```

3.7.2 Date and Time Parsing

This defaults to day/month/year but may be changed with `\DTLsetLocaleOptions`. For example:

```
\DTLsetLocaleOptions{GI}{date-style=dmyyyy, date-variant=slash}
```

An appropriate language file will need to also be installed to parse dates containing month names or day of week names.

Provide a way to configure parsing style.

```
\tl_new:N \l__datatool_GI_datevariant_tl
\tl_set:Nn \l__datatool_GI_datevariant_tl { slash }
```

NB These token lists are used to form command names. The following is not a format string.

```
\tl_new:N \l__datatool_GI_datestyle_tl
\tl_set:Nn \l__datatool_GI_datestyle_tl { ddmmyyyy }
\tl_new:N \l__datatool_GI_timevariant_tl
\tl_set:Nn \l__datatool_GI_timevariant_tl { colon }
```

Each parsing command defined below has final {⟨*true*⟩} and {⟨*false*⟩} arguments (TF). These are used if parsing was successful (true) or if parsing failed (false). The internal commands used by datatool-base have no need for solo branches (only T or only F) so these commands are simply defined with `\cs_new:Nn` not as conditionals.

Time Stamp Parsing

Use command

`\datatool_`⟨*date-style*⟩`_hhmmss_tz_parse_timestamp:nnNnTF`

with date regular expression

`\c_datatool_`⟨*date-variant*⟩`_`⟨*date-style*⟩`_date_regex`

and time regular expression

`\c_datatool_`⟨*time-variant*⟩`_hhmmss_time_regex`

```
\cs_new:Nn \datatool_GI_parse_timestamp:NnTF
 {
   \cs_if_exist:cTF
     {
       datatool_
       \l__datatool_GI_datestyle_tl
       _hhmmss_tz_parse_timestamp:ccNnTF
     }
    {
      \cs_if_exist:cTF
       {
          c_datatool_
          \l__datatool_GI_datevariant_tl
          _
          \l__datatool_GI_datestyle_tl
          _date_regex
       }
       {
         \cs_if_exist:cTF
          {
            c_datatool_
            \l__datatool_GI_timevariant_tl
            _hhmmss_time_regex
          }
          {
            \use:c
             {
               datatool_
              \l__datatool_GI_datestyle_tl
              _hhmmss_tz_parse_timestamp:ccNnTF
             }
             {
               c_datatool_
               \l__datatool_GI_datevariant_tl
               _
               \l__datatool_GI_datestyle_tl
               _date_regex
             }
             {
               c_datatool_
               \l__datatool_GI_timevariant_tl
               _hhmmss_time_regex
             }
              #1 { #2 } { #3 } { #4 }
          }
          {
            \datatool_warn_check_head_language_empty:Vnnn
             \l__datatool_GI_timevariant_tl
             { datatool-GI }
             {
               No ~ language ~ support ~ for ~ time ~ variant ~
               ` \exp_args:Ne \tl_tail:n { \l__datatool_GI_timevariant_tl } '
             }
             {
              No ~ support ~ for ~ time ~ variant ~
              ` \l__datatool_GI_timevariant_tl '
             }
            #4
          }
       }
       {
         \datatool_warn_check_head_language_empty:Vnnn
           \l__datatool_GI_datevariant_tl
           { datatool-GI }
           {
             No ~ language ~ support ~ for ~ date ~ style ~
              ` \l__datatool_GI_datestyle_tl '
           }
           {
             No ~ support ~ for ~ date ~ style ~
             ` \l__datatool_GI_datestyle_tl ' ~ with ~
             variant ~
             ` \l__datatool_GI_datevariant_tl '
           }
         #4
       }
    }
    {
      \datatool_locale_warn:nn { datatool-GI }
       {
          No ~ support ~ for ~ date ~ style ~
          ` \l__datatool_GI_datestyle_tl '
       }
      #4
    }
 }
```

Date Parsing

Use command `\datatool_`⟨*style*⟩`_parse_date:NNnTF` with regular expression

`\c_datatool_`⟨*variant*⟩`_anchored_`⟨*style*⟩`_date_regex`

```
\cs_new:Nn \datatool_GI_parse_date:NnTF
 {
   \cs_if_exist:cTF
     {
       datatool_
       \l__datatool_GI_datestyle_tl
       _parse_date:NNnTF
     }
    {
      \cs_if_exist:cTF
       {
         c_datatool_
         \l__datatool_GI_datevariant_tl
         _anchored_
         \l__datatool_GI_datestyle_tl
         _date_regex
       }
       {
         \exp_args:cc
         {
           datatool_
           \l__datatool_GI_datestyle_tl
           _parse_date:NNnTF
         }
         {
           c_datatool_
           \l__datatool_GI_datevariant_tl
           _anchored_
           \l__datatool_GI_datestyle_tl
           _date_regex
         }
          #1 { #2 } { #3 } { #4 }
       }
       {
         \datatool_warn_check_head_language_empty:Vnnn
          \l__datatool_GI_datevariant_tl
          { datatool-GI }
          {
            No ~ language ~ support ~ for ~ date ~ style ~
             ` \l__datatool_GI_datestyle_tl '
          }
          {
             No ~ support ~ for ~ date ~ style ~
             ` \l__datatool_GI_datestyle_tl ' ~ with ~
             variant ~
             ` \l__datatool_GI_datevariant_tl '
          }
         #4
       }
    }
    {
      \datatool_locale_warn:nn { datatool-GI }
       {
          No ~ support ~ for ~ date ~ style ~
          ` \l__datatool_GI_datestyle_tl '
       }
      #4
    }
 }
```

Time Parsing

Use command `\datatool_`⟨*style*⟩`_parse_time:NNnTF` with regular expression

\c_datatool_⟨*variant*⟩_anchored_⟨*style*⟩_time_regex

```
\cs_new:Nn \datatool_GI_parse_time:NnTF
 {
   \cs_if_exist:cTF
    {
      c_datatool_
      \l__datatool_GI_timevariant_tl
      _anchored_hhmmss_time_regex
    }
    {
      \datatool_hhmmss_parse_time:cNnTF
       {
        c_datatool_
        \l__datatool_GI_timevariant_tl
        _anchored_hhmmss_time_regex
       }
        #1 { #2 } { #3 } { #4 }
    }
    {
      \datatool_warn_check_head_language_empty:Vnnn
        \l__datatool_GI_timevariant_tl
        { datatool-GI }
        {
           No ~ language ~ support ~ for ~ time ~ variant ~
           ` \exp_args:Ne \tl_tail:n { \l__datatool_GI_timevariant_tl } '
        }
        {
           No ~ support ~ for ~ time ~ variant ~
           ` \l__datatool_GI_timevariant_tl '
        }
      #4
    }
 }
```

Time Zone Mappings

Define time zone mapping command for this region:

```
\cs_new:Nn \datatool_GI_get_timezone_map:n
 {
   \datatool_region_get_timezone_map:n { GI / #1 }
 }
```

Define time zone IDs for this region (one-way map from ID to offset):

```
\datatool_region_set_timezone_map:nn { GI / CEST } { +02:00 }
\datatool_region_set_timezone_map:nn { GI / BST } { +01:00 }
\datatool_region_set_timezone_map:nn { GI / CET } { +01:00 }
\datatool_region_set_timezone_map:nn { GI / GMT } { +00:00 }
```

3.7.3 Options

Define options for this region:

```
\datatool_locale_define_keys:nn { GI }
 {
   number-style .choices:nn =
    { official, unofficial }
    {
      \exp_args:NNe \renewcommand
        \datatoolGISetNumberChars
         {
           \exp_not:N \bool_if:NT
            \exp_not:N \l_datatool_region_set_numberchars_bool
             {
               \exp_not:c { datatool_GI_set_numberchars_ \l_keys_choice_tl : }
             }
         }
      \tl_if_eq:NnT \l_datatool_current_region_tl { GI }
       {
         \datatoolGISetNumberChars
       }
    } ,
```

Currency symbol before or after value:

```
   currency-symbol-position .choice: ,
   currency-symbol-position / before .code:n =
    {
      \cs_set_eq:NN \datatool_GI_currency_position:nn \dtlcurrprefixfmt
    } ,
   currency-symbol-position / after .code:n =
    {
      \cs_set_eq:NN \datatool_GI_currency_position:nn \dtlcurrsuffixfmt
    } ,
```

Should the currency symbol be prefixed with the region code:

```
   currency-symbol-prefix .choice: ,
   currency-symbol-prefix / false .code:n =
    {
      \cs_set_eq:NN \datatoolGIsymbolprefix \use_none:n
    } ,
   currency-symbol-prefix / true .code:n =
    {
      \cs_set_eq:NN
       \datatoolGIsymbolprefix
       \datatool_currency_symbol_region_prefix:n
    } ,
   currency-symbol-prefix .default:n = { true } ,
```

Separator between currency symbol (not code) and value:

```
   currency-symbol-sep .choice: ,
   currency-symbol-sep / none .code:n =
    {
      \tl_clear:N \l_datatool_GI_sym_sep_tl
    } ,
   currency-symbol-sep / thin-space .code:n =
    {
      \tl_set:Nn \l_datatool_GI_sym_sep_tl { \, }
    } ,
   currency-symbol-sep / space .code:n =
    {
      \tl_set:Nn \l_datatool_GI_sym_sep_tl { ~ }
    } ,
   currency-symbol-sep / nbsp .code:n =
    {
      \tl_set:Nn \l_datatool_GI_sym_sep_tl { \nobreakspace }
    } ,
   currency-symbol-sep . initial:n = { none } ,
```

Date and time styles:

```
date-style .choice: ,
date-style / dmyyyy .code: n =
 {
   \tl_set:Nn \l__datatool_GI_datestyle_tl { ddmmyyyy }
 } ,
date-style / mdyyyy .code: n =
 {
   \tl_set:Nn \l__datatool_GI_datestyle_tl { mmddyyyy }
 } ,
date-style / yyyymd .code: n =
 {
   \tl_set:Nn \l__datatool_GI_datestyle_tl { yyyymmdd }
 } ,
date-style / dmyy .code: n =
 {
   \tl_set:Nn \l__datatool_GI_datestyle_tl { ddmmyy }
 } ,
date-style / mdyy .code: n =
 {
   \tl_set:Nn \l__datatool_GI_datestyle_tl { mmddyy }
 } ,
date-style / yymd .code: n =
 {
   \tl_set:Nn \l__datatool_GI_datestyle_tl { yymmdd }
 } ,
date-variant .choice: ,
date-variant / slash .code:n =
 {
   \tl_set:Nn \l__datatool_GI_datevariant_tl { slash }
 } ,
date-variant / hyphen .code:n =
 {
   \tl_set:Nn \l__datatool_GI_datevariant_tl { hyphen }
 } ,
date-variant / dot .code:n =
 {
   \tl_set:Nn \l__datatool_GI_datevariant_tl { dot }
 } ,
date-variant / dialect .code:n =
     {
       \tl_set:Nn \l__datatool_GI_datevariant_tl
         { \l_datatool_current_language_tl }
     } ,
    time-variant .choice: ,
    time-variant / colon .code:n =
     {
       \tl_set:Nn \l__datatool_GI_timevariant_tl { colon }
     } ,
    time-variant / dot .code:n =
     {
       \tl_set:Nn \l__datatool_GI_timevariant_tl { dot }
     } ,
    time-variant / dialect .code:n =
     {
       \tl_if_eq:NnTF \l__datatool_GI_timevariant_tl { dot }
        {
          \tl_set:Nn \l__datatool_GI_timevariant_tl
           { \l_datatool_current_language_tl _dot }
        }
        {
          \tl_if_in:NnF
            \l__datatool_GI_timevariant_tl
             { \l_datatool_current_language_tl }
           {
             \tl_set:Nn \l__datatool_GI_timevariant_tl
              { \l_datatool_current_language_tl _colon }
           }
        }
     } ,
    time-variant / dialect-colon .code:n =
     {
       \tl_set:Nn \l__datatool_GI_timevariant_tl
         { \l_datatool_current_language_tl _colon }
     } ,
    time-variant / dialect-dot .code:n =
     {
       \tl_set:Nn \l__datatool_GI_timevariant_tl
         { \l_datatool_current_language_tl _dot }
     } ,
  }
```

3.7.4 Hooks

Command to update temporal parsing commands for this region:

```
\newcommand \datatoolGISetTemporalParsers
 {
   \renewcommand \DTLCurrentLocaleParseTimeStamp
    { \datatool_GI_parse_timestamp:NnTF }
   \renewcommand \DTLCurrentLocaleParseDate
    { \datatool_GI_parse_date:NnTF }
   \renewcommand \DTLCurrentLocaleParseTime
```

```
    { \datatool_GI_parse_time:NnTF }
   \let
    \DTLCurrentLocaleGetTimeZoneMap
    \datatool_GI_get_timezone_map:n
 }
```

Set temporal formatting commands for this region. Currently this just resets to the default, but may change in future. Note that the defaults test if the applicable datetime2 command is available and will fallback on ISO if not defined. Bear in mind that the default style for datetime2 is `iso` so there won't be a noticeable difference unless the datetime2 regional setting is on.

```
\newcommand \datatoolGISetTemporalFormatters
 {
   \let
    \DTLCurrentLocaleFormatDate
    \datatool_default_date_fmt:nnnn
   \let
    \DTLCurrentLocaleFormatTime
    \datatool_default_time_fmt:nnn
   \let
    \DTLCurrentLocaleFormatTimeZone
    \datatool_default_timezone_fmt:nn
   \let
    \DTLCurrentLocaleFormatTimeStampNoZone
    \datatool_default_timestamp_fmt:nnnnnnn
   \let
    \DTLCurrentLocaleFormatTimeStampWithZone
    \datatool_default_timestamp_fmt:nnnnnnnnn
   \renewcommand \DTLCurrentLocaleTimeStampFmtSep { ~ }
 }
```

Command to update currency and temporal parsing commands for this region:

```
\newcommand \DTLGILocaleHook
 {
  \datatoolGISetNumberChars
  \datatoolGISetCurrency
  \datatoolGISetTemporalParsers
  \datatoolGISetTemporalFormatters
```

Allow language files to reference the region:

```
  \tl_set:Nn \l_datatool_current_region_tl { GI }
 }
```

Finished with LaTeX3 syntax.

```
\ExplSyntaxOff
```

Note that the hook is added to the captions by datatool-base not by the region file.

3.8 datatool-IE.ldf

Support for region IE.

```
\TrackLangProvidesResource{IE}[2025/03/01 v1.0]
```

Switch on LaTeX3 syntax:

```
\ExplSyntaxOn
```

3.8.1 Numbers and Currency

Set the number group and decimal symbols for this region.

```
\cs_new:Nn \datatool_IE_set_numberchars_official:
 {
  \datatool_set_numberchars:nn { , } { . }
 }
```

Unofficial style.

```
\cs_new:Nn \datatool_IE_set_numberchars_unofficial:
 {
  \datatool_set_thinspace_group_decimal_char:n { . }
 }
```

Hook to set the number group and decimal characters for the region:

```
\newcommand \datatoolIESetNumberChars
 {
   \bool_if:NT \l_datatool_region_set_numberchars_bool
    {
      \datatool_IE_set_numberchars_official:
    }
 }
```

The EUR currency is already defined by datatool-base, but we need to register the currency code with this region:

```
\datatool_register_regional_currency_code:nn { IE } { EUR }
```

How to format the position of the currency symbol in relation to the value.

```
\cs_new:Nn \datatool_IE_currency_position:nn
 {
   \dtlcurrprefixfmt { #1 } { #2 }
 }
```

Separator between currency symbol and value.

```
\tl_new:N \l_datatool_IE_sym_sep_tl
```

Provide a command to set the currency for this region (for use with any hook used when the locale changes).

```
\newcommand \datatoolIESetCurrency
 {
   \bool_if:NT \l_datatool_region_set_currency_bool
    {
     \DTLsetdefaultcurrency { EUR }
```

Number of digits that `\DTLdecimaltocurrency` should round to:

```
     \renewcommand \DTLCurrentLocaleCurrencyDP { 2 }
```

Update EUR format to reflect current region style:

```
     \renewcommand \DTLdefaultEURcurrencyfmt
       { \datatool_IE_currency_position:nn }
```

Separator between symbol and value:

```
      \renewcommand \dtlcurrfmtsymsep { \l_datatool_IE_sym_sep_tl }
     }
  }
```

3.8.2 Date and Time Parsing

This defaults to day/month/year but may be changed with `\DTLsetLocaleOptions`. For example:

```
\DTLsetLocaleOptions{IE}{date-style=dmyyyy, date-variant=slash}
```

An appropriate language file will need to also be installed to parse dates containing month names or day of week names.

Provide a way to configure parsing style.

```
\tl_new:N \l__datatool_IE_datevariant_tl
\tl_set:Nn \l__datatool_IE_datevariant_tl { slash }
```

NB These token lists are used to form command names. The following is not a format string.

```
\tl_new:N \l__datatool_IE_datestyle_tl
\tl_set:Nn \l__datatool_IE_datestyle_tl { ddmmyyyy }
\tl_new:N \l__datatool_IE_timevariant_tl
\tl_set:Nn \l__datatool_IE_timevariant_tl { colon }
```

Each parsing command defined below has final {⟨*true*⟩} and {⟨*false*⟩} arguments (TF). These are used if parsing was successful (true) or if parsing failed (false). The internal commands used by datatool-base have no need for solo branches (only T or only F) so these commands are simply defined with `\cs_new:Nn` not as conditionals.

Time Stamp Parsing

Use command

\datatool_⟨*date-style*⟩_hhmmss_tz_parse_timestamp:nnNnTF

with date regular expression

\c_datatool_⟨*date-variant*⟩_⟨*date-style*⟩_date_regex

and time regular expression

\c_datatool_⟨*time-variant*⟩_hhmmss_time_regex

```
\cs_new:Nn \datatool_IE_parse_timestamp:NnTF
```

```
{
  \cs_if_exist:cTF
    {
      datatool_
      \l__datatool_IE_datestyle_tl
      _hhmmss_tz_parse_timestamp:ccNnTF
    }
   {
     \cs_if_exist:cTF
      {
         c_datatool_
         \l__datatool_IE_datevariant_tl
         _
         \l__datatool_IE_datestyle_tl
         _date_regex
      }
      {
        \cs_if_exist:cTF
         {
           c_datatool_
           \l__datatool_IE_timevariant_tl
           _hhmmss_time_regex
         }
         {
           \use:c
            {
              datatool_
             \l__datatool_IE_datestyle_tl
             _hhmmss_tz_parse_timestamp:ccNnTF
            }
            {
              c_datatool_
              \l__datatool_IE_datevariant_tl
              _
              \l__datatool_IE_datestyle_tl
              _date_regex
            }
            {
              c_datatool_
              \l__datatool_IE_timevariant_tl
              _hhmmss_time_regex
            }
             #1 { #2 } { #3 } { #4 }
         }
         {
           \datatool_warn_check_head_language_empty:Vnnn
            \l__datatool_IE_timevariant_tl
            { datatool-IE }
            {
              No ~ language ~ support ~ for ~ time ~ variant ~
              ` \exp_args:Ne \tl_tail:n { \l__datatool_IE_timevariant_tl } '
            }
            {
             No ~ support ~ for ~ time ~ variant ~
             ` \l__datatool_IE_timevariant_tl '
            }
           #4
         }
      }
      {
        \datatool_warn_check_head_language_empty:Vnnn
          \l__datatool_IE_datevariant_tl
          { datatool-IE }
          {
            No ~ language ~ support ~ for ~ date ~ style ~
             ` \l__datatool_IE_datestyle_tl '
          }
          {
            No ~ support ~ for ~ date ~ style ~
            ` \l__datatool_IE_datestyle_tl ' ~ with ~
            variant ~
            ` \l__datatool_IE_datevariant_tl '
          }
        #4
      }
   }
   {
     \datatool_locale_warn:nn { datatool-IE }
      {
         No ~ support ~ for ~ date ~ style ~
         ` \l__datatool_IE_datestyle_tl '
      }
     #4
   }
}
```

Date Parsing

Use command `\datatool_`⟨*style*⟩`_parse_date:NNnTF` with regular expression

`\c_datatool_`⟨*variant*⟩`_anchored_`⟨*style*⟩`_date_regex`

```
\cs_new:Nn \datatool_IE_parse_date:NnTF
 {
   \cs_if_exist:cTF
     {
       datatool_
       \l__datatool_IE_datestyle_tl
       _parse_date:NNnTF
     }
    {
      \cs_if_exist:cTF
       {
         c_datatool_
```

```
        \l__datatool_IE_datevariant_tl
        _anchored_
        \l__datatool_IE_datestyle_tl
        _date_regex
      }
      {
        \exp_args:cc
        {
          datatool_
          \l__datatool_IE_datestyle_tl
          _parse_date:NNnTF
        }
        {
          c_datatool_
          \l__datatool_IE_datevariant_tl
          _anchored_
          \l__datatool_IE_datestyle_tl
          _date_regex
        }
         #1 { #2 } { #3 } { #4 }
      }
      {
        \datatool_warn_check_head_language_empty:Vnnn
         \l__datatool_IE_datevariant_tl
         { datatool-IE }
         {
           No ~ language ~ support ~ for ~ date ~ style ~
            ` \l__datatool_IE_datestyle_tl '
         }
         {
            No ~ support ~ for ~ date ~ style ~
            ` \l__datatool_IE_datestyle_tl ' ~ with ~
            variant ~
            ` \l__datatool_IE_datevariant_tl '
         }
        #4
      }
   }
   {
     \datatool_locale_warn:nn { datatool-IE }
      {
         No ~ support ~ for ~ date ~ style ~
         ` \l__datatool_IE_datestyle_tl '
      }
     #4
   }
}
```

Time Parsing

Use command `\datatool_`⟨*style*⟩`_parse_time:NNnTF` with regular expression

\c_datatool_⟨*variant*⟩_anchored_⟨*style*⟩_time_regex

```
\cs_new:Nn \datatool_IE_parse_time:NnTF
 {
   \cs_if_exist:cTF
    {
      c_datatool_
      \l__datatool_IE_timevariant_tl
      _anchored_hhmmss_time_regex
    }
    {
      \datatool_hhmmss_parse_time:cNnTF
       {
        c_datatool_
        \l__datatool_IE_timevariant_tl
        _anchored_hhmmss_time_regex
       }
        #1 { #2 } { #3 } { #4 }
    }
    {
      \datatool_warn_check_head_language_empty:Vnnn
        \l__datatool_IE_timevariant_tl
        { datatool-IE }
        {
           No ~ language ~ support ~ for ~ time ~ variant ~
           ` \exp_args:Ne \tl_tail:n { \l__datatool_IE_timevariant_tl } '
        }
        {
           No ~ support ~ for ~ time ~ variant ~
           ` \l__datatool_IE_timevariant_tl '
        }
      #4
    }
 }
```

Time Zone Mappings

Define time zone mapping command for this region:

```
\cs_new:Nn \datatool_IE_get_timezone_map:n
 {
   \datatool_region_get_timezone_map:n { IE / #1 }
 }
```

Define time zone IDs for this region (one-way map from ID to offset):

```
\datatool_region_set_timezone_map:nn { IE / IST } { +01:00 }
\datatool_region_set_timezone_map:nn { IE / GMT } { +00:00 }
\datatool_region_set_timezone_map:nn { IE / CEST } { +02:00 }
\datatool_region_set_timezone_map:nn { IE / CET } { +01:00 }
```

3.8.3 Options

Define options for this region:

```
\datatool_locale_define_keys:nn { IE }
 {
   number-style .choices:nn =
    { official, unofficial }
    {
      \exp_args:NNe \renewcommand
        \datatoolIESetNumberChars
         {
           \exp_not:N \bool_if:NT
            \exp_not:N \l_datatool_region_set_numberchars_bool
             {
               \exp_not:c { datatool_IE_set_numberchars_ \l_keys_choice_tl : }
             }
         }
      \tl_if_eq:NnT \l_datatool_current_region_tl { IE }
       {
         \datatoolIESetNumberChars
       }
    } ,
```

Currency symbol before or after value:

```
   currency-symbol-position .choice: ,
   currency-symbol-position / before .code:n =
    {
      \cs_set_eq:NN \datatool_IE_currency_position:nn \dtlcurrprefixfmt
    } ,
   currency-symbol-position / after .code:n =
    {
      \cs_set_eq:NN \datatool_IE_currency_position:nn \dtlcurrsuffixfmt
    } ,
```

Separator between currency symbol (not code) and value:

```
   currency-symbol-sep .choice: ,
   currency-symbol-sep / none .code:n =
    {
      \tl_clear:N \l_datatool_IE_sym_sep_tl
    } ,
   currency-symbol-sep / thin-space .code:n =
    {
      \tl_set:Nn \l_datatool_IE_sym_sep_tl { \, }
    } ,
   currency-symbol-sep / space .code:n =
    {
      \tl_set:Nn \l_datatool_IE_sym_sep_tl { ~ }
    } ,
   currency-symbol-sep / nbsp .code:n =
    {
      \tl_set:Nn \l_datatool_IE_sym_sep_tl { \nobreakspace }
    } ,
   currency-symbol-sep .initial:n = { none } ,
```

Date and time styles:

```
date-style .choice: ,
date-style / dmyyyy .code: n =
 {
   \tl_set:Nn \l__datatool_IE_datestyle_tl { ddmmyyyy }
 } ,
date-style / mdyyyy .code: n =
 {
   \tl_set:Nn \l__datatool_IE_datestyle_tl { mmddyyyy }
 } ,
date-style / yyyymd .code: n =
 {
   \tl_set:Nn \l__datatool_IE_datestyle_tl { yyyymmdd }
 } ,
date-style / dmyy .code: n =
 {
   \tl_set:Nn \l__datatool_IE_datestyle_tl { ddmmyy }
 } ,
date-style / mdyy .code: n =
 {
   \tl_set:Nn \l__datatool_IE_datestyle_tl { mmddyy }
 } ,
date-style / yymd .code: n =
 {
   \tl_set:Nn \l__datatool_IE_datestyle_tl { yymmdd }
 } ,
date-variant .choice: ,
date-variant / slash .code:n =
 {
   \tl_set:Nn \l__datatool_IE_datevariant_tl { slash }
 } ,
date-variant / hyphen .code:n =
 {
   \tl_set:Nn \l__datatool_IE_datevariant_tl { hyphen }
 } ,
date-variant / dot .code:n =
 {
   \tl_set:Nn \l__datatool_IE_datevariant_tl { dot }
 } ,
date-variant / dialect .code:n =
 {
   \tl_set:Nn \l__datatool_IE_datevariant_tl
     { \l_datatool_current_language_tl }
 } ,
time-variant .choice: ,
time-variant / colon .code:n =
 {
   \tl_set:Nn \l__datatool_IE_timevariant_tl { colon }
 } ,
time-variant / dot .code:n =
 {
   \tl_set:Nn \l__datatool_IE_timevariant_tl { dot }
 } ,
    time-variant / dialect .code:n =
     {
       \tl_if_eq:NnTF \l__datatool_IE_timevariant_tl { dot }
        {
          \tl_set:Nn \l__datatool_IE_timevariant_tl
           { \l_datatool_current_language_tl _dot }
        }
        {
          \tl_if_in:NnF
            \l__datatool_IE_timevariant_tl
             { \l_datatool_current_language_tl }
           {
             \tl_set:Nn \l__datatool_IE_timevariant_tl
              { \l_datatool_current_language_tl _colon }
           }
        }
     } ,
    time-variant / dialect-colon .code:n =
     {
       \tl_set:Nn \l__datatool_IE_timevariant_tl
         { \l_datatool_current_language_tl _colon }
     } ,
    time-variant / dialect-dot .code:n =
     {
       \tl_set:Nn \l__datatool_IE_timevariant_tl
         { \l_datatool_current_language_tl _dot }
     } ,
  }
```

3.8.4 Hooks

Command to update temporal parsing commands for this region:

```
\newcommand \datatoolIESetTemporalParsers
 {
   \renewcommand \DTLCurrentLocaleParseTimeStamp
    { \datatool_IE_parse_timestamp:NnTF }
   \renewcommand \DTLCurrentLocaleParseDate
    { \datatool_IE_parse_date:NnTF }
   \renewcommand \DTLCurrentLocaleParseTime
    { \datatool_IE_parse_time:NnTF }
   \let
    \DTLCurrentLocaleGetTimeZoneMap
    \datatool_IE_get_timezone_map:n
 }
```

Set temporal formatting commands for this region. Currently this just resets to the default, but may change in future. Note that the defaults test if the applicable datetime2 command is available and will fallback on ISO if not defined. Bear in mind that the default style for datetime2 is `iso` so there won't be a noticeable difference unless the datetime2 regional setting is on.

```
\newcommand \datatoolIESetTemporalFormatters
```

```
  {
    \let
     \DTLCurrentLocaleFormatDate
     \datatool_default_date_fmt:nnnn
    \let
     \DTLCurrentLocaleFormatTime
     \datatool_default_time_fmt:nnn
    \let
     \DTLCurrentLocaleFormatTimeZone
     \datatool_default_timezone_fmt:nn
    \let
     \DTLCurrentLocaleFormatTimeStampNoZone
     \datatool_default_timestamp_fmt:nnnnnnn
    \let
     \DTLCurrentLocaleFormatTimeStampWithZone
     \datatool_default_timestamp_fmt:nnnnnnnnn
    \renewcommand \DTLCurrentLocaleTimeStampFmtSep { ~ }
  }
```

Command to update currency and temporal parsing commands for this region:

```
\newcommand \DTLIELocaleHook
 {
  \datatoolIESetNumberChars
  \datatoolIESetCurrency
  \datatoolIESetTemporalParsers
  \datatoolIESetTemporalFormatters
```

Allow language files to reference the region:

```
  \tl_set:Nn \l_datatool_current_region_tl { IE }
 }
```

Finished with LaTeX3 syntax.

```
\ExplSyntaxOff
```

Note that the hook is added to the captions by datatool-base not by the region file.

3.9 datatool-IM.ldf

Support for region IM.

```
\TrackLangProvidesResource{IM}[2025/03/01 v1.0]
```

Switch on LaTeX3 syntax:

```
\ExplSyntaxOn
```

3.9.1 Numbers and Currency

Set the number group and decimal symbols for this region.

```
\cs_new:Nn \datatool_IM_set_numberchars_official:
 {
  \datatool_set_numberchars:nn { , } { . }
 }
```

Unofficial style.

```
\cs_new:Nn \datatool_IM_set_numberchars_unofficial:
 {
  \datatool_set_thinspace_group_decimal_char:n { . }
 }
```

Hook to set the number group and decimal characters for the region:

```
\newcommand \datatoolIMSetNumberChars
 {
   \bool_if:NT \l_datatool_region_set_numberchars_bool
    {
      \datatool_IM_set_numberchars_official:
    }
 }
```

How to format the position of the currency symbol in relation to the value.

```
\cs_new:Nn \datatool_IM_currency_position:nn
 {
   \dtlcurrprefixfmt { #1 } { #2 }
 }
```

Separator between currency symbol and value.

```
\tl_new:N \l_datatool_IM_sym_sep_tl
```

Set the currency format for this region.

```
\newcommand \datatoolIMcurrencyfmt [ 2 ]
 {
   \datatool_IM_currency_position:nn
    {
      \datatoolIMsymbolprefix { IM }
      #1
    }
    { #2 }
 }
```

Prefix for symbol, if required.

```
\newcommand \datatoolIMsymbolprefix [ 1 ] { }
```

Define the currency symbols for this region.

```
\datatool_def_currency:nnnV
 { \datatoolIMcurrencyfmt }
 { IMP }
 { \pounds }
 \l_datatool_pound_tl
```

Register the currency code with this region:

```
\datatool_register_regional_currency_code:nn { IM } { IMP }
```

Provide a command to set the currency for this region (for use with any hook used when the locale changes). NB this should do nothing with `\DTLsetup{region-currency=false}`

```
\newcommand \datatoolIMSetCurrency
 {
   \bool_if:NT \l_datatool_region_set_currency_bool
    {
      \DTLsetdefaultcurrency { IMP }
```

Number of digits that `\DTLdecimaltocurrency` should round to:

```
        \renewcommand \DTLCurrentLocaleCurrencyDP { 2 }
```

Separator between symbol and value:

```
        \renewcommand \dtlcurrfmtsymsep { \l_datatool_IM_sym_sep_tl }
      }
   }
```

3.9.2 Date and Time Parsing

This defaults to day/month/year but may be changed with `\DTLsetLocaleOptions`. For example:

```
\DTLsetLocaleOptions{IM}{date-style=dmyyyy, date-variant=slash}
```

An appropriate language file will need to also be installed to parse dates containing month names or day of week names.

Provide a way to configure parsing style.

```
\tl_new:N \l__datatool_IM_datevariant_tl
\tl_set:Nn \l__datatool_IM_datevariant_tl { slash }
```

NB These token lists are used to form command names. The following is not a format string.

```
\tl_new:N \l__datatool_IM_datestyle_tl
\tl_set:Nn \l__datatool_IM_datestyle_tl { ddmmyyyy }
\tl_new:N \l__datatool_IM_timevariant_tl
\tl_set:Nn \l__datatool_IM_timevariant_tl { colon }
```

Each parsing command defined below has final {⟨*true*⟩} and {⟨*false*⟩} arguments (TF). These are used if parsing was successful (true) or if parsing failed (false). The internal commands used by datatool-base have no need for solo branches (only T or only F) so these commands are simply defined with `\cs_new:Nn` not as conditionals.

Time Stamp Parsing

Use command

```
\datatool_⟨date-style⟩_hhmmss_tz_parse_timestamp:nnNnTF
```

with date regular expression

```
\c_datatool_⟨date-variant⟩_⟨date-style⟩_date_regex
```

and time regular expression

```
\c_datatool_⟨time-variant⟩_hhmmss_time_regex
```

```
\cs_new:Nn \datatool_IM_parse_timestamp:NnTF
 {
   \cs_if_exist:cTF
     {
       datatool_
       \l__datatool_IM_datestyle_tl
       _hhmmss_tz_parse_timestamp:ccNnTF
     }
    {
      \cs_if_exist:cTF
       {
          c_datatool_
          \l__datatool_IM_datevariant_tl
          _
          \l__datatool_IM_datestyle_tl
          _date_regex
       }
       {
         \cs_if_exist:cTF
          {
            c_datatool_
            \l__datatool_IM_timevariant_tl
            _hhmmss_time_regex
          }
          {
            \use:c
             {
               datatool_
              \l__datatool_IM_datestyle_tl
              _hhmmss_tz_parse_timestamp:ccNnTF
             }
             {
               c_datatool_
               \l__datatool_IM_datevariant_tl
               _
               \l__datatool_IM_datestyle_tl
               _date_regex
             }
             {
               c_datatool_
               \l__datatool_IM_timevariant_tl
               _hhmmss_time_regex
             }
              #1 { #2 } { #3 } { #4 }
          }
          {
            \datatool_warn_check_head_language_empty:Vnnn
             \l__datatool_IM_timevariant_tl
             { datatool-IM }
             {
               No ~ language ~ support ~ for ~ time ~ variant ~
               ` \exp_args:Ne \tl_tail:n { \l__datatool_IM_timevariant_tl } '
             }
             {
              No ~ support ~ for ~ time ~ variant ~
              ` \l__datatool_IM_timevariant_tl '
             }
            #4
          }
       }
       {
         \datatool_warn_check_head_language_empty:Vnnn
           \l__datatool_IM_datevariant_tl
           { datatool-IM }
           {
             No ~ language ~ support ~ for ~ date ~ style ~
              ` \l__datatool_IM_datestyle_tl '
           }
           {
             No ~ support ~ for ~ date ~ style ~
             ` \l__datatool_IM_datestyle_tl ' ~ with ~
             variant ~
             ` \l__datatool_IM_datevariant_tl '
           }
         #4
       }
    }
    {
      \datatool_locale_warn:nn { datatool-IM }
       {
          No ~ support ~ for ~ date ~ style ~
          ` \l__datatool_IM_datestyle_tl '
       }
      #4
    }
 }
```

Date Parsing

Use command `\datatool_`⟨*style*⟩`_parse_date:NNnTF` with regular expression

`\c_datatool_`⟨*variant*⟩`_anchored_`⟨*style*⟩`_date_regex`

```
\cs_new:Nn \datatool_IM_parse_date:NnTF
 {
   \cs_if_exist:cTF
     {
       datatool_
       \l__datatool_IM_datestyle_tl
       _parse_date:NNnTF
     }
    {
      \cs_if_exist:cTF
       {
         c_datatool_
         \l__datatool_IM_datevariant_tl
         _anchored_
         \l__datatool_IM_datestyle_tl
         _date_regex
       }
       {
         \exp_args:cc
         {
           datatool_
           \l__datatool_IM_datestyle_tl
           _parse_date:NNnTF
         }
         {
           c_datatool_
           \l__datatool_IM_datevariant_tl
           _anchored_
           \l__datatool_IM_datestyle_tl
           _date_regex
         }
          #1 { #2 } { #3 } { #4 }
       }
       {
         \datatool_warn_check_head_language_empty:Vnnn
          \l__datatool_IM_datevariant_tl
          { datatool-IM }
          {
            No ~ language ~ support ~ for ~ date ~ style ~
             ` \l__datatool_IM_datestyle_tl '
          }
          {
             No ~ support ~ for ~ date ~ style ~
             ` \l__datatool_IM_datestyle_tl ' ~ with ~
             variant ~
             ` \l__datatool_IM_datevariant_tl '
          }
         #4
       }
    }
    {
      \datatool_locale_warn:nn { datatool-IM }
       {
          No ~ support ~ for ~ date ~ style ~
          ` \l__datatool_IM_datestyle_tl '
       }
      #4
    }
 }
```

Time Parsing

Use command `\datatool_`⟨*style*⟩`_parse_time:NNnTF` with regular expression

`\c_datatool_⟨variant⟩_anchored_⟨style⟩_time_regex`

```
\cs_new:Nn \datatool_IM_parse_time:NnTF
 {
   \cs_if_exist:cTF
    {
      c_datatool_
      \l__datatool_IM_timevariant_tl
      _anchored_hhmmss_time_regex
    }
    {
      \datatool_hhmmss_parse_time:cNnTF
       {
        c_datatool_
        \l__datatool_IM_timevariant_tl
        _anchored_hhmmss_time_regex
       }
        #1 { #2 } { #3 } { #4 }
    }
    {
      \datatool_warn_check_head_language_empty:Vnnn
        \l__datatool_IM_timevariant_tl
        { datatool-IM }
        {
           No ~ language ~ support ~ for ~ time ~ variant ~
           ` \exp_args:Ne \tl_tail:n { \l__datatool_IM_timevariant_tl } '
        }
        {
           No ~ support ~ for ~ time ~ variant ~
           ` \l__datatool_IM_timevariant_tl '
        }
      #4
    }
 }
```

Time Zone Mappings

Define time zone mapping command for this region:

```
\cs_new:Nn \datatool_IM_get_timezone_map:n
 {
   \datatool_region_get_timezone_map:n { IM / #1 }
 }
```

Define time zone IDs for this region (one-way map from ID to offset):

```
\datatool_region_set_timezone_map:nn { IM / CET } { +01:00 }
\datatool_region_set_timezone_map:nn { IM / GMT } { +00:00 }
\datatool_region_set_timezone_map:nn { IM / BST } { +01:00 }
\datatool_region_set_timezone_map:nn { IM / CEST } { +02:00 }
```

3.9.3 Options

Define options for this region:

```
\datatool_locale_define_keys:nn { IM }
 {
   number-style .choices:nn =
    { official, unofficial }
    {
      \exp_args:NNe \renewcommand
        \datatoolIMSetNumberChars
         {
           \exp_not:N \bool_if:NT
            \exp_not:N \l_datatool_region_set_numberchars_bool
             {
               \exp_not:c { datatool_IM_set_numberchars_ \l_keys_choice_tl : }
             }
         }
      \tl_if_eq:NnT \l_datatool_current_region_tl { IM }
       {
         \datatoolIMSetNumberChars
       }
    } ,
```

Currency symbol before or after value:

```
   currency-symbol-position .choice: ,
   currency-symbol-position / before .code:n =
    {
      \cs_set_eq:NN \datatool_IM_currency_position:nn \dtlcurrprefixfmt
    } ,
   currency-symbol-position / after .code:n =
    {
      \cs_set_eq:NN \datatool_IM_currency_position:nn \dtlcurrsuffixfmt
    } ,
```

Should the currency symbol be prefixed with the region code:

```
   currency-symbol-prefix .choice: ,
   currency-symbol-prefix / false .code:n =
    {
      \cs_set_eq:NN \datatoolIMsymbolprefix \use_none:n
    } ,
   currency-symbol-prefix / true .code:n =
    {
      \cs_set_eq:NN
       \datatoolIMsymbolprefix
       \datatool_currency_symbol_region_prefix:n
    } ,
   currency-symbol-prefix .default:n = { true } ,
```

Separator between currency symbol (not code) and value:

```
   currency-symbol-sep .choice: ,
   currency-symbol-sep / none .code:n =
    {
      \tl_clear:N \l_datatool_IM_sym_sep_tl
    } ,
   currency-symbol-sep / thin-space .code:n =
    {
      \tl_set:Nn \l_datatool_IM_sym_sep_tl { \, }
    } ,
   currency-symbol-sep / space .code:n =
    {
      \tl_set:Nn \l_datatool_IM_sym_sep_tl { ~ }
    } ,
   currency-symbol-sep / nbsp .code:n =
    {
      \tl_set:Nn \l_datatool_IM_sym_sep_tl { \nobreakspace }
    } ,
   currency-symbol-sep . initial:n = { none } ,
```

Date and time styles:

```
date-style .choice: ,
date-style / dmyyyy .code: n =
 {
   \tl_set:Nn \l__datatool_IM_datestyle_tl { ddmmyyyy }
 } ,
date-style / mdyyyy .code: n =
 {
   \tl_set:Nn \l__datatool_IM_datestyle_tl { mmddyyyy }
 } ,
date-style / yyyymd .code: n =
 {
   \tl_set:Nn \l__datatool_IM_datestyle_tl { yyyymmdd }
 } ,
date-style / dmyy .code: n =
 {
   \tl_set:Nn \l__datatool_IM_datestyle_tl { ddmmyy }
 } ,
date-style / mdyy .code: n =
 {
   \tl_set:Nn \l__datatool_IM_datestyle_tl { mmddyy }
 } ,
date-style / yymd .code: n =
 {
   \tl_set:Nn \l__datatool_IM_datestyle_tl { yymmdd }
 } ,
date-variant .choice: ,
date-variant / slash .code:n =
 {
   \tl_set:Nn \l__datatool_IM_datevariant_tl { slash }
 } ,
date-variant / hyphen .code:n =
 {
   \tl_set:Nn \l__datatool_IM_datevariant_tl { hyphen }
 } ,
date-variant / dot .code:n =
 {
   \tl_set:Nn \l__datatool_IM_datevariant_tl { dot }
 } ,
date-variant / dialect .code:n =
     {
       \tl_set:Nn \l__datatool_IM_datevariant_tl
         { \l_datatool_current_language_tl }
     } ,
    time-variant .choice: ,
    time-variant / colon .code:n =
     {
       \tl_set:Nn \l__datatool_IM_timevariant_tl { colon }
     } ,
    time-variant / dot .code:n =
     {
       \tl_set:Nn \l__datatool_IM_timevariant_tl { dot }
     } ,
    time-variant / dialect .code:n =
     {
       \tl_if_eq:NnTF \l__datatool_IM_timevariant_tl { dot }
        {
          \tl_set:Nn \l__datatool_IM_timevariant_tl
           { \l_datatool_current_language_tl _dot }
        }
        {
          \tl_if_in:NnF
            \l__datatool_IM_timevariant_tl
             { \l_datatool_current_language_tl }
           {
             \tl_set:Nn \l__datatool_IM_timevariant_tl
              { \l_datatool_current_language_tl _colon }
           }
        }
     } ,
    time-variant / dialect-colon .code:n =
     {
       \tl_set:Nn \l__datatool_IM_timevariant_tl
         { \l_datatool_current_language_tl _colon }
     } ,
    time-variant / dialect-dot .code:n =
     {
       \tl_set:Nn \l__datatool_IM_timevariant_tl
         { \l_datatool_current_language_tl _dot }
     } ,
  }
```

3.9.4 Hooks

Command to update temporal parsing commands for this region:

```
\newcommand \datatoolIMSetTemporalParsers
 {
   \renewcommand \DTLCurrentLocaleParseTimeStamp
    { \datatool_IM_parse_timestamp:NnTF }
   \renewcommand \DTLCurrentLocaleParseDate
    { \datatool_IM_parse_date:NnTF }
   \renewcommand \DTLCurrentLocaleParseTime
```

```
    { \datatool_IM_parse_time:NnTF }
   \let
    \DTLCurrentLocaleGetTimeZoneMap
    \datatool_IM_get_timezone_map:n
 }
```

Set temporal formatting commands for this region. Currently this just resets to the default, but may change in future. Note that the defaults test if the applicable datetime2 command is available and will fallback on ISO if not defined. Bear in mind that the default style for datetime2 is `iso` so there won't be a noticeable difference unless the datetime2 regional setting is on.

```
\newcommand \datatoolIMSetTemporalFormatters
 {
   \let
    \DTLCurrentLocaleFormatDate
    \datatool_default_date_fmt:nnnn
   \let
    \DTLCurrentLocaleFormatTime
    \datatool_default_time_fmt:nnn
   \let
    \DTLCurrentLocaleFormatTimeZone
    \datatool_default_timezone_fmt:nn
   \let
    \DTLCurrentLocaleFormatTimeStampNoZone
    \datatool_default_timestamp_fmt:nnnnnnn
   \let
    \DTLCurrentLocaleFormatTimeStampWithZone
    \datatool_default_timestamp_fmt:nnnnnnnnn
   \renewcommand \DTLCurrentLocaleTimeStampFmtSep { ~ }
 }
```

Command to update currency and temporal parsing commands for this region:

```
\newcommand \DTLIMLocaleHook
 {
  \datatoolIMSetNumberChars
  \datatoolIMSetCurrency
  \datatoolIMSetTemporalParsers
  \datatoolIMSetTemporalFormatters
```

Allow language files to reference the region:

```
  \tl_set:Nn \l_datatool_current_region_tl { IM }
 }
```

Finished with LaTeX3 syntax.

```
\ExplSyntaxOff
```

Note that the hook is added to the captions by datatool-base not by the region file.

3.10 datatool-JE.ldf

Support for region JE.

```
\TrackLangProvidesResource{JE}[2025/03/01 v1.0]
```

Switch on LaTeX3 syntax:

```
\ExplSyntaxOn
```

3.10.1 Numbers and Currency

Set the number group and decimal symbols for this region.

```
\cs_new:Nn \datatool_JE_set_numberchars_official:
 {
  \datatool_set_numberchars:nn { , } { . }
 }
```

Unofficial style.

```
\cs_new:Nn \datatool_JE_set_numberchars_unofficial:
 {
  \datatool_set_thinspace_group_decimal_char:n { . }
 }
```

Hook to set the number group and decimal characters for the region:

```
\newcommand \datatoolJESetNumberChars
 {
   \bool_if:NT \l_datatool_region_set_numberchars_bool
    {
      \datatool_JE_set_numberchars_official:
    }
 }
```

How to format the position of the currency symbol in relation to the value.

```
\cs_new:Nn \datatool_JE_currency_position:nn
 {
   \dtlcurrprefixfmt { #1 } { #2 }
 }
```

Separator between currency symbol and value.

```
\tl_new:N \l_datatool_JE_sym_sep_tl
```

Set the currency format for this region.

```
\newcommand \datatoolJEcurrencyfmt [ 2 ]
 {
   \datatool_JE_currency_position:nn
    {
      \datatoolJEsymbolprefix { JE }
      #1
    }
    { #2 }
 }
```

Prefix for symbol, if required.

```
\newcommand \datatoolJEsymbolprefix [ 1 ] { }
```

Define the currency symbols for this region.

```
\datatool_def_currency:nnnV
 { \datatoolJEcurrencyfmt }
 { JEP }
 { \pounds }
```

```
  \l_datatool_pound_tl
```

Register the currency code with this region:

```
\datatool_register_regional_currency_code:nn { JE } { JEP }
```

Provide a command to set the currency for this region (for use with any hook used when the locale changes). NB this should do nothing with `\DTLsetup{region-currency=false}`

```
\newcommand \datatoolJESetCurrency
 {
   \bool_if:NT \l_datatool_region_set_currency_bool
    {
      \DTLsetdefaultcurrency { JEP }
```

Number of digits that `\DTLdecimaltocurrency` should round to:

```
      \renewcommand \DTLCurrentLocaleCurrencyDP { 2 }
```

Separator between symbol and value:

```
      \renewcommand \dtlcurrfmtsymsep { \l_datatool_JE_sym_sep_tl }
    }
 }
```

3.10.2 Date and Time Parsing

This defaults to day/month/year but may be changed with `\DTLsetLocaleOptions`. For example:

```
\DTLsetLocaleOptions{JE}{date-style=dmyyyy, date-variant=slash}
```

An appropriate language file will need to also be installed to parse dates containing month names or day of week names.

Provide a way to configure parsing style.

```
\tl_new:N \l__datatool_JE_datevariant_tl
\tl_set:Nn \l__datatool_JE_datevariant_tl { slash }
```

NB These token lists are used to form command names. The following is not a format string.

```
\tl_new:N \l__datatool_JE_datestyle_tl
\tl_set:Nn \l__datatool_JE_datestyle_tl { ddmmyyyy }
\tl_new:N \l__datatool_JE_timevariant_tl
\tl_set:Nn \l__datatool_JE_timevariant_tl { colon }
```

Each parsing command defined below has final {⟨*true*⟩} and {⟨*false*⟩} arguments (TF). These are used if parsing was successful (true) or if parsing failed (false). The internal commands used by datatool-base have no need for solo branches (only T or only F) so these commands are simply defined with `\cs_new:Nn` not as conditionals.

Time Stamp Parsing

Use command

`\datatool_`⟨*date-style*⟩`_hhmmss_tz_parse_timestamp:nnNnTF`

with date regular expression

`\c_datatool_`⟨*date-variant*⟩`_`⟨*date-style*⟩`_date_regex`

and time regular expression

`\c_datatool_`⟨*time-variant*⟩`_hhmmss_time_regex`

```
\cs_new:Nn \datatool_JE_parse_timestamp:NnTF
 {
   \cs_if_exist:cTF
     {
       datatool_
       \l__datatool_JE_datestyle_tl
       _hhmmss_tz_parse_timestamp:ccNnTF
     }
    {
      \cs_if_exist:cTF
       {
          c_datatool_
          \l__datatool_JE_datevariant_tl
          _
          \l__datatool_JE_datestyle_tl
          _date_regex
       }
       {
         \cs_if_exist:cTF
          {
            c_datatool_
            \l__datatool_JE_timevariant_tl
            _hhmmss_time_regex
          }
          {
            \use:c
             {
               datatool_
              \l__datatool_JE_datestyle_tl
              _hhmmss_tz_parse_timestamp:ccNnTF
             }
             {
               c_datatool_
               \l__datatool_JE_datevariant_tl
               _
               \l__datatool_JE_datestyle_tl
               _date_regex
             }
             {
               c_datatool_
```

```
                \l__datatool_JE_timevariant_tl
                _hhmmss_time_regex
              }
               #1 { #2 } { #3 } { #4 }
           }
           {
             \datatool_warn_check_head_language_empty:Vnnn
              \l__datatool_JE_timevariant_tl
              { datatool-JE }
              {
                No ~ language ~ support ~ for ~ time ~ variant ~
                ` \exp_args:Ne \tl_tail:n { \l__datatool_JE_timevariant_tl } '
              }
              {
               No ~ support ~ for ~ time ~ variant ~
               ` \l__datatool_JE_timevariant_tl '
              }
             #4
           }
        }
        {
          \datatool_warn_check_head_language_empty:Vnnn
            \l__datatool_JE_datevariant_tl
            { datatool-JE }
            {
              No ~ language ~ support ~ for ~ date ~ style ~
               ` \l__datatool_JE_datestyle_tl '
            }
            {
              No ~ support ~ for ~ date ~ style ~
              ` \l__datatool_JE_datestyle_tl ' ~ with ~
              variant ~
              ` \l__datatool_JE_datevariant_tl '
            }
          #4
        }
     }
     {
       \datatool_locale_warn:nn { datatool-JE }
        {
           No ~ support ~ for ~ date ~ style ~
           ` \l__datatool_JE_datestyle_tl '
        }
       #4
     }
  }
```

Date Parsing

Use command `\datatool_`⟨*style*⟩`_parse_date:NNnTF` with regular expression

\c_datatool_⟨*variant*⟩_anchored_⟨*style*⟩_date_regex

```
\cs_new:Nn \datatool_JE_parse_date:NnTF
 {
   \cs_if_exist:cTF
     {
       datatool_
       \l__datatool_JE_datestyle_tl
       _parse_date:NNnTF
     }
    {
      \cs_if_exist:cTF
       {
         c_datatool_
         \l__datatool_JE_datevariant_tl
         _anchored_
         \l__datatool_JE_datestyle_tl
         _date_regex
       }
       {
         \exp_args:cc
         {
           datatool_
           \l__datatool_JE_datestyle_tl
           _parse_date:NNnTF
         }
         {
           c_datatool_
           \l__datatool_JE_datevariant_tl
           _anchored_
           \l__datatool_JE_datestyle_tl
           _date_regex
         }
          #1 { #2 } { #3 } { #4 }
       }
       {
         \datatool_warn_check_head_language_empty:Vnnn
          \l__datatool_JE_datevariant_tl
          { datatool-JE }
          {
            No ~ language ~ support ~ for ~ date ~ style ~
             ` \l__datatool_JE_datestyle_tl '
          }
          {
             No ~ support ~ for ~ date ~ style ~
             ` \l__datatool_JE_datestyle_tl ' ~ with ~
             variant ~
             ` \l__datatool_JE_datevariant_tl '
          }
         #4
       }
    }
    {
      \datatool_locale_warn:nn { datatool-JE }
       {
          No ~ support ~ for ~ date ~ style ~
          ` \l__datatool_JE_datestyle_tl '
       }
      #4
    }
 }
```

Time Parsing

Use command `\datatool_`⟨*style*⟩`_parse_time:NNnTF` with regular expression

`\c_datatool_`⟨*variant*⟩`_anchored_`⟨*style*⟩`_time_regex`

```
\cs_new:Nn \datatool_JE_parse_time:NnTF
 {
   \cs_if_exist:cTF
    {
      c_datatool_
      \l__datatool_JE_timevariant_tl
      _anchored_hhmmss_time_regex
    }
    {
      \datatool_hhmmss_parse_time:cNnTF
       {
        c_datatool_
        \l__datatool_JE_timevariant_tl
        _anchored_hhmmss_time_regex
       }
        #1 { #2 } { #3 } { #4 }
    }
    {
      \datatool_warn_check_head_language_empty:Vnnn
        \l__datatool_JE_timevariant_tl
        { datatool-JE }
        {
           No ~ language ~ support ~ for ~ time ~ variant ~
           ` \exp_args:Ne \tl_tail:n { \l__datatool_JE_timevariant_tl } '
        }
        {
           No ~ support ~ for ~ time ~ variant ~
           ` \l__datatool_JE_timevariant_tl '
        }
      #4
    }
 }
```

Time Zone Mappings

Define time zone mapping command for this region:

```
\cs_new:Nn \datatool_JE_get_timezone_map:n
 {
   \datatool_region_get_timezone_map:n { JE / #1 }
 }
```

Define time zone IDs for this region (one-way map from ID to offset):

```
\datatool_region_set_timezone_map:nn { JE / CEST } { +02:00 }
\datatool_region_set_timezone_map:nn { JE / GMT } { +00:00 }
\datatool_region_set_timezone_map:nn { JE / CET } { +01:00 }
\datatool_region_set_timezone_map:nn { JE / BST } { +01:00 }
```

3.10.3 Options

Define options for this region:

```
\datatool_locale_define_keys:nn { JE }
 {
   number-style .choices:nn =
    { official, unofficial }
    {
      \exp_args:NNe \renewcommand
        \datatoolJESetNumberChars
         {
           \exp_not:N \bool_if:NT
            \exp_not:N \l_datatool_region_set_numberchars_bool
             {
               \exp_not:c { datatool_JE_set_numberchars_ \l_keys_choice_tl : }
             }
         }
      \tl_if_eq:NnT \l_datatool_current_region_tl { JE }
       {
         \datatoolJESetNumberChars
       }
    } ,
```

Currency symbol before or after value:

```
   currency-symbol-position .choice: ,
   currency-symbol-position / before .code:n =
    {
      \cs_set_eq:NN \datatool_JE_currency_position:nn \dtlcurrprefixfmt
    } ,
   currency-symbol-position / after .code:n =
    {
      \cs_set_eq:NN \datatool_JE_currency_position:nn \dtlcurrsuffixfmt
    } ,
```

Should the currency symbol be prefixed with the region code:

```
   currency-symbol-prefix .choice: ,
   currency-symbol-prefix / false .code:n =
    {
      \cs_set_eq:NN \datatoolJEsymbolprefix \use_none:n
```

```
 } ,
currency-symbol-prefix / true .code:n =
 {
   \cs_set_eq:NN
    \datatoolJEsymbolprefix
    \datatool_currency_symbol_region_prefix:n
 } ,
currency-symbol-prefix .default:n = { true } ,
```

Separator between currency symbol (not code) and value:

```
currency-symbol-sep .choice: ,
currency-symbol-sep / none .code:n =
 {
   \tl_clear:N \l_datatool_JE_sym_sep_tl
 } ,
currency-symbol-sep / thin-space .code:n =
 {
   \tl_set:Nn \l_datatool_JE_sym_sep_tl { \, }
 } ,
currency-symbol-sep / space .code:n =
 {
   \tl_set:Nn \l_datatool_JE_sym_sep_tl { ~ }
 } ,
currency-symbol-sep / nbsp .code:n =
 {
   \tl_set:Nn \l_datatool_JE_sym_sep_tl { \nobreakspace }
 } ,
currency-symbol-sep . initial:n = { none } ,
```

Date and time styles:

```
date-style .choice: ,
date-style / dmyyyy .code: n =
 {
   \tl_set:Nn \l__datatool_JE_datestyle_tl { ddmmyyyy }
 } ,
date-style / mdyyyy .code: n =
 {
   \tl_set:Nn \l__datatool_JE_datestyle_tl { mmddyyyy }
 } ,
date-style / yyyymd .code: n =
 {
   \tl_set:Nn \l__datatool_JE_datestyle_tl { yyyymmdd }
 } ,
date-style / dmyy .code: n =
 {
   \tl_set:Nn \l__datatool_JE_datestyle_tl { ddmmyy }
 } ,
date-style / mdyy .code: n =
 {
   \tl_set:Nn \l__datatool_JE_datestyle_tl { mmddyy }
 } ,
date-style / yymd .code: n =
 {
   \tl_set:Nn \l__datatool_JE_datestyle_tl { yymmdd }
 } ,
date-variant .choice: ,
date-variant / slash .code:n =
 {
   \tl_set:Nn \l__datatool_JE_datevariant_tl { slash }
 } ,
date-variant / hyphen .code:n =
 {
   \tl_set:Nn \l__datatool_JE_datevariant_tl { hyphen }
 } ,
date-variant / dot .code:n =
 {
   \tl_set:Nn \l__datatool_JE_datevariant_tl { dot }
 } ,
date-variant / dialect .code:n =
 {
   \tl_set:Nn \l__datatool_JE_datevariant_tl
     { \l_datatool_current_language_tl }
 } ,
time-variant .choice: ,
time-variant / colon .code:n =
 {
   \tl_set:Nn \l__datatool_JE_timevariant_tl { colon }
 } ,
time-variant / dot .code:n =
 {
   \tl_set:Nn \l__datatool_JE_timevariant_tl { dot }
 } ,
time-variant / dialect .code:n =
 {
   \tl_if_eq:NnTF \l__datatool_JE_timevariant_tl { dot }
    {
      \tl_set:Nn \l__datatool_JE_timevariant_tl
       { \l_datatool_current_language_tl _dot }
    }
    {
      \tl_if_in:NnF
        \l__datatool_JE_timevariant_tl
         { \l_datatool_current_language_tl }
       {
         \tl_set:Nn \l__datatool_JE_timevariant_tl
          { \l_datatool_current_language_tl _colon }
       }
    }
 } ,
time-variant / dialect-colon .code:n =
 {
   \tl_set:Nn \l__datatool_JE_timevariant_tl
     { \l_datatool_current_language_tl _colon }
 } ,
time-variant / dialect-dot .code:n =
 {
   \tl_set:Nn \l__datatool_JE_timevariant_tl
```

```
        { \l_datatool_current_language_tl _dot }
    } ,
  }
```

3.10.4 Hooks

Command to update temporal parsing commands for this region:

```
\newcommand \datatoolJESetTemporalParsers
  {
    \renewcommand \DTLCurrentLocaleParseTimeStamp
     { \datatool_JE_parse_timestamp:NnTF }
    \renewcommand \DTLCurrentLocaleParseDate
     { \datatool_JE_parse_date:NnTF }
    \renewcommand \DTLCurrentLocaleParseTime
     { \datatool_JE_parse_time:NnTF }
    \let
     \DTLCurrentLocaleGetTimeZoneMap
     \datatool_JE_get_timezone_map:n
  }
```

Set temporal formatting commands for this region. Currently this just resets to the default, but may change in future. Note that the defaults test if the applicable datetime2 command is available and will fallback on ISO if not defined. Bear in mind that the default style for datetime2 is `iso` so there won't be a noticeable difference unless the datetime2 regional setting is on.

```
\newcommand \datatoolJESetTemporalFormatters
  {
    \let
     \DTLCurrentLocaleFormatDate
     \datatool_default_date_fmt:nnnn
    \let
     \DTLCurrentLocaleFormatTime
     \datatool_default_time_fmt:nnn
    \let
     \DTLCurrentLocaleFormatTimeZone
     \datatool_default_timezone_fmt:nn
    \let
     \DTLCurrentLocaleFormatTimeStampNoZone
     \datatool_default_timestamp_fmt:nnnnnnn
    \let
     \DTLCurrentLocaleFormatTimeStampWithZone
     \datatool_default_timestamp_fmt:nnnnnnnnn
    \renewcommand \DTLCurrentLocaleTimeStampFmtSep { ~ }
  }
```

Command to update currency and temporal parsing commands for this region:

```
\newcommand \DTLJELocaleHook
  {
   \datatoolJESetNumberChars
   \datatoolJESetCurrency
   \datatoolJESetTemporalParsers
   \datatoolJESetTemporalFormatters
```

Allow language files to reference the region:

```
    \tl_set:Nn \l_datatool_current_region_tl { JE }
  }
```

Finished with LaTeX3 syntax.

```
\ExplSyntaxOff
```

Note that the hook is added to the captions by datatool-base not by the region file.

3.11 datatool-NZ.ldf

Support for region NZ.

```
\TrackLangProvidesResource{NZ}[2025/03/01 v1.0]
```

Switch on LaTeX3 syntax:

```
\ExplSyntaxOn
```

3.11.1 Numbers and Currency

Set the number group and decimal symbols for this region.

```
\cs_new:Nn \datatool_NZ_set_numberchars_official:
  {
   \datatool_set_numberchars:nn { , } { . }
  }
```

Unofficial style.

```
\cs_new:Nn \datatool_NZ_set_numberchars_unofficial:
  {
   \datatool_set_thinspace_group_decimal_char:n { . }
  }
```

Hook to set the number group and decimal characters for the region:

```
\newcommand \datatoolNZSetNumberChars
  {
    \bool_if:NT \l_datatool_region_set_numberchars_bool
     {
       \datatool_NZ_set_numberchars_official:
     }
  }
```

How to format the position of the currency symbol in relation to the value.

```
\cs_new:Nn \datatool_NZ_currency_position:nn
  {
    \dtlcurrprefixfmt { #1 } { #2 }
  }
```

Separator between currency symbol and value.

```
\tl_new:N \l_datatool_NZ_sym_sep_tl
```

Set the currency format for this region.

```
\newcommand \datatoolNZcurrencyfmt [ 2 ]
 {
   \datatool_NZ_currency_position:nn
    {
      \datatoolNZsymbolprefix { NZ }
      #1
    }
    { #2 }
 }
```

Prefix for symbol, if required.

```
\newcommand \datatoolNZsymbolprefix [ 1 ] { }
```

Define the currency symbols for this region.

```
\datatool_def_currency:nnnV
 { \datatoolNZcurrencyfmt }
 { NZD }
 { \$ }
 \c_dollar_str
```

Register the currency code with this region:

```
\datatool_register_regional_currency_code:nn { NZ } { NZD }
```

Provide a command to set the currency for this region (for use with any hook used when the locale changes). NB this should do nothing with `\DTLsetup{region-currency=false}`

```
\newcommand \datatoolNZSetCurrency
 {
   \bool_if:NT \l_datatool_region_set_currency_bool
    {
      \DTLsetdefaultcurrency { NZD }
```

Number of digits that `\DTLdecimaltocurrency` should round to:

```
      \renewcommand \DTLCurrentLocaleCurrencyDP { 2 }
```

Separator between symbol and value:

```
      \renewcommand \dtlcurrfmtsymsep { \l_datatool_NZ_sym_sep_tl }
    }
 }
```

3.11.2 Date and Time Parsing

This defaults to day/month/year but may be changed with `\DTLsetLocaleOptions`. For example:

```
\DTLsetLocaleOptions{NZ}{date-style=dmyyyy, date-variant=slash}
```

An appropriate language file will need to also be installed to parse dates containing month names or day of week names.

Provide a way to configure parsing style.

```
\tl_new:N \l__datatool_NZ_datevariant_tl
\tl_set:Nn \l__datatool_NZ_datevariant_tl { slash }
```

NB These token lists are used to form command names. The following is not a format string.

```
\tl_new:N \l__datatool_NZ_datestyle_tl
\tl_set:Nn \l__datatool_NZ_datestyle_tl { ddmmyyyy }
\tl_new:N \l__datatool_NZ_timevariant_tl
\tl_set:Nn \l__datatool_NZ_timevariant_tl { colon }
```

Each parsing command defined below has final {⟨*true*⟩} and {⟨*false*⟩} arguments (TF). These are used if parsing was successful (true) or if parsing failed (false). The internal commands used by datatool-base have no need for solo branches (only T or only F) so these commands are simply defined with `\cs_new:Nn` not as conditionals.

Time Stamp Parsing

Use command

`\datatool_`⟨*date-style*⟩`_hhmmss_tz_parse_timestamp:nnNnTF`

with date regular expression

`\c_datatool_`⟨*date-variant*⟩`_`⟨*date-style*⟩`_date_regex`

and time regular expression

`\c_datatool_`⟨*time-variant*⟩`_hhmmss_time_regex`

```
\cs_new:Nn \datatool_NZ_parse_timestamp:NnTF
 {
   \cs_if_exist:cTF
     {
       datatool_
       \l__datatool_NZ_datestyle_tl
       _hhmmss_tz_parse_timestamp:ccNnTF
     }
    {
      \cs_if_exist:cTF
       {
          c_datatool_
          \l__datatool_NZ_datevariant_tl
          _
          \l__datatool_NZ_datestyle_tl
          _date_regex
       }
       {
         \cs_if_exist:cTF
          {
            c_datatool_
            \l__datatool_NZ_timevariant_tl
            _hhmmss_time_regex
          }
          {
            \use:c
             {
               datatool_
              \l__datatool_NZ_datestyle_tl
              _hhmmss_tz_parse_timestamp:ccNnTF
             }
             {
               c_datatool_
               \l__datatool_NZ_datevariant_tl
               _
               \l__datatool_NZ_datestyle_tl
               _date_regex
             }
             {
               c_datatool_
               \l__datatool_NZ_timevariant_tl
               _hhmmss_time_regex
             }
              #1 { #2 } { #3 } { #4 }
          }
          {
            \datatool_warn_check_head_language_empty:Vnnn
             \l__datatool_NZ_timevariant_tl
             { datatool-NZ }
             {
               No ~ language ~ support ~ for ~ time ~ variant ~
               ` \exp_args:Ne \tl_tail:n { \l__datatool_NZ_timevariant_tl } '
             }
             {
              No ~ support ~ for ~ time ~ variant ~
              ` \l__datatool_NZ_timevariant_tl '
             }
            #4
          }
       }
       {
         \datatool_warn_check_head_language_empty:Vnnn
           \l__datatool_NZ_datevariant_tl
           { datatool-NZ }
           {
             No ~ language ~ support ~ for ~ date ~ style ~
              ` \l__datatool_NZ_datestyle_tl '
           }
           {
             No ~ support ~ for ~ date ~ style ~
             ` \l__datatool_NZ_datestyle_tl ' ~ with ~
             variant ~
             ` \l__datatool_NZ_datevariant_tl '
           }
         #4
```

```
      }
    }
    {
      \datatool_locale_warn:nn { datatool-NZ }
       {
          No ~ support ~ for ~ date ~ style ~
          ` \l__datatool_NZ_datestyle_tl '
       }
      #4
    }
}
```

Date Parsing

Use command `\datatool_⟨style⟩_parse_date:NNnTF` with regular expression

`\c_datatool_⟨variant⟩_anchored_⟨style⟩_date_regex`

```
\cs_new:Nn \datatool_NZ_parse_date:NnTF
 {
   \cs_if_exist:cTF
     {
       datatool_
       \l__datatool_NZ_datestyle_tl
       _parse_date:NNnTF
     }
    {
      \cs_if_exist:cTF
       {
         c_datatool_
         \l__datatool_NZ_datevariant_tl
         _anchored_
         \l__datatool_NZ_datestyle_tl
         _date_regex
       }
       {
         \exp_args:cc
         {
           datatool_
           \l__datatool_NZ_datestyle_tl
           _parse_date:NNnTF
         }
         {
           c_datatool_
           \l__datatool_NZ_datevariant_tl
           _anchored_
           \l__datatool_NZ_datestyle_tl
           _date_regex
         }
          #1 { #2 } { #3 } { #4 }
       }
       {
         \datatool_warn_check_head_language_empty:Vnnn
          \l__datatool_NZ_datevariant_tl
          { datatool-NZ }
          {
            No ~ language ~ support ~ for ~ date ~ style ~
             ` \l__datatool_NZ_datestyle_tl '
          }
          {
             No ~ support ~ for ~ date ~ style ~
             ` \l__datatool_NZ_datestyle_tl ' ~ with ~
             variant ~
             ` \l__datatool_NZ_datevariant_tl '
          }
         #4
       }
    }
    {
      \datatool_locale_warn:nn { datatool-NZ }
       {
          No ~ support ~ for ~ date ~ style ~
          ` \l__datatool_NZ_datestyle_tl '
       }
      #4
    }
 }
```

Time Parsing

Use command \datatool_⟨*style*⟩_parse_time:NNnTF with regular expression

\c_datatool_⟨*variant*⟩_anchored_⟨*style*⟩_time_regex

```
\cs_new:Nn \datatool_NZ_parse_time:NnTF
 {
   \cs_if_exist:cTF
    {
      c_datatool_
      \l__datatool_NZ_timevariant_tl
      _anchored_hhmmss_time_regex
    }
    {
      \datatool_hhmmss_parse_time:cNnTF
       {
        c_datatool_
        \l__datatool_NZ_timevariant_tl
        _anchored_hhmmss_time_regex
       }
        #1 { #2 } { #3 } { #4 }
    }
    {
      \datatool_warn_check_head_language_empty:Vnnn
        \l__datatool_NZ_timevariant_tl
        { datatool-NZ }
        {
           No ~ language ~ support ~ for ~ time ~ variant ~
           ` \exp_args:Ne \tl_tail:n { \l__datatool_NZ_timevariant_tl } '
        }
        {
           No ~ support ~ for ~ time ~ variant ~
           ` \l__datatool_NZ_timevariant_tl '
        }
      #4
    }
 }
```

Time Zone Mappings

Define time zone mapping command for this region:

```
\cs_new:Nn \datatool_NZ_get_timezone_map:n
 {
   \datatool_region_get_timezone_map:n { NZ / #1 }
 }
```

Define time zone IDs for this region (one-way map from ID to offset):

```
\datatool_region_set_timezone_map:nn { NZ / NZST } { +12:00 }
\datatool_region_set_timezone_map:nn { NZ / CHADT } { +13:45 }
\datatool_region_set_timezone_map:nn { NZ / NZDT } { +13:00 }
\datatool_region_set_timezone_map:nn { NZ / CHAST } { +12:45 }
```

3.11.3 Options

Define options for this region:

```
\datatool_locale_define_keys:nn { NZ }
 {
   number-style .choices:nn =
    { official, unofficial }
    {
      \exp_args:NNe \renewcommand
        \datatoolNZSetNumberChars
         {
           \exp_not:N \bool_if:NT
            \exp_not:N \l_datatool_region_set_numberchars_bool
             {
               \exp_not:c { datatool_NZ_set_numberchars_ \l_keys_choice_tl : }
             }
         }
      \tl_if_eq:NnT \l_datatool_current_region_tl { NZ }
       {
         \datatoolNZSetNumberChars
       }
    } ,
```

Currency symbol before or after value:

```
currency-symbol-position .choice: ,
currency-symbol-position / before .code:n =
 {
   \cs_set_eq:NN \datatool_NZ_currency_position:nn \dtlcurrprefixfmt
 } ,
currency-symbol-position / after .code:n =
 {
   \cs_set_eq:NN \datatool_NZ_currency_position:nn \dtlcurrsuffixfmt
 } ,
```

Should the currency symbol be prefixed with the region code:

```
currency-symbol-prefix .choice: ,
currency-symbol-prefix / false .code:n =
 {
   \cs_set_eq:NN \datatoolNZsymbolprefix \use_none:n
 } ,
currency-symbol-prefix / true .code:n =
 {
   \cs_set_eq:NN
    \datatoolNZsymbolprefix
    \datatool_currency_symbol_region_prefix:n
 } ,
currency-symbol-prefix .default:n = { true } ,
```

Separator between currency symbol (not code) and value:

```
currency-symbol-sep .choice: ,
currency-symbol-sep / none .code:n =
 {
   \tl_clear:N \l_datatool_NZ_sym_sep_tl
 } ,
currency-symbol-sep / thin-space .code:n =
 {
   \tl_set:Nn \l_datatool_NZ_sym_sep_tl { \, }
 } ,
currency-symbol-sep / space .code:n =
 {
   \tl_set:Nn \l_datatool_NZ_sym_sep_tl { ~ }
 } ,
currency-symbol-sep / nbsp .code:n =
 {
   \tl_set:Nn \l_datatool_NZ_sym_sep_tl { \nobreakspace }
 } ,
currency-symbol-sep . initial:n = { none } ,
```

Date and time styles:

```
date-style .choice: ,
date-style / dmyyyy .code: n =
 {
   \tl_set:Nn \l__datatool_NZ_datestyle_tl { ddmmyyyy }
 } ,
date-style / mdyyyy .code: n =
 {
   \tl_set:Nn \l__datatool_NZ_datestyle_tl { mmddyyyy }
 } ,
date-style / yyyymd .code: n =
 {
   \tl_set:Nn \l__datatool_NZ_datestyle_tl { yyyymmdd }
 } ,
date-style / dmyy .code: n =
 {
   \tl_set:Nn \l__datatool_NZ_datestyle_tl { ddmmyy }
 } ,
date-style / mdyy .code: n =
 {
   \tl_set:Nn \l__datatool_NZ_datestyle_tl { mmddyy }
 } ,
date-style / yymd .code: n =
 {
   \tl_set:Nn \l__datatool_NZ_datestyle_tl { yymmdd }
 } ,
date-variant .choice: ,
date-variant / slash .code:n =
 {
   \tl_set:Nn \l__datatool_NZ_datevariant_tl { slash }
 } ,
date-variant / hyphen .code:n =
 {
   \tl_set:Nn \l__datatool_NZ_datevariant_tl { hyphen }
 } ,
date-variant / dot .code:n =
 {
   \tl_set:Nn \l__datatool_NZ_datevariant_tl { dot }
 } ,
date-variant / dialect .code:n =
 {
   \tl_set:Nn \l__datatool_NZ_datevariant_tl
     { \l_datatool_current_language_tl }
 } ,
time-variant .choice: ,
time-variant / colon .code:n =
 {
   \tl_set:Nn \l__datatool_NZ_timevariant_tl { colon }
 } ,
time-variant / dot .code:n =
 {
   \tl_set:Nn \l__datatool_NZ_timevariant_tl { dot }
 } ,
time-variant / dialect .code:n =
 {
   \tl_if_eq:NnTF \l__datatool_NZ_timevariant_tl { dot }
    {
      \tl_set:Nn \l__datatool_NZ_timevariant_tl
       { \l_datatool_current_language_tl _dot }
    }
    {
      \tl_if_in:NnF
```

```
            \l__datatool_NZ_timevariant_tl
              { \l_datatool_current_language_tl }
            {
              \tl_set:Nn \l__datatool_NZ_timevariant_tl
                { \l_datatool_current_language_tl _colon }
            }
        }
     } ,
    time-variant / dialect-colon .code:n =
     {
       \tl_set:Nn \l__datatool_NZ_timevariant_tl
         { \l_datatool_current_language_tl _colon }
     } ,
    time-variant / dialect-dot .code:n =
     {
       \tl_set:Nn \l__datatool_NZ_timevariant_tl
         { \l_datatool_current_language_tl _dot }
     } ,
  }
```

3.11.4 Hooks

Command to update temporal parsing commands for this region:

```
\newcommand \datatoolNZSetTemporalParsers
 {
   \renewcommand \DTLCurrentLocaleParseTimeStamp
    { \datatool_NZ_parse_timestamp:NnTF }
   \renewcommand \DTLCurrentLocaleParseDate
    { \datatool_NZ_parse_date:NnTF }
   \renewcommand \DTLCurrentLocaleParseTime
    { \datatool_NZ_parse_time:NnTF }
   \let
    \DTLCurrentLocaleGetTimeZoneMap
    \datatool_NZ_get_timezone_map:n
 }
```

Set temporal formatting commands for this region. Currently this just resets to the default, but may change in future. Note that the defaults test if the applicable datetime2 command is available and will fallback on ISO if not defined. Bear in mind that the default style for datetime2 is `iso` so there won't be a noticeable difference unless the datetime2 regional setting is on.

```
\newcommand \datatoolNZSetTemporalFormatters
 {
   \let
    \DTLCurrentLocaleFormatDate
    \datatool_default_date_fmt:nnnn
   \let
    \DTLCurrentLocaleFormatTime
    \datatool_default_time_fmt:nnn
   \let
    \DTLCurrentLocaleFormatTimeZone
    \datatool_default_timezone_fmt:nn
   \let
    \DTLCurrentLocaleFormatTimeStampNoZone
    \datatool_default_timestamp_fmt:nnnnnnn
   \let
    \DTLCurrentLocaleFormatTimeStampWithZone
    \datatool_default_timestamp_fmt:nnnnnnnnn
   \renewcommand \DTLCurrentLocaleTimeStampFmtSep { ~ }
 }
```

Command to update currency and temporal parsing commands for this region:

```
\newcommand \DTLNZLocaleHook
 {
  \datatoolNZSetNumberChars
  \datatoolNZSetCurrency
  \datatoolNZSetTemporalParsers
  \datatoolNZSetTemporalFormatters
```

Allow language files to reference the region:

```
  \tl_set:Nn \l_datatool_current_region_tl { NZ }
 }
```

Finished with LaTeX3 syntax.

```
\ExplSyntaxOff
```

Note that the hook is added to the captions by datatool-base not by the region file.

3.12 datatool-US.ldf

Support for region US.

```
\TrackLangProvidesResource{US}[2025/03/01 v1.0]
```

Switch on LaTeX3 syntax:

```
\ExplSyntaxOn
```

3.12.1 Numbers and Currency

Set the number group and decimal symbols for this region.

```
\cs_new:Nn \datatool_US_set_numberchars_official:
 {
  \datatool_set_numberchars:nn { , } { . }
 }
```

Unofficial style.

```
\cs_new:Nn \datatool_US_set_numberchars_unofficial:
 {
  \datatool_set_thinspace_group_decimal_char:n { . }
 }
```

Hook to set the number group and decimal characters for the region:

```
\newcommand \datatoolUSSetNumberChars
 {
   \bool_if:NT \l_datatool_region_set_numberchars_bool
    {
      \datatool_US_set_numberchars_official:
    }
 }
```

How to format the position of the currency symbol in relation to the value.

```
 \cs_new:Nn \datatool_US_currency_position:nn
  {
    \dtlcurrprefixfmt { #1 } { #2 }
  }
```

Separator between currency symbol and value.

```
 \tl_new:N \l_datatool_US_sym_sep_tl
```

Set the currency format for this region.

```
 \newcommand \datatoolUScurrencyfmt [ 2 ]
  {
    \datatool_US_currency_position:nn
     {
       \datatoolUSsymbolprefix { US }
       #1
     }
     { #2 }
  }
```

Prefix for symbol, if required.

```
 \newcommand \datatoolUSsymbolprefix [ 1 ] { }
```

Define the currency symbols for this region.

```
 \datatool_def_currency:nnnV
  { \datatoolUScurrencyfmt }
  { USD }
  { \$ }
  \c_dollar_str
```

Register the currency code with this region:

```
 \datatool_register_regional_currency_code:nn { US } { USD }
```

Provide a command to set the currency for this region (for use with any hook used when the locale changes). NB this should do nothing with `\DTLsetup{region-currency=false}`

```
 \newcommand \datatoolUSSetCurrency
  {
    \bool_if:NT \l_datatool_region_set_currency_bool
     {
       \DTLsetdefaultcurrency { USD }
```

Number of digits that `\DTLdecimaltocurrency` should round to:

```
       \renewcommand \DTLCurrentLocaleCurrencyDP { 2 }
```

Separator between symbol and value:

```
       \renewcommand \dtlcurrfmtsymsep { \l_datatool_US_sym_sep_tl }
     }
  }
```

3.12.2 Date and Time Parsing

This defaults to month/day/year but may be changed with `\DTLsetLocaleOptions`. For example:

```
\DTLsetLocaleOptions{US}{date-style=dmyyyy, date-variant=slash}
```

An appropriate language file will need to also be installed to parse dates containing month names or day of week names.

Provide a way to configure parsing style.

```
\tl_new:N \l__datatool_US_datevariant_tl
\tl_set:Nn \l__datatool_US_datevariant_tl { slash }
```

NB These token lists are used to form command names. The following is not a format string.

```
\tl_new:N \l__datatool_US_datestyle_tl
\tl_set:Nn \l__datatool_US_datestyle_tl { mmddyyyy }
\tl_new:N \l__datatool_US_timevariant_tl
\tl_set:Nn \l__datatool_US_timevariant_tl { colon }
```

Each parsing command defined below has final {⟨*true*⟩} and {⟨*false*⟩} arguments (TF). These are used if parsing was successful (true) or if parsing failed (false). The internal commands used by datatool-base have no need for solo branches (only T or only F) so these commands are simply defined with `\cs_new:Nn` not as conditionals.

Time Stamp Parsing

Use command

```
\datatool_⟨date-style⟩_hhmmss_tz_parse_timestamp:nnNnTF
```

with date regular expression

```
\c_datatool_⟨date-variant⟩_⟨date-style⟩_date_regex
```

and time regular expression

```
\c_datatool_⟨time-variant⟩_hhmmss_time_regex
```

```
\cs_new:Nn \datatool_US_parse_timestamp:NnTF
 {
   \cs_if_exist:cTF
     {
       datatool_
       \l__datatool_US_datestyle_tl
       _hhmmss_tz_parse_timestamp:ccNnTF
     }
```

```
{
  \cs_if_exist:cTF
   {
      c_datatool_
      \l__datatool_US_datevariant_tl
      _
      \l__datatool_US_datestyle_tl
      _date_regex
   }
   {
     \cs_if_exist:cTF
      {
        c_datatool_
        \l__datatool_US_timevariant_tl
        _hhmmss_time_regex
      }
      {
        \use:c
         {
           datatool_
          \l__datatool_US_datestyle_tl
          _hhmmss_tz_parse_timestamp:ccNnTF
         }
         {
           c_datatool_
           \l__datatool_US_datevariant_tl
           _
           \l__datatool_US_datestyle_tl
           _date_regex
         }
         {
           c_datatool_
           \l__datatool_US_timevariant_tl
           _hhmmss_time_regex
         }
          #1 { #2 } { #3 } { #4 }
      }
      {
        \datatool_warn_check_head_language_empty:Vnnn
         \l__datatool_US_timevariant_tl
         { datatool-US }
         {
           No ~ language ~ support ~ for ~ time ~ variant ~
           ` \exp_args:Ne \tl_tail:n { \l__datatool_US_timevariant_tl } '
         }
         {
          No ~ support ~ for ~ time ~ variant ~
          ` \l__datatool_US_timevariant_tl '
         }
        #4
      }
   }
   {
          \datatool_warn_check_head_language_empty:Vnnn
            \l__datatool_US_datevariant_tl
            { datatool-US }
            {
              No ~ language ~ support ~ for ~ date ~ style ~
               ` \l__datatool_US_datestyle_tl '
            }
            {
              No ~ support ~ for ~ date ~ style ~
              ` \l__datatool_US_datestyle_tl ' ~ with ~
              variant ~
              ` \l__datatool_US_datevariant_tl '
            }
          #4
        }
     }
     {
       \datatool_locale_warn:nn { datatool-US }
        {
          No ~ support ~ for ~ date ~ style ~
          ` \l__datatool_US_datestyle_tl '
        }
       #4
     }
  }
```

Date Parsing

Use command `\datatool_`⟨*style*⟩`_parse_date:NNnTF` with regular expression

`\c_datatool_`⟨*variant*⟩`_anchored_`⟨*style*⟩`_date_regex`

```
\cs_new:Nn \datatool_US_parse_date:NnTF
 {
   \cs_if_exist:cTF
     {
       datatool_
       \l__datatool_US_datestyle_tl
       _parse_date:NNnTF
     }
    {
      \cs_if_exist:cTF
       {
         c_datatool_
         \l__datatool_US_datevariant_tl
         _anchored_
         \l__datatool_US_datestyle_tl
         _date_regex
       }
       {
         \exp_args:cc
         {
           datatool_
           \l__datatool_US_datestyle_tl
           _parse_date:NNnTF
         }
         {
           c_datatool_
           \l__datatool_US_datevariant_tl
           _anchored_
           \l__datatool_US_datestyle_tl
           _date_regex
         }
          #1 { #2 } { #3 } { #4 }
       }
       {
         \datatool_warn_check_head_language_empty:Vnnn
          \l__datatool_US_datevariant_tl
          { datatool-US }
          {
            No ~ language ~ support ~ for ~ date ~ style ~
             ` \l__datatool_US_datestyle_tl '
          }
          {
             No ~ support ~ for ~ date ~ style ~
             ` \l__datatool_US_datestyle_tl ' ~ with ~
             variant ~
             ` \l__datatool_US_datevariant_tl '
          }
         #4
       }
    }
    {
      \datatool_locale_warn:nn { datatool-US }
       {
          No ~ support ~ for ~ date ~ style ~
          ` \l__datatool_US_datestyle_tl '
       }
      #4
    }
 }
```

Time Parsing

Use command `\datatool_`⟨*style*⟩`_parse_time:NNnTF` with regular expression

`\c_datatool_`⟨*variant*⟩`_anchored_`⟨*style*⟩`_time_regex`

```
\cs_new:Nn \datatool_US_parse_time:NnTF
 {
   \cs_if_exist:cTF
    {
      c_datatool_
      \l__datatool_US_timevariant_tl
      _anchored_hhmmss_time_regex
    }
    {
      \datatool_hhmmss_parse_time:cNnTF
       {
        c_datatool_
        \l__datatool_US_timevariant_tl
        _anchored_hhmmss_time_regex
       }
        #1 { #2 } { #3 } { #4 }
    }
    {
      \datatool_warn_check_head_language_empty:Vnnn
        \l__datatool_US_timevariant_tl
        { datatool-US }
        {
           No ~ language ~ support ~ for ~ time ~ variant ~
           ` \exp_args:Ne \tl_tail:n { \l__datatool_US_timevariant_tl } '
        }
        {
           No ~ support ~ for ~ time ~ variant ~
           ` \l__datatool_US_timevariant_tl '
        }
      #4
    }
 }
```

Time Zone Mappings

Define time zone mapping command for this region:

```
\cs_new:Nn \datatool_US_get_timezone_map:n
 {
   \datatool_region_get_timezone_map:n { US / #1 }
 }
```

Define time zone IDs for this region (one-way map from ID to offset):

```
\datatool_region_set_timezone_map:nn { US / AST } { -04:00 }
\datatool_region_set_timezone_map:nn { US / ADT } { -03:00 }
\datatool_region_set_timezone_map:nn { US / EST } { -05:00 }
\datatool_region_set_timezone_map:nn { US / EDT } { -04:00 }
\datatool_region_set_timezone_map:nn { US / CST } { -06:00 }
\datatool_region_set_timezone_map:nn { US / CDT } { -05:00 }
\datatool_region_set_timezone_map:nn { US / MST } { -07:00 }
\datatool_region_set_timezone_map:nn { US / MDT } { -06:00 }
\datatool_region_set_timezone_map:nn { US / PST } { -08:00 }
\datatool_region_set_timezone_map:nn { US / PDT } { -07:00 }
\datatool_region_set_timezone_map:nn { US / AKST } { -09:00 }
\datatool_region_set_timezone_map:nn { US / AKDT } { -08:00 }
\datatool_region_set_timezone_map:nn { US / HAST } { -10:00 }
\datatool_region_set_timezone_map:nn { US / HADT } { -9:00 }
```

```
\datatool_region_set_timezone_map:nn { US / SST } { -11:00 }
\datatool_region_set_timezone_map:nn { US / ChST } { 10:00 }
```

3.12.3 Options

Define options for this region:

```
\datatool_locale_define_keys:nn { US }
 {
   number-style .choices:nn =
    { official, unofficial }
    {
      \exp_args:NNe \renewcommand
        \datatoolUSSetNumberChars
         {
           \exp_not:N \bool_if:NT
            \exp_not:N \l_datatool_region_set_numberchars_bool
             {
               \exp_not:c { datatool_US_set_numberchars_ \l_keys_choice_tl : }
             }
         }
      \tl_if_eq:NnT \l_datatool_current_region_tl { US }
       {
         \datatoolUSSetNumberChars
       }
    } ,
```

Currency symbol before or after value:

```
   currency-symbol-position .choice: ,
   currency-symbol-position / before .code:n =
    {
      \cs_set_eq:NN \datatool_US_currency_position:nn \dtlcurrprefixfmt
    } ,
   currency-symbol-position / after .code:n =
    {
      \cs_set_eq:NN \datatool_US_currency_position:nn \dtlcurrsuffixfmt
    } ,
```

Should the currency symbol be prefixed with the region code:

```
   currency-symbol-prefix .choice: ,
   currency-symbol-prefix / false .code:n =
    {
      \cs_set_eq:NN \datatoolUSsymbolprefix \use_none:n
    } ,
   currency-symbol-prefix / true .code:n =
    {
      \cs_set_eq:NN
       \datatoolUSsymbolprefix
       \datatool_currency_symbol_region_prefix:n
    } ,
   currency-symbol-prefix .default:n = { true } ,
```

Separator between currency symbol (not code) and value:

```
   currency-symbol-sep .choice: ,
currency-symbol-sep / none .code:n =
 {
   \tl_clear:N \l_datatool_US_sym_sep_tl
 } ,
currency-symbol-sep / thin-space .code:n =
 {
   \tl_set:Nn \l_datatool_US_sym_sep_tl { \, }
 } ,
currency-symbol-sep / space .code:n =
 {
   \tl_set:Nn \l_datatool_US_sym_sep_tl { ~ }
 } ,
currency-symbol-sep / nbsp .code:n =
 {
   \tl_set:Nn \l_datatool_US_sym_sep_tl { \nobreakspace }
 } ,
currency-symbol-sep . initial:n = { none } ,
```

Date and time styles:

```
date-style .choice: ,
date-style / dmyyyy .code: n =
 {
   \tl_set:Nn \l__datatool_US_datestyle_tl { ddmmyyyy }
 } ,
date-style / mdyyyy .code: n =
 {
   \tl_set:Nn \l__datatool_US_datestyle_tl { mmddyyyy }
 } ,
date-style / yyyymd .code: n =
 {
   \tl_set:Nn \l__datatool_US_datestyle_tl { yyyymmdd }
 } ,
date-style / dmyy .code: n =
 {
   \tl_set:Nn \l__datatool_US_datestyle_tl { ddmmyy }
 } ,
date-style / mdyy .code: n =
 {
   \tl_set:Nn \l__datatool_US_datestyle_tl { mmddyy }
 } ,
date-style / yymd .code: n =
 {
   \tl_set:Nn \l__datatool_US_datestyle_tl { yymmdd }
 } ,
date-variant .choice: ,
date-variant / slash .code:n =
 {
   \tl_set:Nn \l__datatool_US_datevariant_tl { slash }
 } ,
date-variant / hyphen .code:n =
 {
   \tl_set:Nn \l__datatool_US_datevariant_tl { hyphen }
 } ,
date-variant / dot .code:n =
     {
       \tl_set:Nn \l__datatool_US_datevariant_tl { dot }
     } ,
    date-variant / dialect .code:n =
     {
       \tl_set:Nn \l__datatool_US_datevariant_tl
         { \l_datatool_current_language_tl }
     } ,
    time-variant .choice: ,
    time-variant / colon .code:n =
     {
       \tl_set:Nn \l__datatool_US_timevariant_tl { colon }
     } ,
    time-variant / dot .code:n =
     {
       \tl_set:Nn \l__datatool_US_timevariant_tl { dot }
     } ,
    time-variant / dialect .code:n =
     {
       \tl_if_eq:NnTF \l__datatool_US_timevariant_tl { dot }
        {
          \tl_set:Nn \l__datatool_US_timevariant_tl
           { \l_datatool_current_language_tl _dot }
        }
        {
          \tl_if_in:NnF
            \l__datatool_US_timevariant_tl
             { \l_datatool_current_language_tl }
           {
             \tl_set:Nn \l__datatool_US_timevariant_tl
              { \l_datatool_current_language_tl _colon }
           }
        }
     } ,
    time-variant / dialect-colon .code:n =
     {
       \tl_set:Nn \l__datatool_US_timevariant_tl
         { \l_datatool_current_language_tl _colon }
     } ,
    time-variant / dialect-dot .code:n =
     {
       \tl_set:Nn \l__datatool_US_timevariant_tl
         { \l_datatool_current_language_tl _dot }
     } ,
  }
```

3.12.4 Hooks

Command to update temporal parsing commands for this region:

```
\newcommand \datatoolUSSetTemporalParsers
 {
   \renewcommand \DTLCurrentLocaleParseTimeStamp
```

```
      { \datatool_US_parse_timestamp:NnTF }
    \renewcommand \DTLCurrentLocaleParseDate
      { \datatool_US_parse_date:NnTF }
    \renewcommand \DTLCurrentLocaleParseTime
      { \datatool_US_parse_time:NnTF }
    \let
      \DTLCurrentLocaleGetTimeZoneMap
      \datatool_US_get_timezone_map:n
  }
```

Set temporal formatting commands for this region. Currently this just resets to the default, but may change in future. Note that the defaults test if the applicable datetime2 command is available and will fallback on ISO if not defined. Bear in mind that the default style for datetime2 is `iso` so there won't be a noticeable difference unless the datetime2 regional setting is on.

```
\newcommand \datatoolUSSetTemporalFormatters
  {
    \let
      \DTLCurrentLocaleFormatDate
      \datatool_default_date_fmt:nnnn
    \let
      \DTLCurrentLocaleFormatTime
      \datatool_default_time_fmt:nnn
    \let
      \DTLCurrentLocaleFormatTimeZone
      \datatool_default_timezone_fmt:nn
    \let
      \DTLCurrentLocaleFormatTimeStampNoZone
      \datatool_default_timestamp_fmt:nnnnnnn
    \let
      \DTLCurrentLocaleFormatTimeStampWithZone
      \datatool_default_timestamp_fmt:nnnnnnnnn
    \renewcommand \DTLCurrentLocaleTimeStampFmtSep { ~ }
  }
```

Command to update currency and temporal parsing commands for this region:

```
\newcommand \DTLUSLocaleHook
  {
   \datatoolUSSetNumberChars
   \datatoolUSSetCurrency
   \datatoolUSSetTemporalParsers
   \datatoolUSSetTemporalFormatters
```

Allow language files to reference the region:

```
   \tl_set:Nn \l_datatool_current_region_tl { US }
  }
```

Finished with LaTeX3 syntax.

```
\ExplSyntaxOff
```

Note that the hook is added to the captions by datatool-base not by the region file.

3.13 datatool-ZA.ldf

Support for region ZA.

```
\TrackLangProvidesResource{ZA}[2025/03/01 v1.0]
```

Switch on LaTeX3 syntax:

```
\ExplSyntaxOn
```

3.13.1 Numbers and Currency

Official style is to use dot separator and decimal dot.

```
\cs_new:Nn \datatool_ZA_set_numberchars_official:
 {
   \DTLsetnumberchars { . } { , }
 }
```

but allow dialect to use another style.

```
\cs_new:Nn \datatool_ZA_set_numberchars_dialect:
 {
   \cs_if_exist_use:cF
    { datatool \l_datatool_current_language_tl ZASetNumberChars }
    {
      \datatool_ZA_set_numberchars_official:
    }
 }
```

Hook to set the number group and decimal characters for the region:

```
\newcommand \datatoolZASetNumberChars
 {
   \bool_if:NT \l_datatool_region_set_numberchars_bool
    {
      \datatool_ZA_set_numberchars_official:
    }
 }
```

How to format the position of the currency symbol in relation to the value.

```
\cs_new:Nn \datatool_ZA_currency_position:nn
 {
   \dtlcurrprefixfmt { #1 } { #2 }
 }
```

Separator between currency symbol and value.

```
\tl_new:N \l_datatool_ZA_sym_sep_tl
```

Set the currency format for this region.

```
\newcommand \datatoolZAcurrencyfmt { \datatool_ZA_currency_position:nn }
```

Define the currency symbols for this region.

```
\datatool_def_currency:nnnn
 { \datatoolZAcurrencyfmt }
 { ZAR } { R } { R }
```

Register the currency code with this region:

```
\datatool_register_regional_currency_code:nn { ZA } { ZAR }
```

Provide a command to set the currency for this region (for use with any hook used when the locale changes). NB this should do nothing with `\DTLsetup{region-currency=false}`

```
\newcommand \datatoolZASetCurrency
  {
    \bool_if:NT \l_datatool_region_set_currency_bool
      {
        \DTLsetdefaultcurrency { ZAR }
```

Number of digits that `\DTLdecimaltocurrency` should round to:

```
        \renewcommand \DTLCurrentLocaleCurrencyDP { 2 }
```

Separator between symbol and value:

```
        \renewcommand \dtlcurrfmtsymsep { \l_datatool_ZA_sym_sep_tl }
      }
  }
```

3.13.2 Date and Time Parsing

Date parsing is more complicated as there seems to be a confusing mixture of ISO, day/month/year and month/day/year formats. The default is year-month-day but may be changed with `\DTLsetLocaleOptions`. For example:

```
\DTLsetLocaleOptions{ZA}{date-style=dmyyyy, date-variant=slash}
```

An appropriate language file will need to also be installed to parse dates containing month names or day of week names.

```
\tl_new:N \l__datatool_ZA_datevariant_tl
\tl_set:Nn \l__datatool_ZA_datevariant_tl { hyphen }
```

NB These token lists are used to form command names. The following is not a format string.

```
\tl_new:N \l__datatool_ZA_datestyle_tl
\tl_set:Nn \l__datatool_ZA_datestyle_tl { yyyymmdd }
\tl_new:N \l__datatool_ZA_timevariant_tl
\tl_set:Nn \l__datatool_ZA_timevariant_tl { colon }
```

Each parsing command defined below has final {⟨*true*⟩} and {⟨*false*⟩} arguments (TF). These are used if parsing was successful (true) or if parsing failed (false). The internal commands used by datatool-base have no need for solo branches (only T or only F) so these commands are simply defined with `\cs_new:Nn` not as conditionals.

Time Stamp Parsing

Use command

```
\datatool_⟨date-style⟩_hhmmss_tz_parse_timestamp:nnNnTF
```

with date regular expression

\c_datatool_⟨*date-variant*⟩_⟨*date-style*⟩_date_regex

and time regular expression

\c_datatool_⟨*time-variant*⟩_hhmmss_time_regex

```
\cs_new:Nn \datatool_ZA_parse_timestamp:NnTF
 {
   \cs_if_exist:cTF
     {
       datatool_
       \l__datatool_ZA_datestyle_tl
       _hhmmss_tz_parse_timestamp:ccNnTF
     }
    {
      \cs_if_exist:cTF
       {
          c_datatool_
          \l__datatool_ZA_datevariant_tl
          _
          \l__datatool_ZA_datestyle_tl
          _date_regex
       }
       {
         \cs_if_exist:cTF
          {
            c_datatool_
            \l__datatool_ZA_timevariant_tl
            _hhmmss_time_regex
          }
          {
            \use:c
             {
               datatool_
              \l__datatool_ZA_datestyle_tl
              _hhmmss_tz_parse_timestamp:ccNnTF
             }
             {
               c_datatool_
               \l__datatool_ZA_datevariant_tl
               _
               \l__datatool_ZA_datestyle_tl
               _date_regex
             }
             {
               c_datatool_
               \l__datatool_ZA_timevariant_tl
               _hhmmss_time_regex
             }
              #1 { #2 } { #3 } { #4 }
          }
          {
            \datatool_locale_warn:nn { datatool-ZA }
             {
              No ~ support ~ for ~ time ~ variant ~
              ` \l__datatool_ZA_timevariant_tl '
             }
            #4
          }
       }
       {
         \datatool_locale_warn:nn { datatool-ZA }
          {
           No ~ support ~ for ~ date ~ style ~
           ` \l__datatool_ZA_datestyle_tl ' ~ with ~
           variant ~
           ` \l__datatool_ZA_datevariant_tl '
          }
         #4
       }
    }
    {
      \datatool_locale_warn:nn { datatool-ZA }
       {
          No ~ support ~ for ~ date ~ style ~
          ` \l__datatool_ZA_datestyle_tl '
       }
      #4
    }
 }
```

Date Parsing

Use command `\datatool_`⟨*style*⟩`_parse_date:NNnTF` with regular expression

`\c_datatool_`⟨*variant*⟩`_anchored_`⟨*style*⟩`_date_regex`

```
\cs_new:Nn \datatool_ZA_parse_date:NnTF
 {
   \cs_if_exist:cTF
     {
       datatool_
       \l__datatool_ZA_datestyle_tl
       _parse_date:NNnTF
     }
    {
      \cs_if_exist:cTF
       {
         c_datatool_
         \l__datatool_ZA_datevariant_tl
         _anchored_
         \l__datatool_ZA_datestyle_tl
         _date_regex
       }
       {
         \exp_args:cc
         {
           datatool_
           \l__datatool_ZA_datestyle_tl
           _parse_date:NNnTF
         }
         {
           c_datatool_
           \l__datatool_ZA_datevariant_tl
           _anchored_
           \l__datatool_ZA_datestyle_tl
           _date_regex
         }
          #1 { #2 } { #3 } { #4 }
       }
       {
         \datatool_locale_warn:nn { datatool-ZA }
          {
           No ~ support ~ for ~ date ~ style ~
           ` \l__datatool_ZA_datestyle_tl ' ~ with ~
           variant ~
           ` \l__datatool_ZA_datevariant_tl '
          }
         #4
       }
    }
    {
      \datatool_locale_warn:nn { datatool-ZA }
       {
          No ~ support ~ for ~ date ~ style ~
          ` \l__datatool_ZA_datestyle_tl '
       }
      #4
    }
 }
```

Time Parsing

Use command \datatool_⟨*style*⟩_parse_time:NNnTF with regular expression

\c_datatool_⟨*variant*⟩_anchored_⟨*style*⟩_time_regex

```
\cs_new:Nn \datatool_ZA_parse_time:NnTF
 {
   \cs_if_exist:cTF
    {
      c_datatool_
      \l__datatool_ZA_timevariant_tl
      _anchored_hhmmss_time_regex
    }
    {
      \datatool_hhmmss_parse_time:cNnTF
       {
        c_datatool_
        \l__datatool_ZA_timevariant_tl
        _anchored_hhmmss_time_regex
       }
        #1 { #2 } { #3 } { #4 }
    }
    {
      \datatool_locale_warn:nn { datatool-ZA }
       {
        No ~ support ~ for ~ time ~ variant ~
        ` \l__datatool_ZA_timevariant_tl '
       }
      #4
    }
 }
```

Time Zone Mapping

Define time zone mapping command for this region:

```
\cs_new:Nn \datatool_ZA_get_timezone_map:n
 {
   \datatool_region_get_timezone_map:n { ZA / #1 }
 }
```

Define time zone IDs for this region (one-way map from ID to offset):

```
\datatool_region_set_timezone_map:nn { ZA / SAST } { +02:00 }
```

3.13.3 Options

Define options for this region:

```
\datatool_locale_define_keys:nn { ZA }
 {
   number-style .choices:nn =
    { official , dialect }
    {
      \exp_args:NNe \renewcommand
        \datatoolZASetNumberChars
         {
           \exp_not:N \bool_if:NT
            \exp_not:N \l_datatool_region_set_numberchars_bool
             {
               \exp_not:c { datatool_ZA_set_numberchars_ \l_keys_choice_tl : }
             }
         }
      \tl_if_eq:NnT \l_datatool_current_region_tl { ZA }
       {
         \datatoolZASetNumberChars
       }
    } ,
```

Currency symbol before or after value:

```
 currency-symbol-position .choice: ,
 currency-symbol-position / before .code:n =
  {
    \cs_set_eq:NN \datatool_ZA_currency_position:nn \dtlcurrprefixfmt
  } ,
 currency-symbol-position / after .code:n =
  {
    \cs_set_eq:NN \datatool_ZA_currency_position:nn \dtlcurrsuffixfmt
  } ,
```

Separator between currency symbol (not code) and value:

```
 currency-symbol-sep .choice: ,
 currency-symbol-sep / none .code:n =
  {
    \tl_clear:N \l_datatool_ZA_sym_sep_tl
  } ,
 currency-symbol-sep / thin-space .code:n =
  {
    \tl_set:Nn \l_datatool_ZA_sym_sep_tl { \, }
  } ,
 currency-symbol-sep / space .code:n =
  {
    \tl_set:Nn \l_datatool_ZA_sym_sep_tl { ~ }
  } ,
 currency-symbol-sep / nbsp .code:n =
  {
    \tl_set:Nn \l_datatool_ZA_sym_sep_tl { \nobreakspace }
  } ,
 currency-symbol-sep .initial:n = { none } ,
```

Date and time styles:

```
 date-style .choice: ,
 date-style / dmyyyy .code: n =
  {
    \tl_set:Nn \l__datatool_ZA_datestyle_tl { ddmmyyyy }
  } ,
 date-style / mdyyyy .code: n =
  {
    \tl_set:Nn \l__datatool_ZA_datestyle_tl { mmddyyyy }
  } ,
 date-style / yyyymd .code: n =
  {
    \tl_set:Nn \l__datatool_ZA_datestyle_tl { yyyymmdd }
  } ,
 date-style / dmyy .code: n =
  {
    \tl_set:Nn \l__datatool_ZA_datestyle_tl { ddmmyy }
  } ,
 date-style / mdyy .code: n =
  {
    \tl_set:Nn \l__datatool_ZA_datestyle_tl { mmddyy }
  } ,
 date-style / yymd .code: n =
  {
    \tl_set:Nn \l__datatool_ZA_datestyle_tl { yymmdd }
  } ,
 date-variant .choice: ,
 date-variant / slash .code:n =
  {
    \tl_set:Nn \l__datatool_ZA_datevariant_tl { slash }
  } ,
 date-variant / hyphen .code:n =
  {
    \tl_set:Nn \l__datatool_ZA_datevariant_tl { hyphen }
  } ,
 date-variant / dot .code:n =
  {
    \tl_set:Nn \l__datatool_ZA_datevariant_tl { dot }
  } ,
 date-variant / dialect .code:n =
  {
    \tl_set:Nn \l__datatool_ZA_datevariant_tl
      { \l_datatool_current_language_tl }
  } ,
 time-variant .choice: ,
 time-variant / colon .code:n =
  {
    \tl_set:Nn \l__datatool_ZA_timevariant_tl { colon }
  } ,
 time-variant / dot .code:n =
  {
    \tl_set:Nn \l__datatool_ZA_timevariant_tl { dot }
  } ,
 time-variant / dialect .code:n =
  {
    \tl_if_eq:NnTF \l__datatool_ZA_timevariant_tl { dot }
     {
       \tl_set:Nn \l__datatool_ZA_timevariant_tl
        { \l_datatool_current_language_tl _dot }
     }
     {
       \tl_if_in:NnF
         \l__datatool_ZA_timevariant_tl
         { \l_datatool_current_language_tl }
        {
          \tl_set:Nn \l__datatool_ZA_timevariant_tl
           { \l_datatool_current_language_tl _colon }
        }
     }
  } ,
 time-variant / dialect-colon .code:n =
  {
      \tl_set:Nn \l__datatool_ZA_timevariant_tl
        { \l_datatool_current_language_tl _colon }
     } ,
    time-variant / dialect-dot .code:n =
     {
      \tl_set:Nn \l__datatool_ZA_timevariant_tl
        { \l_datatool_current_language_tl _dot }
     } ,
  }
```

3.13.4 Hooks

Command to update temporal parsing commands for this region:

```
 \newcommand \datatoolZASetTemporalParsers
  {
    \renewcommand \DTLCurrentLocaleParseTimeStamp
     { \datatool_ZA_parse_timestamp:NnTF }
    \renewcommand \DTLCurrentLocaleParseDate
     { \datatool_ZA_parse_date:NnTF }
    \renewcommand \DTLCurrentLocaleParseTime
     { \datatool_ZA_parse_time:NnTF }
    \let
     \DTLCurrentLocaleGetTimeZoneMap
     \datatool_ZA_get_timezone_map:n
  }
```

Set temporal formatting commands for this region. Currently this just resets to the default, but may change in future. Note that the defaults test if the applicable datetime2 command is available and will fallback on ISO if not defined. Bear in mind that the default style for datetime2 is `iso` so there won't be a noticeable difference unless the datetime2 regional setting is on.

```
 \newcommand \datatoolZASetTemporalFormatters
  {
    \let
     \DTLCurrentLocaleFormatDate
     \datatool_default_date_fmt:nnnn
    \let
     \DTLCurrentLocaleFormatTime
     \datatool_default_time_fmt:nnn
    \let
     \DTLCurrentLocaleFormatTimeZone
     \datatool_default_timezone_fmt:nn
    \let
     \DTLCurrentLocaleFormatTimeStampNoZone
     \datatool_default_timestamp_fmt:nnnnnnn
    \let
     \DTLCurrentLocaleFormatTimeStampWithZone
     \datatool_default_timestamp_fmt:nnnnnnnnn
    \renewcommand \DTLCurrentLocaleTimeStampFmtSep { ~ }
  }
```

Command to update currency and temporal parsing commands for this region:

```
\newcommand \DTLZALocaleHook
 {
  \datatoolZASetNumberChars
  \datatoolZASetCurrency
  \datatoolZASetTemporalParsers
  \datatoolZASetTemporalFormatters
```

Allow language files to reference the region:

```
  \tl_set:Nn \l_datatool_current_region_tl { ZA }
 }
```

Finished with LaTeX3 syntax.

```
\ExplSyntaxOff
```

Note that the hook is added to the captions by datatool-base not by the region file.

Index

after value, 9, 12, 16, 19, 22, 26, 29, 32, 35, 39, 42, 45, 48

before value, 9, 12, 16, 19, 22, 26, 29, 32, 35, 39, 42, 45, 48

colon value, 10, 13, 17, 20, 24, 27, 30, 33, 36, 40, 43, 46, 49
Commands
  \datatoolAUcurrencyfmt, 11
  \datatoolAUSetNumberChars, 11
  \datatoolBESetCurrency, 14
  \datatoolBESetNumberChars, 14
  \datatoolCAcurrencyfmt, 17
  \datatoolCASetNumberChars, 17
  \datatoolFKcurrencyfmt, 21
  \datatoolFKSetNumberChars, 21
  \datatoolGBcurrencyfmt, 24
  \datatoolGBSetNumberChars, 24
  \datatoolGGcurrencyfmt, 27
  \datatoolGGSetNumberChars, 27
  \datatoolGIcurrencyfmt, 31
  \datatoolGISetNumberChars, 31
  \datatoolIESetCurrency, 34
  \datatoolIESetNumberChars, 34
  \datatoolIMcurrencyfmt, 37
  \datatoolIMSetNumberChars, 37
  \datatoolJEcurrencyfmt, 40
  \datatoolJESetNumberChars, 40
  \datatoolNZcurrencyfmt, 44
  \datatoolNZSetNumberChars, 44
  \datatoolUScurrencyfmt, 47
  \datatoolUSSetNumberChars, 47
  \datatoolZAcurrencyfmt, 50
  \datatoolZASetNumberChars, 50
  \DTLAULocaleHook, 11
  \DTLBELocaleHook, 14
  \DTLCALocaleHook, 17
  \DTLFKLocaleHook, 21
  \DTLGBLocaleHook, 24
  \DTLGGLocaleHook, 27
  \DTLGILocaleHook, 31
  \DTLIELocaleHook, 34
  \DTLIMLocaleHook, 37
  \DTLJELocaleHook, 40
  \DTLNZLocaleHook, 44
  \DTLUSLocaleHook, 47
  \DTLZALocaleHook, 50
currency-symbol-position key, 9, 12, 16, 19, 22, 26, 29, 32, 35, 39, 42, 45, 48
currency-symbol-prefix key, 9, 15, 19, 22, 25, 29, 35, 38, 42, 45
currency-symbol-sep key, 9, 13, 16, 19, 22, 26, 29, 32, 35, 39, 42, 45, 49

\datatoolAUcurrencyfmt, 11
\datatoolAUSetNumberChars, 11
\datatoolBESetCurrency, 14
\datatoolBESetNumberChars, 14
\datatoolCAcurrencyfmt, 17
\datatoolCASetNumberChars, 17
\datatoolFKcurrencyfmt, 21
\datatoolFKSetNumberChars, 21
\datatoolGBcurrencyfmt, 24
\datatoolGBSetNumberChars, 24
\datatoolGGcurrencyfmt, 27
\datatoolGGSetNumberChars, 27
\datatoolGIcurrencyfmt, 31
\datatoolGISetNumberChars, 31
\datatoolIESetCurrency, 34
\datatoolIESetNumberChars, 34
\datatoolIMcurrencyfmt, 37
\datatoolIMSetNumberChars, 37
\datatoolJEcurrencyfmt, 40
\datatoolJESetNumberChars, 40
\datatoolNZcurrencyfmt, 44
\datatoolNZSetNumberChars, 44
\datatoolUScurrencyfmt, 47
\datatoolUSSetNumberChars, 47
\datatoolZAcurrencyfmt, 50
\datatoolZASetNumberChars, 50
date-style key, 10, 13, 17, 20, 23, 27, 30, 33, 36, 40, 43, 46, 49
date-variant key, 10, 13, 17, 20, 23, 27, 30, 33, 36, 40, 43, 46, 49
default value, 13
dialect value, 10, 13, 17, 20, 23, 24, 27, 30, 33, 36, 40, 43, 46, 49
dialect-colon value, 10, 13, 17, 20, 24, 27, 30, 33, 36, 40, 43, 46, 49

dialect-dot value, 10, 13, 17, 20, 24, 27, 30, 33, 36, 40, 43, 46, 49
dmyy value, 10, 13, 17, 20, 23, 27, 30, 33, 36, 40, 43, 46, 49
dmyyyy value, 10, 13, 17, 20, 23, 27, 30, 33, 36, 40, 43, 46, 49
dot value, 10, 13, 17, 20, 23, 24, 27, 30, 33, 36, 40, 43, 46, 49
\DTLAULocaleHook, 11
\DTLBELocaleHook, 14
\DTLCALocaleHook, 17
\DTLFKLocaleHook, 21
\DTLGBLocaleHook, 24
\DTLGGLocaleHook, 27
\DTLGILocaleHook, 31
\DTLIELocaleHook, 34
\DTLIMLocaleHook, 37
\DTLJELocaleHook, 40
\DTLNZLocaleHook, 44
\DTLUSLocaleHook, 47
\DTLZALocaleHook, 50

education value, 23

hyphen value, 10, 13, 17, 20, 23, 27, 30, 33, 36, 40, 43, 46, 49

Keys
- /AU/
  - currency-symbol-position, 9
  - currency-symbol-prefix, 9
  - currency-symbol-sep, 9
  - date-style, 10
  - date-variant, 10
  - number-style, 10
  - time-variant, 10
- /BE/
  - currency-symbol-position, 12
  - currency-symbol-sep, 13
  - date-style, 13
  - date-variant, 13
  - number-style, 13
  - time-variant, 13
- /CA/
  - currency-symbol-position, 16
  - currency-symbol-prefix, 15
  - currency-symbol-sep, 16
  - date-style, 17
  - date-variant, 17
  - number-style, 16
  - time-variant, 17
- /FK/
  - currency-symbol-position, 19
  - currency-symbol-prefix, 19
  - currency-symbol-sep, 19
  - date-style, 20
  - date-variant, 20
  - number-style, 19
  - time-variant, 20
- /GB/
  - currency-symbol-position, 22
  - currency-symbol-prefix, 22
  - currency-symbol-sep, 22
  - date-style, 23
  - date-variant, 23
  - number-style, 23
  - time-variant, 23
- /GG/
  - currency-symbol-position, 26
  - currency-symbol-prefix, 25
  - currency-symbol-sep, 26
  - date-style, 27
  - date-variant, 27
  - number-style, 26
  - time-variant, 27
- /GI/
  - currency-symbol-position, 29
  - currency-symbol-prefix, 29
  - currency-symbol-sep, 29
  - date-style, 30
  - date-variant, 30
  - number-style, 29
  - time-variant, 30
- /IE/
  - currency-symbol-position, 32
  - currency-symbol-sep, 32
  - date-style, 33
  - date-variant, 33
  - number-style, 32
  - time-variant, 33
- /IM/
  - currency-symbol-position, 35
  - currency-symbol-prefix, 35
  - currency-symbol-sep, 35
  - date-style, 36
  - date-variant, 36
  - number-style, 36
  - time-variant, 36
- /JE/

    - `currency-symbol-position`, 39
    - `currency-symbol-prefix`, 38
    - `currency-symbol-sep`, 39
    - `date-style`, 40
    - `date-variant`, 40
    - `number-style`, 39
    - `time-variant`, 40
  - `/NZ/`
    - `currency-symbol-position`, 42
    - `currency-symbol-prefix`, 42
    - `currency-symbol-sep`, 42
    - `date-style`, 43
    - `date-variant`, 43
    - `number-style`, 42
    - `time-variant`, 43
  - `/US/`
    - `currency-symbol-position`, 45
    - `currency-symbol-prefix`, 45
    - `currency-symbol-sep`, 45
    - `date-style`, 46
    - `date-variant`, 46
    - `number-style`, 46
    - `time-variant`, 46
  - `/ZA/`
    - `currency-symbol-position`, 48
    - `currency-symbol-sep`, 49
    - `date-style`, 49
    - `date-variant`, 49
    - `number-style`, 49
    - `time-variant`, 49

- `mdyy` value, 10, 13, 17, 20, 23, 27, 30, 33, 36, 40, 43, 46, 49
- `mdyyyy` value, 10, 13, 17, 20, 23, 27, 30, 33, 36, 40, 43, 46, 49

- `nbsp` value, 9, 13, 16, 19, 22, 26, 29, 32, 35, 39, 42, 45, 49
- `none` value, 9, 13, 16, 19, 22, 26, 29, 32, 35, 39, 42, 45, 49
- `number-style` key, 10, 13, 16, 19, 23, 26, 29, 32, 36, 39, 42, 46, 49

- `official` value, 10, 19, 23, 26, 29, 32, 36, 39, 42, 46, 49
- `old` value, 23

- `slash` value, 10, 13, 17, 20, 23, 27, 30, 33, 36, 40, 43, 46, 49
- `space` value, 9, 13, 16, 19, 22, 26, 29, 32, 35, 39, 42, 45, 49

- `thin-space` value, 9, 13, 16, 19, 22, 26, 29, 32, 35, 39, 42, 45, 49
- `thinspace` value, 13
- `time-variant` key, 10, 13, 17, 20, 23, 27, 30, 33, 36, 40, 43, 46, 49

- `unofficial` value, 10, 19, 26, 29, 32, 36, 39, 42, 46

- Values
  - `after`, 9, 12, 16, 19, 22, 26, 29, 32, 35, 39, 42, 45, 48
  - `before`, 9, 12, 16, 19, 22, 26, 29, 32, 35, 39, 42, 45, 48
  - `colon`, 10, 13, 17, 20, 24, 27, 30, 33, 36, 40, 43, 46, 49
  - `default`, 13
  - `dialect`, 10, 13, 17, 20, 23, 24, 27, 30, 33, 36, 40, 43, 46, 49
  - `dialect-colon`, 10, 13, 17, 20, 24, 27, 30, 33, 36, 40, 43, 46, 49
  - `dialect-dot`, 10, 13, 17, 20, 24, 27, 30, 33, 36, 40, 43, 46, 49
  - `dmyy`, 10, 13, 17, 20, 23, 27, 30, 33, 36, 40, 43, 46, 49
  - `dmyyyy`, 10, 13, 17, 20, 23, 27, 30, 33, 36, 40, 43, 46, 49
  - `dot`, 10, 13, 17, 20, 23, 24, 27, 30, 33, 36, 40, 43, 46, 49
  - `education`, 23
  - `hyphen`, 10, 13, 17, 20, 23, 27, 30, 33, 36, 40, 43, 46, 49
  - `mdyy`, 10, 13, 17, 20, 23, 27, 30, 33, 36, 40, 43, 46, 49
  - `mdyyyy`, 10, 13, 17, 20, 23, 27, 30, 33, 36, 40, 43, 46, 49
  - `nbsp`, 9, 13, 16, 19, 22, 26, 29, 32, 35, 39, 42, 45, 49
  - `none`, 9, 13, 16, 19, 22, 26, 29, 32, 35, 39, 42, 45, 49
  - `official`, 10, 19, 23, 26, 29, 32, 36, 39, 42, 46, 49
  - `old`, 23
  - `slash`, 10, 13, 17, 20, 23, 27, 30, 33, 36, 40, 43, 46, 49
  - `space`, 9, 13, 16, 19, 22, 26, 29, 32, 35, 39, 42, 45, 49

  - `thin-space`, 9, 13, 16, 19, 22, 26, 29, 32, 35, 39, 42, 45, 49
  - `thinspace`, 13
  - `unofficial`, 10, 19, 26, 29, 32, 36, 39, 42, 46
  - `yymd`, 10, 13, 17, 20, 23, 27, 30, 33, 36, 40, 43, 46, 49
  - `yyyymd`, 10, 13, 17, 20, 23, 27, 30, 33, 36, 40, 43, 46, 49

yymd value, 10, 13, 17, 20, 23, 27, 30, 33, 36, 40, 43, 46, 49
yyyymd value, 10, 13, 17, 20, 23, 27, 30, 33, 36, 40, 43, 46, 49